Online-Journals.org

iJIM International Journal of Interactive Mobile Technologies

iJIM | eISSN: 1865-7923 | Vol. 20 No. 6 (2026) | OPEN ACCESS

https://doi.org/10.3991/ijim.v20i06.58489

PAPER

Enhancing Algorithmic Design through Mobile-Supported Constructivist Learning: A Problem-Based Approach

Rafael Ricardo Mantilla Guiza (✉)

Universidad de Investigación y Desarrollo, Bucaramanga, Colombia

rmantilla1@udi.edu.co

ABSTRACT

This study examines a mobile-supported constructivist learning environment (CLE) grounded in problem-based learning (PBL) to foster computational thinking (CT) and algorithm design skills among first-year engineering students. A quasi-experimental pretest and posttest design (n = 62) was implemented using a custom web and mobile platform that enabled ubiquitous access, immediate automated feedback, and iterative practice on structured algorithmic challenges aligned with CT dimensions related to knowing, doing, and being. This instructional design builds upon established CLE and PBL frameworks reported in prior studies, and is consistent with recent evidence on the role of mobile and intelligent technologies in supporting CT development in higher education contexts. Paired sample analyses showed statistically significant improvements across CT components, with t values ranging between 8.22 and 22.92 and Cohen's d values exceeding 1.0, indicating very large effect sizes. These findings demonstrate a substantial increase in student proficiency from the baseline. Qualitative evidence derived from rubric-based reflections highlighted increased autonomy, collaboration, and learner motivation facilitated by mobile access and interactive problem solving. The contribution of this work lies in three aspects: a hybrid CLE and PBL instructional model operationalized through mobile-supported practice, a competency framework integrating knowing, doing, and being dimensions of learning, and empirical evidence of learning gains in algorithmic problem solving. The findings support mobile and interactive technologies as effective catalysts for flexible and learner-centered instruction in programming education.

KEYWORDS

mobile learning, interactive learning environments, computational thinking (CT), algorithmic problem solving, educational technology

Mantilla Guiza, R. R. (2026). Enhancing Algorithmic Design through Mobile-Supported Constructivist Learning: A Problem-Based Approach. *International Journal of Interactive Mobile Technologies (iJIM)*, 20(6), pp. 125–141. https://doi.org/10.3991/ijim.v20i06.58489

Article submitted 2025-09-02. Revision uploaded 2026-02-02. Final acceptance 2026-02-02.

1 INTRODUCTION

Teaching algorithm design to first-year engineering students remains a complex challenge, as many struggle to bridge the gap between abstract logic and practical execution. In this scenario, computational thinking (CT) has emerged as a fundamental bridge, encompassing cognitive processes such as decomposition, pattern recognition, abstraction, and algorithm design [1–6]. These skills are not merely technical requirements but are essential for solving the complex problems of current times, which is why they have been increasingly embedded into modern educational policies and international frameworks [2], [5].

To move beyond broad definitions and truly capture student progress, it is necessary to interpret CT through measurable indicators. This perspective is supported by recent frameworks that highlight the critical interplay between systematic logic and cognitive evolution as measurable indicators of learning [7], [8]. Specifically, these dimensions are defined as follows:

- Decomposition: Breaking down data, processes, or problems into smaller, manageable parts
- Pattern recognition: Observing trends and regularities within data
- Abstraction: Identifying underlying principles that generate these patterns
- Algorithm design: Creating step-by-step instructions to solve problems

It is important to recognize that CT is not a static concept; it is an evolving pedagogical strategy that adapts depending on the context and the researcher's intent [6]. In the scope of this work, CT is understood as a dynamic set of abilities that empower learners to approach problems systematically, integrating the dimensions mentioned above to reach effective and creative solutions [5], [6], [9].

The recent global shift toward digital education has accelerated this evolution. While prior initiatives explored both connected and unplugged approaches to CT development [7], [10], the post-pandemic landscape has emphasized the need for flexible, student-centered models that promote genuine autonomy and critical thinking [11]. In this context, mobile-supported learning environments have proven to be powerful tools for fostering CT, especially when they are grounded in constructivist and problem-based learning (PBL) strategies [2], [6], [12].

This study introduces a hybrid constructivist learning environment (CLE) and PBL instructional model designed to facilitate ubiquitous learning and iterative practice beyond the traditional classroom. The environment is structured around three core dimensions (knowing, doing, and being) and is implemented through algorithmic challenges that challenge the student's logic in real-time. By evaluating this approach, the objective is to demonstrate how interactive, real-world scenarios can significantly enhance CT and algorithm design skills in the early stages of engineering education [4].

2 RELATED WORK

The integration of CT into educational systems has gained significant momentum, particularly in the context of mobile learning and constructivist pedagogies. CT is widely recognized as a core competency for the 21st century, essential for problem-solving, algorithmic reasoning, and digital fluency [5], [6], [9]. Global frameworks and educational policies have also emphasized CT as a key component of digital literacy and lifelong learning [10], [11]. Recent conceptual models highlight

CT as a multidimensional construct involving conceptual, practical, and reflective dimensions, which aligns with competency-based education [2], [7], [8].

2.1 Mobile learning and CLE and PBL integration

Mobile-supported learning environments have emerged as effective tools for fostering CT, especially when combined with constructivist and PBL strategies [2], [6], [12]. These environments enable ubiquitous access, personalized learning, and interactive engagement, allowing students to practice algorithmic problem-solving beyond classroom boundaries. Mobile technologies enhance learner autonomy, motivation, and collaboration, while providing immediate feedback factors critical for CT development [4], [6], [13]. Furthermore, mobile learning supports flexible and student-centered approaches, which became essential during the COVID-19 pandemic [11].

