Jurnal Inovasi dan Teknologi Pembelajaran
Vol 13, No 3, (2026), page 172-186
https://doi.org/10.17977/um031v13i32026p172
P-ISSN: 2406-8780 E-ISSN: 2654-7953
Open access: https://journal-fip.um.ac.id/index.php/jinotep/index

AR-CODE-AT for analytical thinking in elementary coding education: Design, expert validation, and user trial

Alfiah Fajriani, Hendra Nelva Saputra* and Priagung Zafitrah Pamungkas

Information Technology Education Study Program, Universitas Muhammadiyah Kendari
**Corresponding author, e-mail: hendra.nelva@umkendari.ac.id*

ARTICLE INFO

Article history:
Received: 09-09-2026
Revised: 09-10-2026
Accepted: 30-10-2026

Keywords:
augmented reality;
coding education;
analytical thinking;
computational thinking;
elementary education;
instructional design.

This is an open access article under the Creative Commons Attribution-ShareAlike 4.0 International license.

Copyright © 2026 by Authors. Published by Universitas Negeri Malang.

ABSTRACT

Augmented reality (AR) can make abstract programming logic more tangible, but its educational value depends on an instructional architecture that specifies how AR supports reasoning, coding, scaffolding, and assessment. This study developed, expert-validated, and empirically piloted AR-CODE-AT, a seven-stage AR-assisted coding strategy for upper-primary education. A four-phase formative design-and-validation model was used: needs analysis, design translation, expert validation and revision, and a limited user trial. Phase 1 analysed 16 valid needs responses and showed very high development support (M = 4.55/5, SD = 0.46). Phase 2 involved six experts evaluating 48 items across nine design domains; expert validation yielded M = 4.36, SD = 0.54, and mean Aiken's V = .841 (item range = .750–.917), with 39/48 items (81.25%) reaching V ≥ .80. Phase 3 implemented the revised prototype with 15 Grade 5 students and one classroom teacher. Student practicality reached M = 3.39/4 (SD = 0.26; 84.7%), implementation fidelity was 5/7 stages (71.4%), and teacher practicality was M = 3.67/5 (73.3%). The analytical-thinking process rubric yielded M = 2.83/4 (SD = 0.54). Design-theoretically, the study specifies how AR functions, coding actions, scaffolding/fading, and process assessment are aligned across the seven stages; methodologically, it provides a traceable evidence chain from needs to expert judgment to user implementation. The pilot is descriptive and does not establish learning gains or comparative effectiveness.

INTRODUCTION

Computational thinking and coding are increasingly positioned as foundational learning experiences in primary education because they require learners to represent problems, organise procedures, test solutions, and revise errors (Bati, 2022; Falloon, 2024; Lee et al., 2022; Nordby et al., 2022). Recent syntheses show that coding can support computational thinking and related problem-solving outcomes when activities are developmentally appropriate, and instruction is deliberately structured rather than reduced to software use (Mills et al., 2025; Santaengracia et al., 2026; Su & Yang, 2023). Evidence from primary programming studies likewise shows that theory-informed sequencing and active problem solving can strengthen computational-thinking performance (Gao & Hew, 2022; Yang & Lin, 2024). In the Indonesian context, a recent Sinta 2 review also highlights coding as a promising route for computational-thinking development in primary education while emphasising the importance of pedagogy, teacher readiness, and implementation conditions (Purwono et al., 2026).

In this study, analytical thinking is operationalised as a set of observable processes that can be elicited through coding: decomposing a problem, recognising patterns, constructing a logical sequence, evaluating errors, refining a solution, and explaining the reasoning behind a chosen strategy. These processes overlap with key computational-thinking practices but are not treated as synonymous with computational thinking. The distinction is important because a student may complete a program without making the underlying reasoning visible. Reviews of primary-school critical-thinking instruction similarly emphasise the need for theoretically grounded classroom practices that make reasoning, explanation, and evaluation explicit (Jegstad et al., 2025; Leon Diaz et al., 2026). AR-CODE-AT therefore treats program products as only one source of evidence and foregrounds observable reasoning during analysis, algorithm design, debugging, and reflection.

Augmented reality offers a relevant representational bridge for this purpose (Faria & Miranda, 2024; Kalemkuş & Kalemkuş, 2023; Sakr & Abdullah, 2024). Programming concepts such as sequence, direction, loop, condition, and debugging can be difficult for novice learners because the relation between symbolic commands and their consequences is abstract. Kao & Ruan (2022) showed with fifth-grade learners that an interactive AR programming environment can make programming logic tangible and improve programming achievement, while the degree of interactivity matters for motivation and cognitive load. A recent review of immersive technologies in computer science and computational-thinking education likewise identifies visualisation as a major affordance but calls for greater attention to learning processes rather than outcomes alone (Jang et al., 2026). At the same time, AR does not automatically reduce cognitive demand: a meta-analysis indicates that its cognitive-load benefits depend on how visual information and instructional tasks are designed (Lin & Yu, 2023; Zhu et al., 2026).

Accordingly, AR must be integrated as a pedagogical mechanism, not as a decorative layer. Indonesian evidence in elementary education shows that AR can support visualisation and higher-order learning when it is embedded in an instructional approach, yet implementation is influenced by infrastructure, teacher preparedness, curriculum alignment, and access (Rusnilawati et al., 2026). A fifth-grade study also reports positive effects of AR-supported mathematics learning on critical-thinking outcomes, reinforcing the potential of interactive visualisation when it is tied to a reasoning task (Akbar et al., 2025). These findings imply that a viable AR coding strategy should specify what students do with AR, what reasoning the interaction is intended to trigger, and how teachers mediate the activity.

Scaffolding and debugging are particularly vital for turning coding activity into visible analytical reasoning. Elementary learners benefit when instructional support helps them externalise abstract thinking, while support is gradually reduced as they become more capable of regulating their own problem-solving (Gao et al., 2025; Tawfik et al., 2024). Debugging is not simply a technical correction step; it can function as an analytical cycle in which learners compare predicted and observed behaviour, locate the source of an error, revise an instruction, and justify the revision. Empirical work with elementary students shows that systematic debugging instruction can improve debugging performance while managing cognitive load (Gao & Hew, 2023). Teacher digital competence remains central because the quality of technology integration depends on pedagogical decisions, not only on operational familiarity with devices (Claro et al., 2024).

