E-ISSN: 2997-9382
American Journal of Technology Advancement
https://semantjournals.org/index.php/AJTA

| Research Article

Check for updates

Developing Computational Thinking in Secondary-School Informatics through a Blended Project-Based Learning Model: A Quasi-Experimental Study

Egamberdiyeva Zuhra Imomqulovna
Teacher at School No. 24, Samarkand City

Annotation

Computational thinking (CT) is widely recognised as a core competence of general education, yet school informatics instruction in many national systems remains oriented towards the syntax of a programming language rather than towards transferable problem-solving strategies. This study designs and empirically tests a blended project-based learning (BPBL) model for developing CT in lower-secondary informatics. The model integrates three sequential phases — a visual block-based phase, a hybrid block-to-text transition phase, and a collaborative project phase — with asynchronous online scaffolding and structured in-class feedback. A quasi-experimental pre-test/post-test design with non-equivalent groups was implemented over twelve weeks with 164 eighth-grade students (experimental group n = 82; control group n = 82) in four public schools. CT was measured with a 40-item criterion-referenced instrument covering abstraction, decomposition, algorithmic design, and debugging (Cronbach's α = .87); learning motivation was measured with an adapted ARCS-based scale (α = .91). Groups did not differ at pre-test, $t(162) = 0.29$, $p = .770$. At post-test the experimental group significantly outperformed the control group, $t(162) = 6.76$, $p < .001$, Cohen's $d = 1.06$; ANCOVA with pre-test scores as covariate confirmed the effect, $F(1, 161) = 48.36$, $p < .001$, partial $\eta^2 = .231$. The largest gains occurred in algorithmic design ($d = 1.33$) and abstraction ($d = 1.04$), the smallest in debugging ($d = 0.70$). Motivation also favoured the experimental group ($d = 1.02$). The findings indicate that sequencing representational transition and embedding CT components as explicit instructional objectives — rather than treating them as by-products of coding practice — produces substantial and broadly distributed learning gains.

Keywords: computational thinking; informatics education; blended learning; project-based learning; block-based programming; cognitive load; secondary education; quasi-experiment..

This is an open-access article under the CC-BY 4.0 license

INTRODUCTION

Since Wing's influential formulation of computational thinking as a universally applicable analytical attitude, CT has moved from the margins of computer science education to the centre of curricular reform in more than thirty national education systems. The underlying claim is that the mental operations exercised in programming — decomposing an ill-structured problem, abstracting away irrelevant detail, recognising recurring patterns, specifying an unambiguous procedure, and systematically diagnosing failure — constitute a general intellectual toolkit rather than a vocational specialisation.

The curricular claim, however, has outpaced the methodological one. Reviews of the field repeatedly observe that CT is frequently declared as an aim while instruction continues to be organised around the surface features of a particular programming language. Under such conditions learners acquire syntactic fluency without acquiring the strategic knowledge that CT is supposed to denote, and transfer to non-programming contexts remains weak. This gap is particularly visible in systems where informatics is taught with limited weekly hours and where teachers' own preparation emphasised language-specific skills.

Two design problems follow. The first concerns representation. Block-based environments reduce extraneous cognitive load by eliminating syntax errors, but an abrupt migration to text-based coding can undo the gains, since learners must simultaneously manage notation and logic. The second concerns task structure. Project-based learning promises authenticity and motivation, yet unguided project work risks leaving novices without the schema support they need, a concern raised persistently in the instructional-design literature.

The present study addresses both problems through a blended project-based learning model in which representational transition is deliberately staged and project autonomy is granted only after schema formation has been supported. The study asks three research questions. RQ1: Does the BPBL model produce higher overall CT attainment than conventional informatics instruction? RQ2: Are the effects uniform across the four CT sub-constructs? RQ3: Does the model affect learning motivation?

2. Theoretical Framework

2.1. Computational thinking as an instructional construct

Operationalisations of CT converge on a small set of components. The framework adopted here follows the widely used four-component decomposition — abstraction, decomposition, algorithmic design, and debugging/evaluation — because each component can be expressed as an observable performance and therefore assessed criterion-referentially. This is a deliberate narrowing: broader frameworks additionally include computational perspectives and dispositions, which are valuable but resist short-cycle measurement.

2.2. Representational transition and cognitive load

Cognitive load theory distinguishes intrinsic load, determined by element interactivity, from extraneous load imposed by the instructional format. In introductory programming, notation is largely extraneous with respect to the target schema. Block-based environments therefore function as a load-management device rather than as a simplified toy. The pedagogical risk is that learners build representation-bound schemas. The hybrid phase used in this study — in which the same problem is solved twice, first in blocks and then in text, with an explicit correspondence mapping — is designed to force schema abstraction across representations.