2.2 Didactic strategy based on PBL within a mobile constructivist learning environment

Problem-based learning was adopted as the core instructional strategy to operationalize the CLE in a mobile context, emphasizing active learning, collaboration, and iterative refinement [3], [14], [15], [16]. In this approach, students act as primary agents, while instructors serve as facilitators by guiding discussions and providing support rather than delivering traditional lectures. This aligns with the goal of PBL to develop the teaching and learning process through activities that require the construction of knowledge and the development of skills [13, p. 125].

The design of algorithmic problems followed a structured metadata format including title, scenario description, input and output specifications, and examples, similar to competitive programming platforms such as Top Coder, Coder byte, Project Euler, and ICPC [17]–[22]. In the custom web and mobile platform developed for this study, this format was optimized for small screens and integrated with immediate automated feedback, enabling students to validate solutions, debug errors, and iteratively refine their code ubiquitously. This design promoted key CT practices such as decomposition, abstraction, and debugging [5], [6], [7], [9].

The PBL cycle was implemented through five sequential stages:

- Problem framing on the platform using contextualized scenarios
- Team analysis and discussion to identify knowledge gaps
- Individual coding and peer review supported by mobile access to enhance flexibility
- Iterative testing and debugging with real-time feedback provided by the system
- Reflection and rubric-based appraisal, reinforcing the knowing, doing, and being dimensions of learning [2], [8]

Mobile learning technologies enhanced this instructional strategy by enabling ubiquitous access, personalized learning, and interactive engagement, acting as catalysts for educational transformation and supporting lifelong learning [2], [6]. This integration ensured that students could engage with algorithmic challenges anytime and anywhere, strengthening learner autonomy and motivation while maintaining alignment with competency-based education principles.

2.3 Empirical evidence and trends

Prior to 2020, various initiatives explored both connected and unplugged approaches to CT development [7], [10]. The pandemic accelerated the adoption of digital and mobile learning environments, reinforcing the need for scalable models that integrate technology and active methodologies [11]. Bibliometric analyses confirm a growing trend in publications on CT, mobile learning, and interactive environments, with recent PRISMA-guided analyses highlighting the specific surge of mobile learning applications within higher education settings [23], supported by over 2,800 documents indexed between 1987 and 2025 [4]. This reflects increasing academic interest in strategies that combine algorithmic thinking with digital fluency.

Several empirical studies have examined CT development in diverse contexts, emphasizing the effectiveness of constructivist approaches and technology integration. Table 1 summarizes representative studies that informed this study design:

Table 1. Similar studies (based on [5]–[7], [9], [12], [13], [24])

Ref.	Population	Meth.	CT Components
[5]	Africa, higher education students	Qual.	Decomposition, pattern recognition, abstraction, algorithms
[6]	Thailand, higher education, 3rd-year students	Quant.	Decomposition, pattern recognition, abstraction, algorithms
[7]	Spain, higher education, programming course	Quant.	Decomposition, pattern recognition, abstraction, algorithms
[9]	Africa, secondary school students	Quant.	Abstraction through discovery, extraction, creation, and assembly
[12]	Korea, higher education, liberal arts course	Mixed	Divergent, critical, and logical thinking
[24]	Korea, secondary school students	Quant.	Problem analysis, data collection, abstraction, algorithm, coding, testing
[13]	Turkey, secondary school students	Mixed	Creativity, algorithmic thinking, collaboration, critical thinking

These studies consistently report that constructivist approaches combined with digital or mobile technologies improve CT skills and learner engagement. However, most interventions lack a structured competency framework and do not fully exploit mobile access for iterative practice and feedback, gaps that are addressed in this study.

2.4 Connection to the current study

This review underscores the need for hybrid models that combine CLE and PBL strategies with mobile-supported practice to foster CT in authentic contexts. The proposed approach addresses this gap by implementing a custom web and mobile platform grounded in CLE and PBL principles, structured around the dimensions of knowing, doing, and being, and operationalized through algorithmic challenges aligned with real-world scenarios [4].

3 MATERIALS AND METHODS

This study adopted a quasi-experimental mixed-methods design to examine the effects of a mobile-supported CLE, grounded in PBL, on the development of CT and algorithmic problem-solving skills among first-year university students.

The methodological design was structured to ensure alignment between the pedagogical model, the technological environment, and the assessment instruments,

allowing for a robust evaluation of learning outcomes across the dimensions of knowing, doing, and being.

3.1 Research design

A mixed methods approach was employed, combining quantitative and qualitative techniques to capture both measurable learning gains and process-oriented evidence of student development. Quantitatively, a pretest and posttest design was used to assess changes in algorithmic performance and CT-related competencies. Qualitatively, rubric-based evaluations and reflective artifacts were analyzed to interpret how students engaged with problem-solving processes throughout the intervention [17], [25].

The study followed a quasi-experimental design without a control group, which is appropriate for authentic educational contexts where random assignment is not feasible [15]. The intervention was embedded within a regular academic course, ensuring ecological validity.

The instructional strategy was implemented within a CLE framework [24], operationalized through PBL cycles [26], [27]. Learning activities were structured as contextualized algorithmic problems designed to promote active knowledge construction, iterative testing, and reflective practice. The development of CT and computational skills was assessed using established frameworks and indicators [6], [28].

To evaluate the impact of the intervention, paired sample t-tests were applied to compare pre- and post-intervention results. The assumptions of paired observations and approximate normality were verified prior to analysis, supporting the use of parametric statistical procedures.

3.2 Participants

The sample consisted of 62 first-year engineering students enrolled in the Fundamentals of Programming course during the second academic semester of 2023. Participants were selected using a non-probabilistic convenience sampling method, ensuring that all students had access to the mobile-supported platform as part of their regular laboratory sessions. Participation was voluntary, and the study protocol was approved by the institutional Ethics Committee and Research Coordination Office. All participants signed informed consent forms outlining the objectives and procedures of the study, as well as their right to withdraw at any time. All collected data were anonymized prior to analysis.