AR-CODE-AT is therefore organised around four linked theoretical commitments. First, constructivist and experiential learning position students as active interpreters of contextual problems rather than passive users of AR. Second, cognitive-load management requires task-relevant and segmented visual information because the benefits of AR can be offset by extraneous detail (Lin & Yu, 2023; Zhu et al., 2026). Third, scaffolding and fading regulate how support is provided to novice learners and then withdrawn as learner control increases (Gao et al., 2025; Tawfik et al., 2024). Fourth, process-oriented assessment treats intermediate artifacts and explanations as evidence of reasoning instead of equating a successful program with analytical thinking (Jegstad et al., 2025; Mills et al., 2025). These commitments yield four design propositions: AR should externalise otherwise abstract relations; coding should require prediction, execution, and revision; support should fade as learners become more independent; and assessment should capture decomposition, sequencing, error evaluation, solution refinement,

and explanation. The seven AR-CODE-AT stages operationalise these propositions as an integrated instructional architecture.

Despite the growing literature on AR, coding, and computational thinking, three design problems remain under-specified for upper-primary contexts. First, AR studies often report achievement or engagement without tracing the pedagogical function of AR across a complete learning sequence. Second, coding studies frequently measure computational outcomes but provide less detail about how decomposition, error evaluation, solution refinement, and reflective explanation are elicited and assessed as process evidence. Third, model implementation can fail when teacher scaffolding, device constraints, curriculum fit, and assessment resources are treated as secondary issues. These gaps are particularly relevant to Indonesian elementary classrooms, where recent AR and coding studies also point to practical readiness and instructional-alignment challenges (Izzati et al., 2026; Purwono et al., 2026; Rusnilawati et al., 2026).

Building on the needs-analysis stage, this study moves from identifying a need to specifying, validating, and formatively testing an instructional-design prototype. It addresses four research questions: (1) What pedagogical, technological, and cognitive requirements emerge from the needs analysis for AR-assisted coding in elementary education? (2) How can those requirements be translated into a coherent seven-stage AR-CODE-AT learning strategy in which AR, coding, scaffolding, analytical-thinking indicators, and assessment are aligned? (3) To what extent do instructional-design experts support the content and design validity of the prototype, and what revisions do they prioritise? (4) How practical and usable is the revised prototype during a limited small-group trial, and what preliminary evidence of analytical-thinking processes is observable in students' work and classroom activity? The study does not test comparative effectiveness; the user trial is descriptive and is intended to establish preliminary feasibility, usability, and process evidence before a larger field evaluation.

METHODS

Research design

This study used a study-specific formative design-and-validation development model with four linked phases: (1) needs analysis, (2) design translation, (3) expert validation and revision, and (4) limited user trial and refinement. Each phase served a distinct design decision. Phase 1 identified pedagogical, technological, and cognitive constraints; the design-translation phase converted those constraints into the seven-stage AR-CODE-AT architecture; Phase 2 examined content relevance, coherence, and implementation logic through structured expert judgment; and Phase 3 examined practicality, implementation fidelity, and the visibility of intended analytical-thinking processes with intended users. This formative sequence was selected because the research question concerns the quality and usability of a developing instructional architecture rather than the summative effectiveness of a finished intervention. Consequently, no causal claim about student learning effectiveness was tested.

Phase 1: Needs analysis

The needs-analysis dataset contained 20 submitted responses. Four entries were clearly labelled as test responses and were removed before analysis, leaving 16 valid adult respondents: five educational technology experts, six coding practitioners, and five elementary school teachers (see Table 1). Each group completed a 20-statement questionnaire adapted to its role. Across the instruments, the statements addressed the need for elementary coding, AR-supported visualisation and interaction, developmentally appropriate learning sequences, teacher and technology readiness, and the use of coding activities to strengthen analytical-thinking processes. Responses used a five-point agreement scale. Because the phase was descriptive, the heterogeneous subgroup sizes were small, and the purpose was requirement mapping rather than group comparison, the analysis focused on means and overall respondent-level standard deviation; no inferential between-group test was conducted.

Prototype development and implementation specification

The Phase 1 requirements were translated into AR-CODE-AT (Augmented Reality-Assisted Coding Learning Strategy for Analytical Thinking). The model is intended for upper-primary learners; to make the next trial operational, the current specification delimits the initial pilot

scope to Grades 4–6 and to visual/block-based coding tasks involving sequence, simple repetition, conditions, and debugging. AR is defined as a representational and feedback mechanism that links a contextual problem to visible objects, routes, constraints, and consequences of commands. Analytical thinking is operationalised through six process indicators: problem decomposition, pattern recognition, logical sequencing, error evaluation, solution refinement, and reflective explanation.

Table 1. Needs-analysis sample profile

Respondent group	Raw responses	Removed test entries	Valid(n)	Items
Educational technology experts	6	1	5	20
Coding practitioners	7	1	6	20
Elementary school teachers	7	2	5	20
Total	20	4	16	—

The revised design also makes scaffolding and timing explicit. Three support levels are specified: (a) high support during problem orientation, AR exploration, and early decomposition through demonstrations, visual cues, and worked prompts; (b) adaptive support during algorithm design, coding, and debugging through questions and partial hints after students attempt a solution; and (c) faded support during reflection and transfer, where students are expected to explain and adapt strategies with minimal prompting. For pilot planning, the seven stages can be distributed across two 50-minute sessions (100 minutes total): approximately 10 minutes for orientation, 15 minutes for AR exploration, 25 minutes for analysis and algorithm design, 30 minutes for coding and debugging, 10 minutes for peer evaluation, and 10 minutes for reflection. These times are design estimates, not evidence of classroom feasibility; they must be calibrated in the limited pilot.

Phase 2: Expert design validation

Six experts independently evaluated the prototype. They represented five higher-education institutions; five validators were external to the authors' home institution and one was internal. The panel combined educational/learning-technology expertise with learning-media evaluation and computing-related expertise (see Table 2).

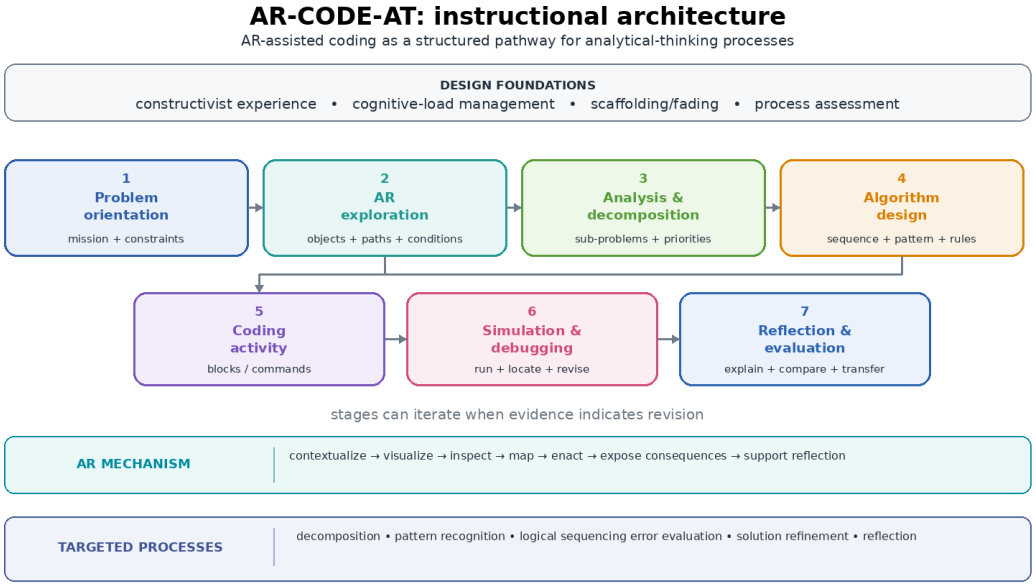


Figure 1. AR-CODE-AT instructional architecture and targeted analytical-thinking processes