2.3. Blended project-based learning

Meta-analytic evidence supports moderate positive effects of project-based learning on academic achievement, and comparable evidence supports blended designs over purely face-to-face delivery. The mechanisms typically invoked — extended time on task, self-pacing, and richer feedback opportunities — are also the mechanisms most likely to benefit CT acquisition, which requires iterative revision. The blended component in this study is not a delivery convenience but a structural one: asynchronous scaffolds carry worked examples and self-checks, freeing classroom time for feedback dialogue and peer code review.

3. Materials and Methods

3.1. Research design

A quasi-experimental pre-test/post-test design with non-equivalent groups was employed. Random assignment of individual students was not feasible because instruction is delivered in intact classes; assignment was therefore performed at class level, with two classes in each school allocated to conditions by lot. The design is reported in accordance with standard quasi-experimental transparency expectations, including the acknowledgment of selection threats addressed analytically through ANCOVA.

3.2. Participants

Participants were 164 eighth-grade students (mean age 13.8 years, SD = 0.51; 51.2% female) from four public secondary schools. Table 1 summarises the distribution. All participating classes were taught by teachers with at least five years of experience; experimental-group teachers received a 16-hour preparatory workshop on the model.

Table 1. Distribution of participants across schools and conditions

School	EG (n)	CG (n)	Total	Female (%)
School A	21	20	41	48.8

School	EG (n)	CG (n)	Total	Female (%)
School B	20	22	42	52.4
School C	21	20	41	53.7
School D	20	20	40	50.0
Total	82	82	164	51.2

3.3. Instruments

CT was assessed with a 40-item criterion-referenced test constructed for this study, scored out of 100 points and balanced across the four sub-constructs (25 points each). Items were adapted from published CT assessment frameworks and validated by five subject experts; the content validity index was .92. Internal consistency in the present sample was acceptable (Cronbach's α = .87; sub-scales .74–.83). Motivation was measured with a 24-item five-point scale adapted from the ARCS motivational model (α = .91). Qualitative data comprised 18 semi-structured student interviews and teachers' lesson observation protocols.

3.4. The instructional intervention

The experimental condition implemented the BPBL model across twelve weeks (two 45-minute lessons weekly, 24 lessons in total). Phase 1 (weeks 1–4) used a block-based environment to establish sequence, selection, iteration, and event handling. Phase 2 (weeks 5–8) required each problem to be solved first in blocks and then in Python, with students maintaining a block-to-code correspondence table and studying faded worked examples. Phase 3 (weeks 9–12) allocated students to teams of three to develop a small applied artefact, with two mandatory code-review checkpoints and a final reflective report. Asynchronous materials — short screencasts, self-check quizzes with automated feedback, and an error-pattern catalogue — were available throughout. The control condition covered identical curricular content and received the same total instructional time using conventional lecture-and-exercise delivery in Python.

3.5. Data analysis

Assumptions of normality (Shapiro–Wilk) and homogeneity of variance (Levene's test) were satisfied for all outcome variables. Between-group differences were tested with independent-samples t-tests, complemented by ANCOVA using pre-test scores as covariate. Effect sizes are reported as Cohen's d and partial eta squared. Alpha was set at .05; sub-scale comparisons were Bonferroni-corrected. Analyses were conducted in SPSS 27. Interview transcripts were analysed thematically by two independent coders (Cohen's κ = .81).

4. Results

4.1. Baseline equivalence

At pre-test the experimental group (M = 52.14, SD = 9.86) and the control group (M = 51.68, SD = 10.21) did not differ significantly, t(162) = 0.29, p = .770, d = 0.05, supporting baseline comparability.

4.2. Overall computational thinking attainment

Post-test results are reported in Table 2. The experimental group achieved substantially higher scores, and the difference corresponds to a large effect.

Table 2. Pre-test and post-test computational thinking scores by condition

Measure	EG M (SD)	CG M (SD)	t(162)	p	d
Pre-test	52.14 (9.86)	51.68 (10.21)	0.29	.770	0.05
Post-test	74.63 (10.42)	63.27 (11.08)	6.76	< .001	1.06

Measure	EG M (SD)	CG M (SD)	t(162)	p	d
Gain	22.49 (8.13)	11.59 (8.74)	8.28	< .001	1.29

ANCOVA controlling for pre-test performance confirmed a significant effect of condition on post-test CT, $F(1, 161) = 48.36$, $p < .001$, partial $\eta^2 = .231$, indicating that approximately 23% of the variance in post-test scores was attributable to the instructional condition after adjustment.