3.3 Materials and instruments

The design and implementation of the intervention were guided by a structured conceptual framework for CT, which informed both the learning environment and the assessment instruments. This framework was primarily based on Brennan's model [2], which conceptualizes CT as an integration of concepts, practices, and perspectives. In this study, the framework was operationalized through three complementary learning dimensions: knowing, doing, and being.

The knowing dimension refers to students' conceptual understanding of programming structures and computational principles. Core programming concepts

were organized into progressive instructional packages, including sequences, data handling, variables, operators, conditionals, loops, procedures, modularity, and debugging. This conceptual organization ensured a gradual increase in cognitive demand and supported systematic knowledge construction. An overview of the conceptual structures aligned with this dimension is presented in Table 2.

Table 2. Provides a detailed overview of the CLE's conceptual structures as defined by this framework (based on [2])

Package	Topics	Conceptual Description (Knowing)
Package 1	Sequences	A series of steps from individual instructions.
	Data	Ability to store, retrieve, and update values.
	Variables	Memory space to store a value.
	Operators	Support for mathematical, logical, and string expressions.
	Debugging	Identifying review points.
Package 2	Conditionals	Executing different sequences based on decisions.
Package 3	Loops	Mechanisms to execute the same sequence multiple times.
	Events	Actions involving other sequences.
	Procedure	Functions, or methods, are groupings of instructions for specific tasks.
	Parallelism	Sequences of instructions occurring simultaneously.

The doing dimension focuses on the practical application of CT through the design and implementation of algorithms. Learning activities emphasized abstraction, decomposition, generalization, algorithmic thinking, and debugging, which were embedded in iterative problem-solving processes. These practices guided students in constructing, testing, refining, and reusing algorithmic solutions. The operationalization of CT practices and their associated skills is summarized in Table 3.

Table 3. Provides a detailed overview of the CLE's practical structures according to this framework (based on [2], [29])

Practices (Knowing How)	Description	Required and Developing Skills
Incremental and Iterative	Imagining and building through constant progress.	Abstraction, Decomposition
Testing and Debugging	Trial and error, seeking experience to anticipate.	Generalization, Algorithm, Debugging
Reuse and Mixing	Consulting and reusing old code in a new adaptation.	Generalization, Algorithm, Debugging
Abstraction and Modularity	Building something large from small, functional modules.	Abstraction, Decomposition

The being dimension addresses reflective, contextual, and collaborative aspects of computational thinking. This perspective emphasizes learners' ability to express ideas through computing, connect with peers in collaborative problem-solving contexts, and critically engage with technological challenges. These elements supported the development of learner agency, motivation, and ethical awareness in digital environments. The perspective-based structures adopted in this study are summarized in Table 4.

Table 4. Provides a detailed overview of the CLE's perspective-based structures within this framework (based on [2])

Perspectives (Knowing Being)	Description
Express	Not just consuming technology, but using computing to create, use, design, and express oneself.
Connect	The added value of creating with others, communities, and teams gives extra value to the outcome.
Ask	Empowerment to accept technological changes and be a part of them.

The instructional content was delivered through a set of fifteen contextualized algorithmic problems distributed across three instructional packages, each containing five problems with increasing levels of complexity. Problems followed a standardized structure including scenario description, input and output specifications, and illustrative examples, inspired by competitive programming platforms [17]–[22]. This design facilitated clarity, comparability, and automated evaluation. The distribution of problems across instructional packages is presented in Table 5.

Table 5. List of problems distributed by packages

#	Package	Title of the Problem
1	1	Distance Traveled by a Body
2	1	The Loan
3	1	The Height of the Ladder
4	1	The Parts of a Division
5	1	Final Grades in Technology and Computing
6	2	Overtime Hours
7	2	My Soulmate
8	2	Averages with Range
9	2	Leap Year
10	2	Days of the Week
11	3	LCM
12	3	Greater and Lesser
13	3	Are you a cousin?
14	3	Binary House Power Calculation
15	3	Palindrome Phrases

All algorithmic problems were implemented within a custom web-based application with full mobile support. The platform enabled students to submit solutions, receive immediate automated feedback, and iteratively refine their code. This mobile-supported interaction allowed learners to engage with problem-solving activities beyond scheduled class time, reinforcing ubiquitous learning and sustained practice.

To evaluate both the learning environment and student outputs, a set of software quality and instructional design metrics was applied. These metrics addressed alignment with learning objectives, adaptation and feedback mechanisms, usability, accessibility, reusability, and compliance with international standards. The quality metrics used in this study are summarized in Table 6.

Table 6. The quality metrics for software development (based on [28])

Measured Factors	Description
Learning Objectives	Alignment of content with learning objectives, activities, assessments, and student characteristics.
Adaptation and Feedback	Content adapts to student needs, with feedback driven by differential participation.
Design and Presentation	Visual and auditory design enhances learning and promotes efficient mental processing.
Interaction Usability	The learning object allows for easy navigation, a predictable user interface, and quality help features.
Accessibility	Controls and presentation formats are designed for better accessibility for students.
Reusability	The learning object is designed for use in different learning contexts and for students at various learning levels.
Standards Compliance	Adherence to international standards such as SCORM, IEEE LOM, W3C HTML, etc.

3.4 Mobile-supported learning environment

The instructional intervention was operationalized through a custom-built web-based platform featuring a fully responsive design. While the environment is accessed via a mobile browser rather than a native application, it was specifically engineered to support mobile-learning by leveraging the portability and ubiquity of smartphones, embracing digital innovation to facilitate transformative learning experiences across different contexts [30]. This approach ensures "just-in-time" practice and learning in diverse physical environments.

The platform's design aligns with mobile learning affordances by providing:

- Ubiquitous Access: Students can engage with algorithmic challenges regardless of their location, promoting continuous interaction with the content.
- Iterative Practice: The interface is optimized for mobile interaction, allowing for short, focused sessions of problem-solving that fit into the students' daily routines.
- Immediate Automated Feedback: Essential for mobile contexts where a human tutor is not present, the system provides real-time validation of code and logic.