Table 2. Expert-panel profiles

Code	Institution	Primary expertise	Qualification	Status
E1	Universitas Muhammadiyah Makassar	Learning-media development and evaluation	Master's	External
E2	Universitas Muhammadiyah Kendari	Educational technology	Master's	Internal
E3	Universitas Pasir Pengaraian	Computer vision, data mining, mobile programming, deep learning	Master's	External
E4	Universitas Negeri Malang	Educational technology	Doctorate	External
E5	Institut Agama Kristen Negeri Ambon	Learning technology	Master's	External
E6	Universitas Muhammadiyah Makassar	Educational technology (learning)	Doctorate	External

Four validators held master's degrees and two held doctoral degrees. To protect personal identity in the manuscript while preserving methodological transparency, validators are reported using codes E1-E6 and by institutional affiliation, expertise, qualification, and internal/external status.

The validation instrument contained 48 statements organised into nine domains: rationale and contribution (5 items), theoretical foundation (6), objectives and alignment (4), learning syntax (7), teacher–student roles (6), pedagogical AR integration (5), analytical-thinking integration (6), developmental feasibility and consistency (6), and distinctiveness, novelty, and continuation (3). Experts rated each statement on a five-point agreement scale (1 = strongly disagree to 5 = strongly agree) and provided open-ended recommendations. The instrument therefore generated 288 item-level ratings (6 experts × 48 items) plus qualitative comments.

Phase 3: Limited user trial

After the expert-recommended revisions were incorporated, the revised AR-CODE-AT prototype was implemented in a limited small-group classroom trial with 15 Grade 5 students and one classroom teacher at one elementary school in Kendari, Indonesia. The students had no prior coding or AR experience. The pilot consisted of two sessions totaling 100 minutes and used an individual-device arrangement (one device per student). The purpose was to examine practicality, usability, implementation feasibility, and the visibility of analytical-thinking processes. The pilot did not include a pretest–posttest learning-outcome measure or a comparison condition; therefore, it was not designed to estimate learning gains or effectiveness.

Four sources of user-trial evidence were collected: (1) a student response questionnaire using a four-point scale to assess instruction clarity, ease of AR use, ease of coding/debugging, engagement, and perceived support for reasoning; (2) a seven-stage implementation-fidelity checklist completed by the principal researcher acting as observer; (3) the six indicator analytical thinking process rubric embedded in the model problem decomposition, pattern recognition, logical sequencing, error evaluation, solution refinement, and reflective explanation scored from 1 to 4 using students' problem maps, algorithms, code, debugging records, reflections, and classroom observations; and (4) a teacher practicality questionnaire using a five-point scale addressing lesson flow, time, device management, scaffolding, assessment, and overall feasibility. Open-ended field notes documented timing, classroom-management constraints, scaffolding demands, and student difficulties. These instruments were used as formative implementation measures, not standardised learning-outcome tests. The implementation checklist retained the operational labels used in the classroom guide; for reporting against the current seven-stage architecture, the combined decomposition-and-algorithm activity was mapped to current Stages 3 and 4, while collaborative evaluation and reflection/transfer were treated as evidence for current Stage 7.

Data analysis

Needs-analysis responses were summarised descriptively. To avoid presenting an unavailable subgroup statistic, Table 3 reports subgroup means only; the overall respondent-level variability is reported in the text (SD = 0.46). For transparent interpretation of the 1–5 needs scale, equal-width descriptive categories were used: 1.00–1.80 = very low, 1.81–2.60 = low, 2.61–3.40 = moderate, 3.41–4.20 = high, and 4.21–5.00 = very high. These labels describe the level of development need and should not be interpreted as product-feasibility criteria. For expert validation, the analysis used domain and overall means, pooled standard deviations of item ratings, the percentage of the maximum possible score, and Aiken's V as an index of expert agreement on item relevance/appropriateness. For each item, V was calculated as $\Sigma(r - l_0)/[n(c - 1)]$, where r is the expert rating, $l_0 = 1$ is the lowest possible rating, n = 6 is the number of experts, and c = 5 is the number of response categories. Recent Indonesian measurement research demonstrates the use of Aiken's V to provide transparent content-validity evidence based on expert judgments (Kania et al., 2024).

For transparent design decisions, $V \geq 0.80$ was treated as strong expert convergence, V = 0.60–0.79 as adequate but revision-indicating agreement, and $V < 0.60$ as a signal for major revision. These bands were used as practical design rules rather than inferential significance cut-offs. The distribution of item-level V values was also reported, so the overall coefficient did not mask weaker items. Cronbach's alpha was computed as a secondary descriptive check of consistency across the 48 ratings ($\alpha = 0.983$), but it was not treated as standalone psychometric evidence because only six judges participated. Open-ended comments were condensed into revision themes and linked directly to changes in the prototype specification. For Phase 3, student questionnaire scores were summarised using means, standard deviations, and percentages of the maximum possible score; teacher practicality was summarised using the mean and percentage of maximum; implementation fidelity was reported as the number and percentage of stages meeting the intended criterion; and analytical-thinking rubric scores were summarised using means, standard deviations, and counts of ratings at levels 3–4. Because the pilot involved 15 students with no baseline measure and no comparison group, it could not calculate the inferential tests, gain scores, or effect sizes. The Phase 3 rubric therefore indicates concurrent observability of analytical processes during implementation, not improvement attributable to AR-CODE-AT.

After the expert-recommended revisions were incorporated, the revised AR-CODE-AT prototype was implemented in a limited small-group classroom trial with 15 Grade 5 students and one classroom teacher at one elementary school in Kendari, Indonesia. The students had no prior coding or AR experience. The pilot was conducted across two sessions totaling 100 minutes and used an individual-device arrangement (one device per student). The purpose was to examine practicality, usability, implementation feasibility, and the visibility of analytical-thinking processes; it was not designed as an effectiveness experiment.

Four sources of user-trial evidence were collected: (1) a student response questionnaire using a four-point scale to assess instruction clarity, ease of AR use, ease of coding/debugging, engagement, and perceived support for reasoning; (2) a seven-stage implementation-fidelity checklist completed by the principal researcher acting as observer; (3) the six-indicator analytical-thinking process rubric embedded in the model—problem decomposition, pattern recognition, logical sequencing, error evaluation, solution refinement, and reflective explanation—scored from 1 to 4 using students' problem maps, algorithms, code, debugging records, reflections, and classroom observations; and (4) a teacher practicality questionnaire using a five-point scale addressing lesson flow, time, device management, scaffolding, assessment, and overall feasibility. Open-ended field notes documented timing, classroom-management constraints, scaffolding demands, and student difficulties. The implementation checklist retained the operational labels used in the classroom guide; for reporting against the current seven-stage architecture, the combined decomposition-and-algorithm activity was mapped to current Stages 3 and 4, while collaborative evaluation and reflection/transfer were treated as evidence for current Stage 7.

RESULTS

Needs analysis: converging design requirements

The 16 valid needs-analysis responses unveiled a very high development need for an AR-assisted coding learning strategy. The overall respondent-level mean was 4.55 out of 5 (SD = 0.46). Educational technology experts reported M = 4.58, coding practitioners M = 4.52, and elementary school teachers M = 4.70; all fell within the predefined 4.21–5.00 'very high' category. Because Phase 1 was intended to map requirements rather than test subgroup differences, these means are interpreted descriptively and are not treated as feasibility or effectiveness evidence.

Table 3. Needs-analysis results by respondent group

Respondent group	Valid (n)	Mean (1–5)	Development
Educational technology experts	5	4.58	Very High
Coding practitioners	6	4.52	Very High
Elementary school teachers	5	4.70	Very High
Overall	16	4.55	Very High

The content of the subgroup responses converged on a coherent set of requirements. Educational technology experts emphasised experiential learning, cognitive engagement, a structured learning flow, and explicit analytical-thinking indicators. Coding practitioners strongly endorsed interactive visualisation, simulation, problem-solving, gradual difficulty, and developmental appropriateness; they were comparatively more cautious about claiming that AR would be more effective than conventional instruction before experimental testing. Teachers prioritised practical, step-by-step guidance, visual-interactive learning, and implementation support. These convergent needs became the design constraints for the prototype: the strategy had to be visual and concrete, problem-oriented, staged, interactive, reflective, feasible for teachers, and explicit about how coding activity trains analytical processes.

Translation of requirements into the seven-stage AR-CODE-AT syntax

The prototype operationalises the needs findings in a seven-stage cycle. A key revision in the current version is that AR now has a traceable function beyond the exploration stage. It introduces the mission and constraints, exposes relevant visual information, supports decomposition by allowing students to revisit and inspect the scene, anchors algorithmic decisions to positions and directions, provides visible consequences of code, and supplies evidence for debugging and reflection. This stage-by-stage specification directly addresses the expert concern that an AR-assisted model should make the pedagogical function of AR visible throughout the learning process.

The revised model also specifies a fading rule for teacher assistance. Full prompts and visual cues are concentrated in the first three stages; in algorithm design and coding, the teacher first gives students time to attempt a solution and then uses partial hints or questions; during debugging and reflection, support is reduced to prompts that require learners to locate evidence and explain their own reasoning (see Table 4). Thus, the model aims to prevent scaffolding from becoming answer-giving. The process products problem map, algorithm, initial code, debugging log, revised solution, and reflection also create a chain of evidence that can later be used for process-oriented assessment.

Expert validation of the instructional design

The six-expert panel produced 288 item-level ratings. Across all ratings, the prototype obtained M = 4.36 (SD = 0.54), equal to 87.29% of the maximum possible score. The mean Aiken's V across the 48 items was 0.841, with item-level coefficients ranging from 0.750 to 0.917. Thirty-nine items (81.25%) reached V ≥ 0.80. The remaining nine items fell between 0.750 and 0.792; no item was below 0.750. This pattern indicates strong overall convergence while also identifying a limited set of items that required targeted design clarification rather than wholesale rejection of the model (see Table 5).

Table 4. Revised AR-CODE-AT syntax, AR function, and analytical-thinking evidence

Stage	Core learner activity	Pedagogical function of AR/coding	Observable analytical-thinking evidence
1. Problem orientation	Interpret a contextual mission and identify the goal and constraints.	AR preview situates the mission in a concrete scene and makes the goal/constraints visible.	Problem statement; identification of relevant information and initial constraints.
2. AR exploration	Scan the marker and inspect objects, paths, directions, conditions, and obstacles.	AR externalises spatial and causal information that would otherwise remain abstract.	Observation notes; recognition of relevant objects, relationships, and possible patterns.
3. Analysis & decomposition	Break the mission into sub-goals and identify what must be solved first.	Students pause/revisit the AR scene and connect visual components to sub-problems.	Problem map; prioritised sub-goals; explanation of why the decomposition is reasonable.
4. Algorithm design	Plan a sequence and identify repeated or conditional actions before coding.	AR positions and directions are mapped to commands; students predict what the planned route will do.	Flowchart/ordered commands; pattern or loop identification; predicted outcome.
5. Coding activity	Translate the algorithm into visual/block commands and inspect command–action correspondence.	Coding turns the plan into executable instructions linked to intended AR actions.	Initial code; justification of command order and relation to the algorithm.
6. Simulation & debugging	Run the solution, compare predicted and observed behaviour, locate errors, revise, and retest.	AR displays the consequences of commands, making mismatches and error locations visible.	Debugging log; error diagnosis; revised code; explanation of the correction.
7. Reflection & evaluation	Explain why the solution works, compare alternatives, and transfer the strategy to a new case.	AR replay/screens provide shared evidence for explanation and evaluation rather than entertainment.	Reasoned explanation; comparison of alternatives; refined solution; transfer statement.

Table 5. Expert-validation results by design domain (n = 6)

Design domain	Items	M	SD	Mean Aiken's V	Items V ≥ 0.80
Rationale and contribution	5	4.40	0.50	0.850	4/5
Theoretical foundation	6	4.47	0.51	0.868	6/6
Objectives and alignment	4	4.29	0.55	0.823	3/4
Learning syntax	7	4.31	0.52	0.827	6/7
Teacher–student roles	6	4.44	0.56	0.861	5/6
Pedagogical AR integration	5	4.27	0.58	0.817	2/5
Analytical thinking integration	6	4.31	0.52	0.826	5/6
Developmental feasibility and consistency	6	4.36	0.59	0.840	5/6
Distinctiveness, novelty, and continuation	3	4.44	0.51	0.861	3/3

The theoretical foundation obtained the strongest domain result (M = 4.47, V = 0.868), followed by teacher–student roles and distinctiveness/novelty (both M = 4.44, V = 0.861). Pedagogical AR integration had the lowest mean and V among the nine domains (M = 4.27, V = 0.817), although it remained above the study's strong-convergence rule at the domain level. Its item-level pattern was more diagnostic: only two of five AR-integration items reached V ≥ 0.80, whereas the remaining three were V = 0.792. The lowest item-level values across the full instrument (V = 0.750) concerned alignment with concrete/visual/staged needs, observability of analytical outcomes, evaluation of error through simulation/debugging, and feasibility under limited-facility conditions. These results explain why the revisions focus on pedagogical traceability, assessment evidence, and implementation constraints.