The environment is structured into three instructional packages (P1, P2, and P3) that guide the learner through increasing levels of complexity in algorithm design. This responsive architecture ensures that the CLE is not confined to a desktop laboratory but is integrated into the learner's mobile ecosystem.

3.5 Use of artificial intelligence tools

Generative artificial intelligence tools were utilized during the preparation of this manuscript exclusively for language refinement, grammar correction, and stylistic polishing. Tools such as ChatGPT and Gemini were employed to ensure linguistic precision and academic clarity. The author explicitly states that these tools were not used to draft, compose, or generate the scientific narrative, the interpretation of data, or the conclusions of the study. The entire intellectual content and the development

of the instructional framework were authored solely by the researcher, who takes full responsibility for the originality and accuracy of the work presented.

3.6 Procedure

The methodology followed a structured sequence consisting of three main phases: pretest, intervention, and posttest.

At the beginning of the study, students completed a diagnostic assessment designed to establish a baseline of their CT and algorithmic problem-solving skills. This pretest consisted of fifteen algorithmic problems distributed across the three instructional packages and was administered digitally through the custom web application in a supervised laboratory setting.

Following the diagnostic phase, the instructional intervention was conducted over one academic semester. Students engaged in CLE activities three times per week, with each session lasting approximately ninety minutes, as part of the Fundamentals of Programming course. The overall workflow of the intervention and student interaction with the platform is illustrated in Figure 1.

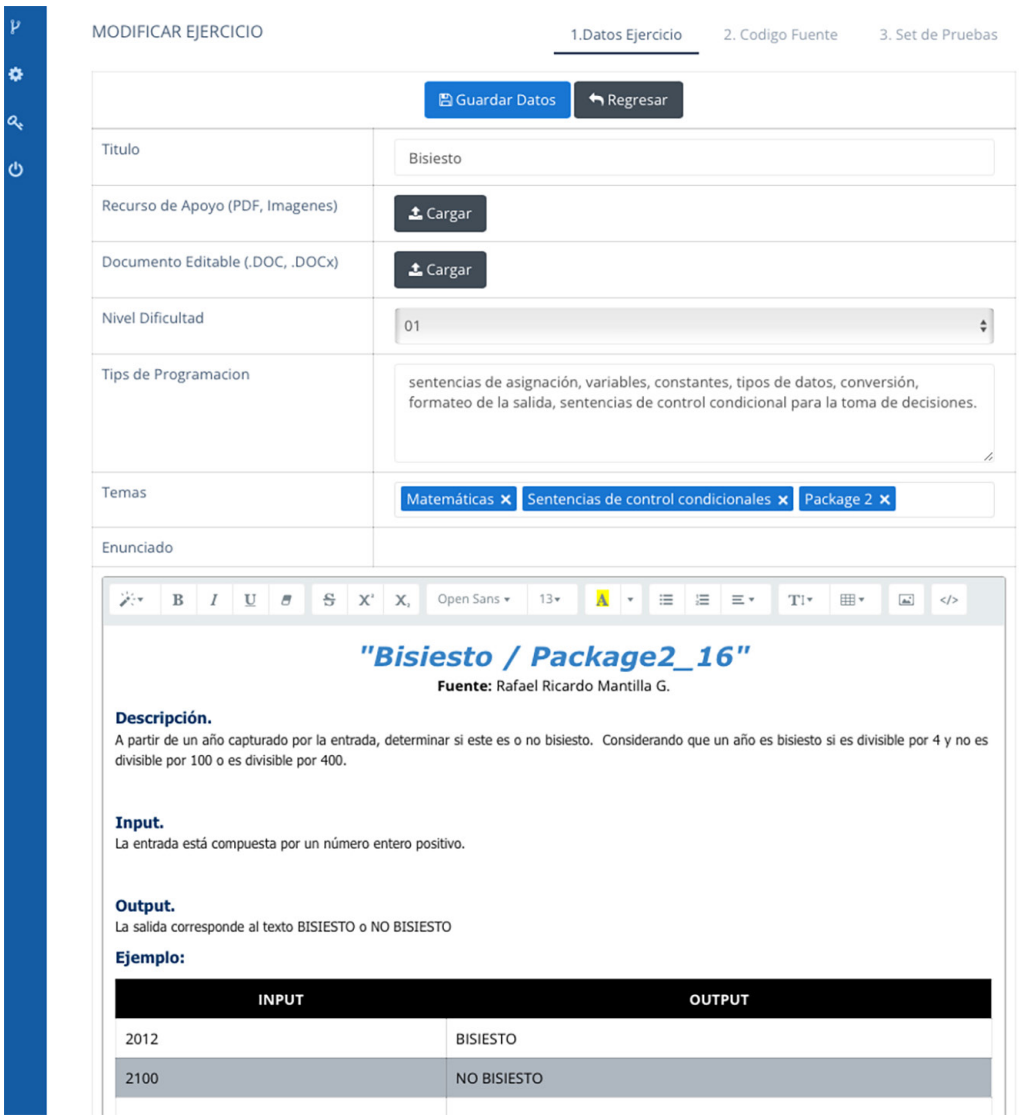

Fig. 1. Problem within the implemented EAC

During the intervention, students worked both individually and collaboratively on the three packages of algorithmic challenges, which progressively increased in complexity and required the application of multiple CT components. The processes of algorithm design, testing, and debugging supported by the platform are shown in Figure 2. Instructors acted primarily as facilitators, guiding discussions and providing targeted feedback rather than delivering traditional lectures.