The secondary internal-consistency estimate was α = 0.983. This high value indicates that the six experts used the 48-item rating set consistently, but it should be interpreted cautiously because the number of judges is small. Besides, the items intentionally cover multiple design domains. Accordingly, the substantive interpretation rests on Aiken's V, domain patterns, item-level distributions, and qualitative comments rather than on alpha alone (see Figure 2).

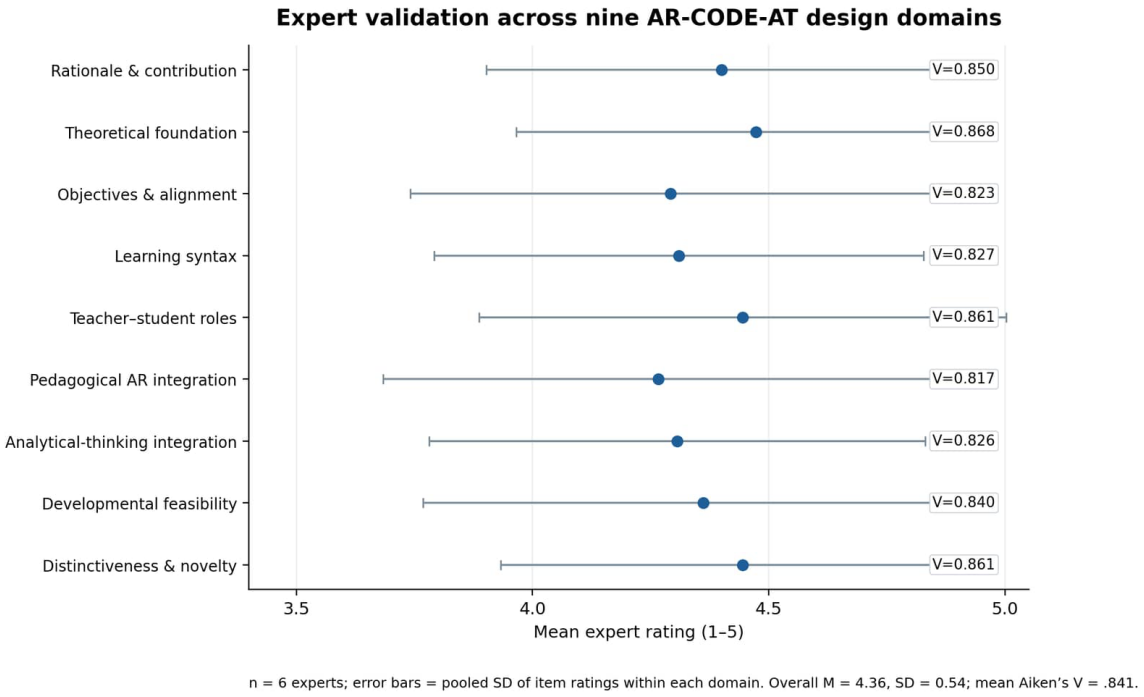


Figure 2. Expert-validation ratings across nine AR-CODE-AT design domains. Error bars show the pooled SD of item ratings within each domain

Expert comments and revision actions

Open-ended feedback was consistent with the quantitative profile. Experts did not challenge the central direction of the model, but they requested a more operational specification for classroom use. Table 6 summarises the six most consequential themes and the changes implemented in this revised manuscript and prototype description.

Limited user trial: practicality, usability, and analytical-thinking process evidence

Following expert-directed revision, the prototype was trialed with 15 Grade 5 students and one classroom teacher. The actual duration lasted for 100 minutes across two sessions. After mapping the classroom checklist to the current syntax, five of seven stages met the intended implementation criterion, corresponding to 71.4% fidelity. Coding construction and simulation/debugging did not meet the intended fidelity criterion because students required substantial support to externalise their thinking into code and to understand debugging. Classroom management and the limited 100-minute allocation were also identified as implementation constraints.

Table 6. Expert recommendations and revisions implemented in the current prototype specification

Revision theme	Expert concern	Revision implemented
Pedagogical role of AR	AR was clear during exploration but insufficiently explicit during decomposition, coding, debugging, evaluation, and reflection.	Table 4 now specifies the AR action and pedagogical function at every stage, including use of AR evidence for decomposition, prediction, debugging, and reflection.
Grade and curriculum scope	The model needed clearer target-grade and curriculum boundaries.	The initial pilot is delimited to upper-primary Grades 4–6 and visual/block-based coding concepts (sequence, repetition, conditions, debugging); curriculum alignment must be checked against the target school before trial.
Timing and implementation readiness	Experts requested realistic duration, complete media/assessment resources, and guidance when devices fail.	A provisional two-session, 100-minute plan is stated; the teacher guide is specified to include device/marker pre-checks, offline or shared-device contingencies, mission cards, algorithm sheets, debugging logs, and assessment rubrics.
Cognitive-load control	AR displays could overburden working memory if information is presented simultaneously or decoratively.	The design now requires segmenting AR information into task-relevant cues, consistent icons, short instructions, and removal of nonessential animation; pilot observation must monitor overload.
Scaffolding and fading	The guide did not clearly state when support should be reduced.	A three-level support rule is specified: explicit support in early orientation/decomposition, adaptive hints during algorithm/coding/debugging, and minimal prompts during reflection/transfer.
Conceptual coherence and terminology	Experts requested a clearer relationship among theory, syntax, roles, analytical thinking, and assessment, plus consistent use of coding terminology.	Figure 1 integrates the conceptual architecture; stage outputs are linked to analytical-thinking evidence; the English manuscript consistently uses “coding” and “AR-CODE-AT.”

Table 7. Limited user-trial practicality and usability results

User-trial indicator	n	Scale	M	SD	Percentage of maximum
Overall student response	15	1–4	3.39	0.26	84.7%
Instruction clarity	15	1–4	3.40	0.51	85.0%
Ease of AR use	15	1–4	3.67	0.49	91.7%
Ease of coding/debugging	15	1–4	3.40	0.51	85.0%
Engagement	15	1–4	3.27	0.46	81.7%
Reasoning support	15	1–4	3.20	0.41	80.0%
Teacher practicality	1	1–5	3.67	—	73.3%

Table 7 shows the student responses that indicated generally positive practicality (M = 3.39/4, SD = 0.26; 84.7% of the maximum), and no student response fell below 3 on any of the five questionnaire aspects. Ease of AR use received the highest rating (M = 3.67/4, SD = 0.49; 91.7%), whereas perceived support for reasoning was comparatively lowest (M = 3.20/4, SD = 0.41; 80.0%). The classroom teacher rated the strategy positively overall (M = 3.67/5; 73.3%), noting

good command of the lesson stages while identifying classroom management and scaffolding implementation as areas needing further refinement. These findings are preliminary because the trial involved one small class and no comparison condition.

Table 8. Analytical-thinking process evidence observed in the limited user trial

Process indicator	Scale	M	SD	Observed evidence / upper-level attainment
Problem decomposition	1–4	2.87	0.74	15/15 produced a decomposed representation; 10/15 scored 3–4
Pattern recognition	1–4	2.67	0.62	9/15 scored 3–4
Logical sequencing	1–4	2.93	0.70	11/15 scored 3–4
Error evaluation	1–4	3.00	0.76	11/15 scored 3–4
Solution refinement	1–4	2.53	0.64	8/15 revised code after an identified error; 7/15 scored 3–4
Reflective explanation	1–4	3.00	0.65	12/15 scored 3–4; 7/15 explained the solution with minimal prompting
Overall	1–4	2.83	0.54	60/90 indicator ratings (66.7%) were at levels 3–4

The analytical-thinking process rubric yielded an overall mean of 2.83/4 (SD = 0.54). Error evaluation and reflective explanation were the highest-rated indicators (both M = 3.00), followed by logical sequencing (M = 2.93), while solution refinement was lowest (M = 2.53). All 15 students produced a decomposed problem representation; 8 of 15 revised their code after identifying an error, and 7 of 15 could explain why the final solution worked with minimal prompting. At the indicator level, 10/15 students scored 3–4 for problem decomposition, 9/15 for pattern recognition, 11/15 for logical sequencing, 11/15 for error evaluation, 7/15 for solution refinement, and 12/15 for reflective explanation. These observations show that the model elicited the intended analytical processes to varying degrees in the limited setting, but they do not establish that AR-CODE-AT caused improvement in analytical thinking.

Importantly, Phase 3 did not include a learning-outcome improvement test. No pretest–posttest achievement measure or comparison condition was administered. Accordingly, the scores in Table 8 describe the level at which the intended analytical-thinking processes were observable during the pilot; they are not gain scores and should not be interpreted as evidence that AR-CODE-AT improved learning outcomes.

Discussion

The study contributes a three-layer formative evidence chain for AR-CODE-AT: need identification, expert-reviewed design coherence, and limited user-based implementation evidence. Phase 1 established a very high development need (M = 4.55), but need alone does not establish an instructional contribution. The design-translation phase therefore made explicit how those requirements became problem orientation, AR exploration, decomposition, algorithm design, coding, debugging, and reflection. Phase 2 then examined whether that architecture was coherent and relevant (M = 4.36; mean Aiken's V = 0.841), while Phase 3 tested whether the revised architecture could be enacted and generate observable process evidence in a Grade 5 setting. This sequence is methodologically influential because each evidence source answers a distinct design question rather than treating expert approval, user satisfaction, or a single score as interchangeable forms of 'validation.'

The strongest interpretation is not that "AR works," but that AR-CODE-AT gives AR a pedagogically constrained role (Mustika et al., 2026). This distinction is supported by current literature. Kao & Ruan (2022) found that interactive AR can make programming logic tangible for fifth graders, yet the level of interaction influenced motivation and cognitive load. Jang et al. (2026) similarly argue that immersive computer-science research has concentrated more on outcomes than on learning processes. Zhu et al. (2026) further show that AR's cognitive-load effects depend on design characteristics. The expert panel's comparatively lower rating for

pedagogical AR integration therefore identifies a genuine design risk: if AR is visible only as a technological feature, the model can become visually attractive without making analytical reasoning more observable. The revised syntax responds by assigning AR a different function at each stage, from contextualisation and inspection to prediction, execution feedback, error localisation, and evidence-supported reflection.

This process orientation also sharpens the relationship between coding and analytical thinking. Current K–6 reviews show that computational-thinking implementation often narrows broad theoretical constructs to programming or algorithmic thinking in classroom practice (Santaengracia et al., 2026), while broader coding research shows positive cross-curricular outcomes but substantial variation in task design and measurement (Mills et al., 2025). AR-CODE-AT addresses this issue by not treating successful code as sufficient evidence. A problem map evidences decomposition; an algorithm evidences logical sequencing and pattern recognition; a debugging record captures error evaluation; a revised solution captures refinement; and reflection requires explanation of why a solution works. This chain of process evidence is also consistent with primary critical-thinking literature that emphasises explanation, evaluation, dialogic activity, and theoretically grounded assessment rather than outcome scores alone (Jegstad et al., 2025; Leon Diaz et al., 2026).

The model's emphasis on scaffolding and debugging is likewise supported by recent elementary programming research. Gao et al. (2025) reveal that scaffolding can help learners externalise computational-thinking processes in block-based programming. Tawfik et al. (2024) emphasise the learner-experience design of block-coding scaffolds, and (Gao & Hew, 2023) demonstrate that systematic debugging instruction can improve elementary learners' debugging performance while optimising cognitive load. The expert request for explicit fading is therefore consequential. In the revised AR-CODE-AT specification, support is not a constant teacher behaviour: it is intentionally reduced as learners move from interpreting a new mission toward explaining and transferring their own solution strategy. This makes teacher assistance part of the model's mechanism rather than an unspecified classroom variable.

Implementation feasibility remains the area in which design validity must be distinguished most carefully from classroom practicality (Usfinit et al., 2025). The lowest item-level validity signals included feasibility under limited-facility conditions, and experts requested time estimates, device contingencies, curriculum alignment, and ready-to-use assessment tools. These concerns align with Indonesian AR studies reporting that access, infrastructure, teacher readiness, and curriculum integration influence whether AR innovations can move from a prototype to routine classroom use (Izzati et al., 2026; Latifah et al., 2026; Rusnilawati et al., 2026). They also align with the broader evidence that teacher digital competence is multidimensional and strongly related to how teachers plan and orchestrate learning activities, not merely to device operation (Claro et al., 2024). For that reason, the two-session schedule and device-sharing contingencies in the revised design are explicitly provisional and require testing rather than being presented as proven feasibility.