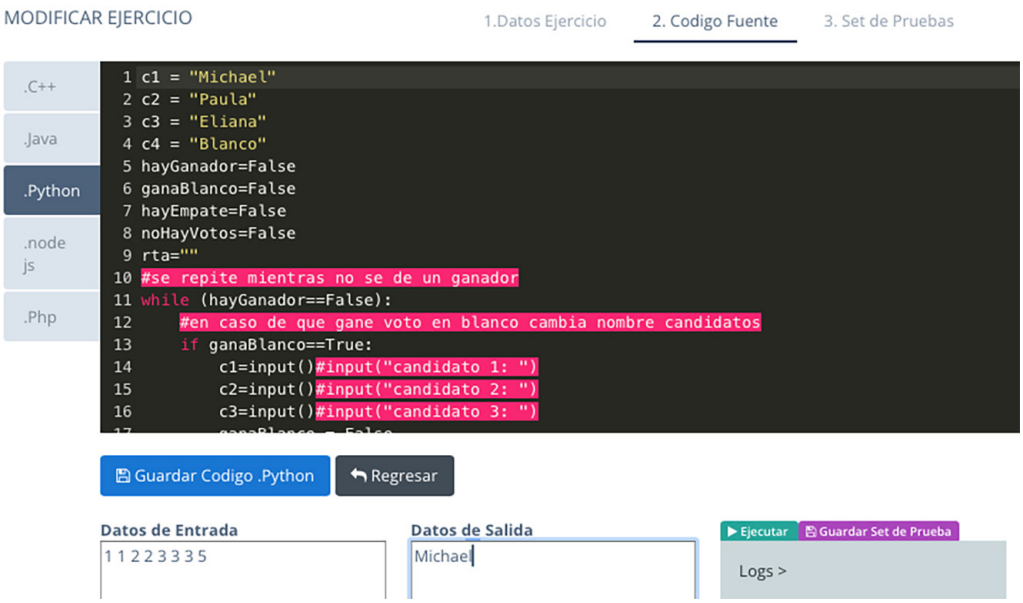

Fig. 2. Design, debugging, and evaluation tools

At the conclusion of the intervention period, the same diagnostic instrument used in the pretest was administered under identical conditions as a posttest. Traceability and performance monitoring resources used to support evaluation are illustrated in Figure 3.

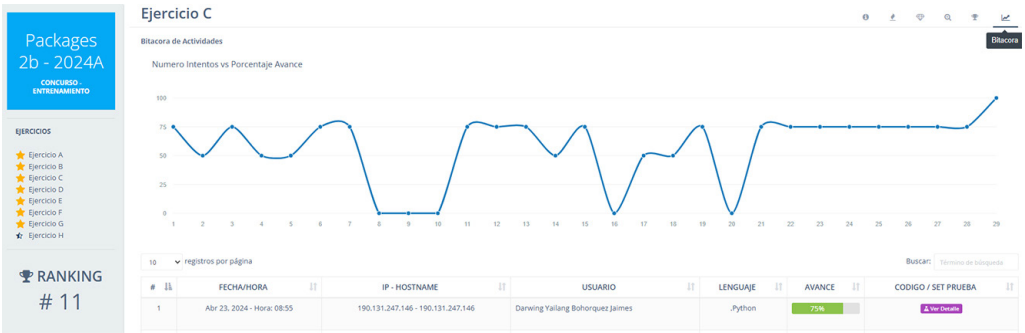

Fig. 3. Traceability resources regarding student performance

3.7 Data analysis

Quantitative data were analyzed using IBM SPSS Statistics v25. Prior to inferential analysis, descriptive statistics, including means and standard deviations, were calculated for all pretest and posttest measures to provide an initial overview of student performance.

To examine the impact of the intervention, paired-sample t-tests were conducted to compare pretest and posttest scores across the dimensions of knowing, doing, and being, as well as across the three instructional problem packages. The paired t-test was selected because the same participants were assessed under two conditions, and the data met the assumptions of paired observations and approximate normality.

Effect sizes were calculated using Cohen's *d* to assess the magnitude and practical significance of observed differences. When correlation-based statistics were obtained, Pearson's *r* values were converted into Cohen's *d* using the standard transformation:

$$d = \frac{2r}{\sqrt{1 - r^2}}$$

This conversion enabled a consistent interpretation of effect magnitude across paired comparisons and facilitated comparison with related studies in CT and programming education. The combination of statistical significance testing and effect size estimation provided a more comprehensive interpretation of the results.

Qualitative data derived from rubric-based evaluations and student reflections were analyzed thematically. This analysis focused on identifying recurring patterns related to CT development, learner autonomy, collaborative problem solving, and engagement within the mobile-supported learning environment.

3.8 Hypotheses

H0: There is no statistically significant difference in students' CT and algorithmic problem-solving skills between the pretest and posttest assessments after participating in a mobile-supported CLE grounded in CT and problem-based learning.

H1: There is a statistically significant difference in students' CT and algorithmic problem-solving skills between the pretest and posttest assessments after participating in a mobile-supported CLE grounded in CT and problem-based learning.

3.9 Research question

How can a mobile-supported PBL model implemented within a CLE foster the development of CT and algorithmic problem-solving skills in first-year university students?

4 RESULTS

This section presents the results obtained from the implementation of the mobile-supported CLE grounded in CT and PBL. The findings are reported using quantitative and qualitative evidence to describe changes in students' CT and algorithmic problem-solving skills following the instructional intervention.

4.1 Quantitative results

The analysis of student performance began with a comparison of descriptive statistics to establish the baseline and the magnitude of improvement. Prior to the intervention, students exhibited a low proficiency level across all dimensions, with mean scores ranging from M = 0.74 to M = 1.50. Following the implementation of the mobile-supported CLE, post-test measures showed a substantial increase, with averages rising to between M = 3.44 and M = 4.19.

While the paired samples correlation analysis showed generally low and non-significant relationships (r between – 0.22 and 0.23; $p > 0.05$), this lack of correlation reflects the drastic shift in student performance levels rather than a lack of consistency. In educational interventions with high impact, it is common for the post-test results to decouple from the pre-test baseline as students acquire new competencies.