The validation statistics should also be interpreted at the appropriate level. Recent Indonesian work demonstrates the usefulness of Aiken's V for making expert-judgment evidence transparent during instrument and design development (Kania et al., 2024). In the present study, reporting the mean V alone would have been insufficient. The item distribution shows that 81.25% of items reached $V \geq 0.80$, while nine items remained at 0.750–0.792. Those lower values are not hidden; they signal revisions. This approach is preferable to treating a single global percentage as proof that the model is "valid" in all respects. Similarly, $\alpha = 0.983$ is reported only as secondary descriptive information because six experts are too few to support strong psychometric claims about internal consistency.

The limited user trial provides an empirical bridge between expert judgment and classroom feasibility requested during initial screening. Student practicality reached 3.39/4 (84.7%), implementation fidelity was 71.4%, and teacher practicality was 3.67/5 (73.3%). The process rubric yielded 2.83/4, with error evaluation and reflective explanation more readily demonstrated than solution refinement. The value of these results lies less in their magnitude than in showing that the revised design can generate observable user interactions, analytical artifacts,

debugging behaviour, and reflection in an authentic Grade 5 setting. This is consistent with the model's process orientation and with prior evidence that interactive AR, structured scaffolding, and systematic debugging can make otherwise abstract programming relations more visible to elementary learners (Gao et al., 2025; Gao & Hew, 2022; Kao & Ruan, 2022). At the same time, the coding and debugging stages did not meet the intended fidelity criterion, and the 100-minute schedule was insufficient for fully completing one topic. These issues should be treated as design feedback for the next iteration rather than as evidence against the model.

Taken together, the contribution is not an effectiveness claim. Design-theoretically, AR-CODE-AT specifies a mechanism-to-evidence chain: AR externalises contextual and command–consequence relations; coding and debugging create opportunities for prediction, error diagnosis, and revision; scaffolding is intentionally faded; and intermediate artifacts make six analytical-thinking processes observable. Methodologically, the study demonstrates a transparent formative sequence from needs mapping to design translation, expert agreement, revision, and user-trial evidence, while treating lower validity values and implementation failures as revision signals rather than suppressing them. Practically, the pilot identifies where the architecture remains fragile, coding construction, simulation/debugging, reasoning support, and time allocation. This explicit link between mechanism, activity, and evidence is the central contribution of the study.

The study has several limitations. First, the needs-analysis sample was small and intentionally heterogeneous, so subgroup means should not be interpreted as population estimates or compared inferentially. Second, the expert panel included six validators, including one from the authors' institution, although five were external. Third, the user trial involved only 15 Grade 5 students, one teacher, and one school, and implementation fidelity was observed by the principal researcher; these conditions limit transferability and introduce possible observer bias. Fourth, the pilot had no pretest, posttest, comparison condition, or standardised learning-outcome measure, so the analytical-thinking rubric scores cannot be interpreted as learning gains or causal effects. Fifth, fidelity reached only 71.4%, and the 100-minute allocation was insufficient for fully completing the coding and debugging sequence. Future work should refine those stages, extend implementation time, use independent or multiple observers to establish fidelity agreement, replicate the model across schools, and then test analytical-thinking and coding outcomes with pretest–posttest and appropriate comparison designs.

CONCLUSION

This study establishes AR-CODE-AT as a design-theoretic instructional architecture rather than as evidence that AR is inherently effective. The central theoretical finding is that AR becomes pedagogically meaningful when its representational function is coordinated with problem decomposition, algorithm design, coding, debugging, fading scaffolds, and process-oriented assessment; AR-CODE-AT operationalises this relationship across seven stages. Methodologically, the study contributes a formative evidence chain from needs analysis ($M = 4.55$, $SD = 0.46$), through expert validation (mean Aiken's $V = 0.841$; 39/48 items ≥ 0.80), to a limited user trial showing positive student practicality (84.7%), partial implementation fidelity (71.4%), and observable analytical processes ($M = 2.83/4$). However, the small, heterogeneous needs sample; six-expert panel; single-class pilot; researcher observation; partial fidelity; and absence of pretest–posttest or comparison data limit generalisation and preclude causal claims about learning improvement. Future studies should revise the coding/debugging stages and timing, test the model across schools with independent fidelity scoring, and then use pretest–posttest and comparison designs to determine whether the architecture improves analytical-thinking and coding outcomes.

Acknowledgments

The authors gratefully acknowledge the Muhammadiyah Research Grant (RisetMu) 2025/2026 for providing financial support for this research under Contract Number: 0259.841/I.3/D/2025. The authors also thank the Directorate of Research, Technology, and Community Service (DRTPM), Universitas Muhammadiyah Kendari, for its facilitation, assistance,

and administrative support throughout the implementation of the research and the preparation of this manuscript.

Author contributions

AF Conceptualisation, Writing - Original Draft, Editing and Visualisation. HNS Writing - Review & Editing, Formal analysis, and Methodology. PZP Validation and Supervision.

Conflict of interest

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

Data availability statement

All data are available from the authors.

REFERENCES

Akbar, A., Herman, T., Suryadi, D., Mursalim, Alman, Putra, E. D., & Blegur, J. (2025). Integrating augmented reality in mathematics learning to improve critical thinking skills of elementary school students. *Emerging Science Journal*, *9*(2), 764–779. https://doi.org/10.28991/ESJ-2025-09-02-014

Bati, K. (2022). A systematic literature review regarding computational thinking and programming in early childhood education. *Education and Information Technologies*, *27*(2), 2059–2082. https://doi.org/10.1007/s10639-021-10700-2

Claro, M., Castro-Grau, C., Ochoa, J. M., Hinostroza, J. E., & Cabello, P. (2024). Systematic review of quantitative research on digital competences of in-service school teachers. *Computers & Education*, *215*, 105030. https://doi.org/10.1016/j.compedu.2024.105030

Falloon, G. (2024). Advancing young students' computational thinking: An investigation of structured curriculum in early years primary schooling. *Computers & Education*, *216*, 105045. https://doi.org/10.1016/j.compedu.2024.105045

Faria, A., & Miranda, G. L. (2024). Augmented reality in natural sciences and biology teaching: Systematic literature review and meta-analysis. *Emerging Science Journal*, *8*(4), 1666–1685. https://doi.org/10.28991/ESJ-2024-08-04-025

Gao, X., & Hew, K. F. (2022). Toward a 5E-based flipped classroom model for teaching computational thinking in elementary school: Effects on student computational thinking and problem-solving performance. *Journal of Educational Computing Research*, *60*(2), 512–543. https://doi.org/10.1177/07356331211037757

Gao, X., & Hew, K. F. (2023). A flipped systematic debugging approach to enhance elementary students' program debugging performance and optimize cognitive load. *Journal of Educational Computing Research*, *61*(5), 1064–1095. https://doi.org/10.1177/07356331221133560

Gao, X., Yang, Y., Du, Y., & Sun, D. (2025). Effect of mind mapping-based scaffolding on elementary students' computational thinking in block-based programming. *Journal of Educational Computing Research*, *63*(1), 236–271. https://doi.org/10.1177/07356331241303067

Izzati, A. A. T., Sa'dijah, C., Arifin, S., Dewi, R. S. I., & Ekawati, R. (2026). Integrating augmented reality in mathematics: A mixed-methods systematic review of spatial and geometric learning. *Jurnal Inovasi Teknologi Pendidikan*, *13*(2), 143–159. https://doi.org/10.21831/jitp.v13i2.85197

Jang, J., Jeon, J., & Jung, S. K. (2026a). Immersive Technologies (AR/VR) for Computer Science and Computational Thinking Education: A Systematic Literature Review of Trends and Gaps. *IEEE Access*, *14*, 54414–54441. https://doi.org/10.1109/access.2026.3680395

Jegstad, K. M., Heggernes, S. L., Jøsok, E., Ryen, E., Svanes, I. K., & Tørnby, H. (2025). Approaches to critical thinking in primary education classrooms: A systematic review. *Educational Research Review*, *48*, 100711. https://doi.org/10.1016/j.edurev.2025.100711

Kalemkuş, J., & Kalemkuş, F. (2023). Effect of the use of augmented reality applications on academic achievement of student in science education: Meta analysis review. *Interactive Learning Environments*, *31*(9), 6017–6034. https://doi.org/10.1080/10494820.2022.2027458

Kania, N., Kusumah, Y. S., Dahlan, J. A., Nurlaelah, E., Gürbüz, F., & Bonyah, E. (2024). Constructing and providing content validity evidence through the Aiken's V index based on the experts' judgments of

the instrument to measure mathematical problem-solving skills. *REID (Research and Evaluation in Education)*, *10*(1), 64–79. https://doi.org/10.21831/reid.v10i1.71032

Kao, G. Y.-M., & Ruan, C.-A. (2022). Designing and evaluating a high interactive augmented reality system for programming learning. *Computers in Human Behavior*, *132*, 107245. https://doi.org/10.1016/j.chb.2022.107245

Latifah, A. Y. N., Wahyudin, D., & Mulyasari, E. (2026). Planetaria: Augmented reality-based learning media on solar system lessons for elementary school students. *Jurnal Inovasi Dan Teknologi Pembelajaran*, *13*(1), 24–35. https://doi.org/10.17977/um031v13i12026p024

Lee, S. J., Francom, G. M., & Nuatomue, J. (2022). Computer science education and K-12 students' computational thinking: A systematic review. *International Journal of Educational Research*, *114*, 102008. https://doi.org/10.1016/j.ijer.2022.102008

Leon Diaz, F. Y., Boude Figueredo, O. R., & Vargas Sanchez, A. D. (2026). Pedagogical mediation with ICT for the development of critical thinking in primary education: A systematic review. *Thinking Skills and Creativity*, *60*, 102085. https://doi.org/10.1016/j.tsc.2025.102085

Lin, Y., & Yu, Z. (2023). A meta-analysis of the effects of augmented reality technologies in interactive learning environments (2012–2022). *Computer Applications in Engineering Education*, *31*(4), 1111–1131. https://doi.org/10.1002/cae.22628

Mills, K. A., Cope, J., Scholes, L., & Rowe, L. (2025). Coding and computational thinking across the curriculum: A review of educational outcomes. *Review of Educational Research*, *95*(3), 581–618. https://doi.org/10.3102/00346543241241327

Mustika, D., Kustandi, D., Farida, I. A., & Ihsan, I. R. (2026). Experiential psychological mechanisms in immersive learning: Presence, embodiment, and engagement as mediator in learning outcomes. *Jurnal Inovasi Dan Teknologi Pembelajaran*, *13*(1), 48–60. https://doi.org/10.17977/um031v13i12026p048

Nordby, S. K., Bjerke, A. H., & Mifsud, L. (2022). Computational thinking in the primary mathematics classroom: A systematic review. *Digital Experiences in Mathematics Education*, *8*, 27–49. https://doi.org/10.1007/s40751-022-00102-5

Purwono, R., Nuswowati, M., Subali, B., & Ellianawati. (2026). Coding-based instruction for developing computational thinking in primary education: A systematic review. *Jurnal Ilmiah Pendidikan Dasar*, *13*(1), 66–84. https://doi.org/10.30659/pendas.13.1.66-84

Rusnilawati, Rizki, A. F., Rahman, F., Hazima, A. A., Utomo, A. C., & Sulaimon, J. T. (2026). Augmented inquiry: Developing AR-based media for spatial geometry in elementary schools. *Jurnal Ilmiah Pendidikan Dasar*, *13*(1), 147–174. https://doi.org/10.30659/pendas.13.1.147-174

Sakr, A., & Abdullah, T. (2024). Virtual, augmented reality and learning analytics impact on learners, and educators: A systematic review. *Education and Information Technologies*, *29*(15), 19913–19962. https://doi.org/10.1007/s10639-024-12602-5

Santaengracia, J. J., Palop, B., & Rodríguez-Muñiz, L. J. (2026). Computational thinking in K-6 education: A scoping review of instructional practices. *Journal of Computers in Education*, *13*, 983–1016. https://doi.org/10.1007/s40692-025-00380-z

Su, J., & Yang, W. (2023). A systematic review of integrating computational thinking in early childhood education. *Computers & Education Open*, *4*, 100122. https://doi.org/10.1016/j.caeo.2023.100122

Tawfik, A. A., Payne, L., & Olney, A. M. (2024). Scaffolding computational thinking through block coding: A learner experience design study. *Technology, Knowledge and Learning*, *29*(1), 21–43. https://doi.org/10.1007/s10758-022-09636-4

Usfinit, I. F., Kuswandi, D., & Soepriyanto, Y. (2025). Development of augmented reality feature modules on the introduction of historical monuments in Timor Leste. *Jurnal Inovasi Dan Teknologi Pembelajaran*, *11*(1), 11–23. https://doi.org/10.17977/um031v11i12024p011

Yang, T.-C., & Lin, Z.-S. (2024). Enhancing elementary school students' computational thinking and programming learning with graphic organizers. *Computers & Education*, *209*, 104962. https://doi.org/10.1016/j.compedu.2023.104962

Zhu, X., Peng, K., Yu, S., & Wang, G. (2026). Can augmented reality technology reduce learners' cognitive load? A meta-analysis. *Smart Learning Environments*, *13*, 9. https://doi.org/10.1186/s40561-025-00429-7