The paired sample t-test results confirmed that these improvements were statistically significant across all variables, with t values ranging from 8.22 to 22.92 ($p < 0.001$). These results strongly support the rejection of the null hypothesis. To assess the practical significance of these gains, effect sizes were calculated using Cohen's d. The analysis revealed exceptionally large effects, with d values exceeding 1.5 in most comparisons. Notably, the most substantial improvement was observed in the "Doing" dimension (t = –18.08), where the mean score tripled from 1.16 to 4.10.

The convergence of these significant t-values and massive effect sizes provides robust evidence that the instructional intervention effectively fostered CT skills. A detailed summary of these statistics is provided in Table 7.

Table 7. Statistical results and effect sizes

Pair	Dimension/Variable	N	Pre-Test M (SD)	Post-Test M (SD)	Sig. (p)	t-Value	Cohen's *d*
1	**Doing (Overall)**	62	1.16 (0.41)	4.10 (1.18)	< .001	–18.08	2.30
2	**Knowing (Overall)**	62	1.26 (0.51)	3.63 (0.73)	< .001	–22.92	2.91
3	**Being (Overall)**	62	1.00 (1.09)	3.52 (0.94)	< .001	–12.92	1.64
4	P1: Doing	62	1.48 (0.82)	3.95 (1.12)	< .001	–14.13	1.79
5	P1: Knowing	62	1.45 (0.76)	3.60 (0.78)	< .001	–17.68	2.25
6	P1: Being	62	1.45 (1.28)	3.53 (1.28)	< .01	–8.22	1.04
7	P2: Doing	62	1.16 (0.49)	3.98 (1.15)	< .001	–17.10	2.17
8	P2: Knowing	62	1.50 (0.74)	3.89 (0.77)	< .001	–19.85	2.52
9	P2: Being	62	0.90 (1.17)	3.48 (0.99)	< .001	–12.35	1.57
10	P3: Doing	62	1.00 (0.00)	4.19 (1.35)	< .001	–18.58	2.36
11	P3: Knowing	62	1.00 (0.00)	3.55 (1.00)	< .001	–20.01	2.54
12	P3: Being	62	0.74 (1.01)	3.44 (0.95)	< .001	–15.17	1.93

4.2 Performance across learning dimensions

An additional analysis of the results focused on student performance across the learning dimensions of knowing, doing, and being, based on the statistical outcomes

reported in the previous section. Improvements were observed between pretest and posttest measures in all three dimensions.

In the knowing dimension, results indicated gains in students' conceptual understanding of programming fundamentals, including algorithmic structures and control flow. Higher post-test scores reflected improved ability to identify appropriate solution strategies and select relevant programming constructs.

In the doing dimension, students demonstrated enhanced performance in the execution of algorithmic solutions, code implementation, and debugging processes. Improvements were evident in tasks requiring the translation of problem specifications into functional code and the iterative refinement of solutions.

In the being dimension, the results reflected positive changes in students' metacognitive engagement, including greater persistence, confidence, and self-regulation during problem-solving activities. Students exhibited more systematic approaches to addressing complex challenges and increased awareness of their learning processes.

4.3 Qualitative insights

Rubric-based evaluations and student reflections indicated that learners strengthened algorithmic thinking, applied abstraction and decomposition strategies, engaged in collaborative learning processes, and developed increased autonomy and motivation. These outcomes were observed consistently throughout the instructional intervention and reflect changes in both problem-solving behaviors and learning engagement.

Systematic observation of student performance further revealed improvements in the organization of algorithmic solutions and the use of logical reasoning when addressing programming tasks. Over time, students demonstrated more structured approaches to problem solving, including clearer decomposition of tasks, more deliberate selection of programming constructs, and increased use of iterative testing and debugging strategies.

Students also exhibited greater persistence when facing complex challenges and relied less on direct instructor intervention, opting instead for peer collaboration and independent exploration of alternative solution strategies. These qualitative outcomes align with the characteristics of constructivist, mobile-supported learning environments reported in related studies [5], [6], [7].

5 DISCUSSION

The results indicate a substantial improvement in student performance, particularly within the practical dimensions of algorithm design. The statistically significant increase in the 'Doing' dimension, where mean scores rose approximately threefold, reflects an enhanced capacity for procedural application. This improvement suggests that the immediate, automated feedback provided by the platform functioned as a scaffolding mechanism, allowing students to address syntax and logic errors in real time. By reducing the latency between error generation and correction, the system supported iterative practice, enabling students to refine their solutions asynchronously rather than relying exclusively on scheduled laboratory interventions [29].

The ubiquity of the mobile interface played a critical role in these findings. Unlike desktop-based coding, the mobile app enabled 'micro-learning' sessions during fragmented time intervals (e.g., commuting). This frequency of interaction facilitated

spaced repetition of algorithmic syntax, directly contributing to the high proficiency scores observed in the post-test. By lowering the friction of entry into coding tasks, the mobile environment transformed passive downtime into active problem-solving opportunities.

While recent systematic reviews on educational technology [2], [4] focus primarily on the integration of Artificial Intelligence and Computational Thinking, our findings extend these discussions into the realm of mobile learning. Specifically, Weng et al. [2] emphasize the need for automated scaffolding to reduce cognitive load in programming education, a requirement we addressed through the mobile app's instant feedback. Similarly, Husin et al. [16] highlight the effectiveness of PBL in engineering; our study demonstrates that mobile ubiquity acts as a catalyst for this PBL approach, allowing continuous engagement beyond the physical classroom.

The statistically significant improvements observed across the dimensions of knowing, doing, and being suggest that the CLE–PBL model supported balanced development of conceptual understanding, procedural fluency, and reflective engagement. These findings are consistent with prior research highlighting the multidimensional nature of CT and the importance of active, student-centered pedagogical approaches in programming education [2], [5], [6].

The use of a custom web-based platform accessible through mobile devices emerged as a key enabling factor in supporting flexible and self-directed learning. As reflected in the results, students engaged with algorithmic challenges both inside and outside the classroom, promoting autonomy and sustained practice. These conditions have been widely recognized as critical for the development of CT skills and align with existing studies on mobile--supported learning environments [4], [7]. Quantitative gains were complemented by qualitative evidence indicating increased motivation, collaboration, and engagement, in line with prior findings on the role of mobile technologies in accommodating diverse learning styles [6], [13].

An additional contribution of this study lies in the adoption of a competency-based evaluation framework structured around knowing, doing, and being. The rubric-based assessment approach enabled a more nuanced interpretation of learning outcomes by capturing not only technical performance but also students' reflective and metacognitive processes. This aspect is particularly relevant in programming education, where traditional assessments often emphasize correctness while overlooking higher-order cognitive and reflective dimensions.

Overall, the findings support the potential of interactive and technology-enhanced pedagogical models to transform introductory programming education. By aligning instructional design with CT frameworks and leveraging mobile technologies, educators may create more inclusive, scalable, and sustainable learning environments. Nevertheless, the results should be interpreted within the scope of the study. Future research is required to examine the scalability of the proposed model across larger cohorts, different institutional settings, and longitudinal implementations.

6 CONCLUSION

This study suggests that integrating programmed instruction within a mobile-supported CLE grounded in PBL can effectively support the development of CT and algorithmic problem-solving skills in higher education contexts. The results indicate improvements across conceptual, procedural, and reflective dimensions of learning, as evidenced by both quantitative and qualitative findings.

The proposed CLE–PBL model combines pedagogical and technological elements to support flexible and self-directed learning. The use of a custom web-based platform accessible through mobile devices enabled students to engage with algorithmic challenges in varied contexts, while the competency-based evaluation framework facilitated the assessment of conceptual understanding, algorithm design, and reflective problem solving. Together, these components contribute to a scalable and adaptable instructional model for programming education.

By aligning instructional design with CT frameworks and leveraging mobile technologies, the model supports the integration of theoretical knowledge with practical application. These findings align with current trends in educational technology and highlight the potential of interactive, learner-centered environments in introductory programming courses.

Future research should further examine the long-term impact of this approach on student learning outcomes, its applicability across different institutional settings and STEM disciplines, and its potential to support the development of lifelong learning skills in digitally mediated educational contexts.

6.1 Future work

Future research on using mobile-supported CLEs to develop CT should focus on several key areas. First, it is crucial to examine the long-term impact, evaluating how students retain, transfer, and apply these skills in real-world, interdisciplinary, and professional contexts.

Additionally, comparative studies are needed to analyze the CLE-PBL model across different disciplines and educational levels to determine its scalability and applicability. To improve instruction, adaptive technologies such as artificial intelligence and learning analytics should be incorporated. This would allow for the personalization of algorithmic instruction based on each learner's characteristics and needs, helping to create more intelligent and effective learning ecosystems.

7 ACKNOWLEDGMENTS

This study was conducted with institutional approval and adhered to ethical standards for educational research. The author also acknowledges the contributions of the development team responsible for the design and maintenance of the web-based educational platform used in this study.

8 REFERENCES

[1] J. Arabit-García, P. A. García-Tudela, and M. P. Prendes-Espinosa, "Uso de tecnologías avanzadas para la educación científica," *Revista Iberoamericana de Educación*, vol. 87, no. 1, pp. 173–194, 2021. https://doi.org/10.35362/rie8714591

[2] X. Weng, H. Ye, Y. Dai, and O. Ng, "Integrating artificial intelligence and computational thinking in educational contexts: A systematic review," *Journal of Educational Computing Research*, vol. 62, no. 6, pp. 1420–1450, 2024. https://doi.org/10.1177/07356331241248686

[3] P. M. Bueno, Y. Victoria, and L. Fitzgerald, "Aprendizaje basado en problemas problem-based learning," *Theoria*, vol. 13, pp. 145–157, 2004.

[4] R. Tariq, M. A. Shibli, A. Saeed, M. Usman, and M. Asif, "Computational thinking in STEM education: Current state-of-the-art and future research directions," *Frontiers in Computer Science*, vol. 6, p. 1480404, 2025. https://doi.org/10.3389/fcomp.2024.1480404

[5] N. O. Ezeamuzie, J. S. C. Leung, R. C. C. Garcia, and F. S. T. Ting, "Discovering computational thinking in everyday problem solving: A multiple case study of route planning," *Journal of Computer Assisted Learning*, vol. 38, no. 6, pp. 1779–1796, 2022. https://doi.org/10.1111/jcal.12720

[6] S. Supaluk and J. Khlaisang, "Effects of a mobile cloud-based learning system using a P2P reverse engineering approach on enhancing computational thinking," *International Journal of Interactive Mobile Technologies*, vol. 15, no. 21, pp. 67–87, 2021. https://doi.org/10.3991/ijim.v15i21.23143

[7] C. Cachero, P. Barra, S. Melia, and O. Lopez, "Impact of programming exposure on the development of computational thinking capabilities: An empirical study," *IEEE Access*, vol. 8, pp. 72316–72325, 2020. https://doi.org/10.1109/ACCESS.2020.2987254

[8] R. Zakwandi and E. Istiyono, "A framework for assessing computational thinking skills in the physics classroom: Study on cognitive test development," *SN Social Sciences*, vol. 3, no. 3, 2023. https://doi.org/10.1007/s43545-023-00633-7

[9] N. O. Ezeamuzie, "Abstractive-based programming approach to computational thinking: Discover, extract, create, and assemble," *Journal of Educational Computing Research*, vol. 61, no. 3, pp. 605–638, 2022. https://doi.org/10.1177/07356331221134423

[10] R. Casado and M. Checa-Romero, "Robótica y proyectos STEAM: Desarrollo de la creatividad en las aulas de educación primaria," *Pixel-Bit. Revista de Medios y Educación*, vol. 58, pp. 51–69, 2020. https://doi.org/10.12795/pixelbit.73672

[11] D. A. Cueva Gaibor, "Educational technology in times of crisis," *Revista de Conrado*, vol. 16, no. 74, pp. 341–348, 2020.

[12] M. Cruz Lozano-Ramírez, "El aprendizaje basado en problemas en estudiantes universitarios," *Tendencias Pedagógica*, vol. 37, pp. 90–103, 2021. https://doi.org/10.15366/tp2021.37.008

[13] B. Tonbuloğlu and I. Tonbuloğlu, "The effect of unplugged coding activities on computational thinking skills of middle school students," *Informatics in Education*, vol. 18, no. 2, pp. 403–426, 2019. https://doi.org/10.15388/infedu.2019.19

[14] G. G. Verónica Jacqueline and E. F. Eudaldo Enrique, "Aprendizaje basado en problemas para el proceso de enseñanza-aprendizaje," *Revista Universidad y Sociedad*, vol. 14, no. 2, pp. 124–131, 2022.

[15] K. N. Huggett, "Teaching in small groups," in *An Introduction to Medical Teaching*, 2014, pp. 27–39. https://doi.org/10.1007/978-94-017-9066-6_3

[16] Husin *et al.*, "Project-based problem learning: Improving problem-solving skills in higher education engineering students," *Journal of Engineering Education Transformations*, 2025. http://dx.doi.org/10.17583/rise.15125

[17] Y. Watanobe *et al.*, "Online judge system: Requirements, architecture, and experiences," *International Journal of Software Engineering and Knowledge Engineering*, vol. 32, no. 6, pp. 917–946, 2022. https://doi.org/10.1142/S0218194022500346

[18] Md. M. Rahman, Y. Watanobe, A. Shirafuji, and M. Hamada, "Exploring automated code evaluation systems and resources for code analysis: A comprehensive survey," *Journal of the ACM*, vol. 37, no. 111, 2023. https://doi.org/10.1007/978-3-031-36822-6_33

[19] M. A. Revilla, S. Manzoor, and R. Liu, "Competitive learning in informatics: The UVa online judge experience," *Olympiads in Informatics*, vol. 2, pp. 131–148, 2008.

[20] S. Wasik, M. Antczak, J. Badura, A. Laskowski, and T. Sternal, "A survey on online judge systems and their applications," *ACM Computing Surveys*, vol. 51, no. 1, 2018. https://doi.org/10.1145/3143560

[21] Y. Watanobe, M. M. Rahman, T. Matsumoto, U. K. Rage, and P. Ravikumar, "Online judge system: Requirements, architecture, and experiences," *International Journal of Software Engineering and Knowledge Engineering*, vol. 32, no. 6, pp. 917–946, 2022. https://doi.org/10.1142/S0218194022500346

[22] S. Halim, "Competitive programming 4: The new lower bound of programming contests in the 2020s," *Olympiads in Informatics*, vol. 14, pp. 177–180, 2020. https://doi.org/10.15388/ioi.2020.14

[23] S. R. Maeng, "Educational effects of SW coding notes on computational thinking," *International Journal of Recent Technology and Engineering*, vol. 8, no. 2S6, pp. 270–274, 2019. https://doi.org/10.35940/ijrte.B1051.0782S619

[24] A. D. Samala, S. Papadakis, and S. Rawas, "Global insights into mobile learning in higher education: A PRISMA-guided bibliometric analysis from 2007 to 2023," *International Journal of Educational Reform*, pp. 1–25, 2025. https://doi.org/10.1177/10567879251341869

[25] S. A. M. Hogenboom, F. F. J. Hermans, and H. L. J. Van der Maas, "Computerized adaptive assessment of understanding of programming concepts in primary school children," *Computer Science Education*, vol. 32, no. 4, pp. 418–448, 2022. https://doi.org/10.1080/08993408.2021.1914461

[26] H. J. Park and Y. J. Jeon, "A design and application of software liberal arts course based on CT-CPS model for developing creative problem-solving ability and learning motivation of non-software majors," *International Journal on Informatics Visualization*, vol. 6, no. 2, pp. 317–326, 2022. https://doi.org/10.30630/joiv.6.2.996

[27] J. S. Krajcik and N. Shin, "Phases and activities of technology-integrated project-based learning in K-12: Findings from a systematic literature review," *Education Sciences*, 2022. https://doi.org/10.3390/educsci15081021

[28] A. B. Urbina-Najera, "Estrategia tecnológica para mejorar el rendimiento académico universitario," *Píxel-Bit. Revista de Medios y Educación*, vol. 56, pp. 71–93, 2019. https://doi.org/10.12795/pixelbit.2019.i56.04

[29] T. C. Hsu, S. C. Chang, and Y. T. Hung, "How to learn and how to teach computational thinking: Suggestions based on a review of the literature," *Computers & Education*, vol. 126, pp. 296–310, 2018. https://doi.org/10.1016/j.compedu.2018.04.010

[30] S. Papadakis, A. M. Striuk, H. M. Kravtsov, M. P. Shyshkina, M. V. Marienko, and H. B. Danylchuk, "Embracing digital innovation and cloud technologies for transformative learning experiences," in *Proc. 11th Workshop on Cloud Technologies in Education (CTE 2023)*, 2024, pp. 1–15.

9 AUTHOR

Rafael Ricardo Mantilla Guiza is a teacher-researcher in the Faculty of Engineering, Systems Engineering Program, at the Universidad de Investigación y Desarrollo (UDI). His primary research interests and areas of expertise include Educational Technology, Computer Science, emerging technologies, and the development of Computational Thinking (E-mail: rmantilla1@udi.edu.co).