4.3. Sub-construct analysis

Table 3 disaggregates post-test performance by CT component. All four comparisons remained significant after Bonferroni correction, but the magnitude of advantage varied considerably.

Table 3. Post-test performance by computational thinking component (max 25 points)

Component	EG M (SD)	CG M (SD)	t(162)	p	d
Abstraction	18.42 (2.86)	15.31 (3.11)	6.67	< .001	1.04
Decomposition	19.05 (2.64)	16.88 (2.97)	4.94	< .001	0.77
Algorithmic design	19.86 (2.42)	16.04 (3.28)	8.49	< .001	1.33
Debugging/evaluation	17.30 (3.05)	15.04 (3.42)	4.47	< .001	0.70

The strongest differential gain occurred in algorithmic design, which is the component most directly targeted by the staged representational transition. Debugging showed the weakest advantage and also the lowest absolute score in both conditions, a pattern consistent with reports that error diagnosis is rarely taught explicitly.

4.4. Learning motivation

The experimental group reported higher overall motivation ($M = 4.12$, $SD = 0.53$) than the control group ($M = 3.54$, $SD = 0.61$), $t(162) = 6.50$, $p < .001$, $d = 1.02$. Sub-scale inspection showed the largest difference on confidence and the smallest on attention. Interview data converged with this pattern: students in the experimental condition attributed confidence gains to being able to see a working solution before facing textual notation, whereas control-group students more frequently described early failure experiences as discouraging.

5. Discussion

The results support an affirmative answer to RQ1: the blended project-based model produced a large advantage in overall CT attainment, robust to covariate adjustment. The magnitude is at the upper end of what meta-analyses of project-based learning typically report, which suggests that the effect here is not attributable to project work alone. The most plausible interpretation is compositional — the blended asynchronous scaffolding supplied the schema support that unguided project work notoriously lacks, addressing the guidance critique directly rather than ignoring it.

The answer to RQ2 is negative, and instructively so. Effects were not uniform. The concentration of gains in algorithmic design and abstraction aligns with the mechanism proposed in Section 2.2: solving the same problem in two notations and constructing an explicit correspondence mapping compels learners to represent the procedure independently of its surface form, which is precisely what abstraction denotes. Conversely, the comparatively modest debugging effect suggests that the model's treatment of error diagnosis — confined largely to the error-pattern catalogue and peer review checkpoints — was insufficient. Debugging appears to require its own deliberate instructional cycle, including systematic hypothesis testing on deliberately faulty code.

Regarding RQ3, the motivational advantage, concentrated in the confidence dimension, is consistent with expectancy-based accounts of motivation. This finding carries a caution as well: motivational gains that derive from reduced early failure could in principle mask insufficient challenge. The parallel attainment gains argue against that reading in the present data, but the possibility warrants monitoring in longer implementations.

For practice, three implications follow. First, CT components should appear as explicit lesson objectives with matching assessment criteria, not as anticipated by-products of programming exercises. Second, the transition from block-based to text-based representation should be treated as a designed instructional event rather than an administrative switch. Third, teacher preparation is a non-trivial precondition; the intervention presupposed pedagogical content knowledge that cannot be assumed by default.

6. Limitations

Four limitations qualify these conclusions. Assignment was at class rather than individual level, so unmeasured class-level characteristics may have contributed to the observed differences. The twelve-week horizon does not permit claims about retention or far transfer. The CT instrument, although validated for this study, is not an established published measure, which constrains comparability with other reports. Finally, experimental-group teachers received preparatory training that control-group teachers did not, so a portion of the effect may reflect teacher preparation rather than the model itself; disentangling these contributions requires a design in which both conditions receive equivalent professional development.

7. Conclusion

This study designed and tested a three-phase blended project-based learning model for developing computational thinking in lower-secondary informatics. Across 164 students and twelve weeks, the model produced a large advantage in overall CT attainment and in learning motivation, with the strongest effects on algorithmic design and abstraction and the weakest on debugging. The central methodological argument is that CT develops when its components are made explicit instructional targets and when representational transition is staged rather than assumed. Future work should test the model over a full academic year, incorporate an explicit debugging strand, and examine transfer to non-programming problem-solving tasks.

Funding: This research received no specific grant from any funding agency.

Conflict of interest: The author declares no conflict of interest.

Ethical approval: The study was conducted with the approval of the participating schools; informed consent was obtained from parents and assent from students.

REFERENCES

[1] Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. Proceedings of the 2012 Annual Meeting of the American Educational Research Association, 1–25.

[2] Chen, C.-H., & Yang, Y.-C. (2019). Revisiting the effects of project-based learning on students' academic achievement: A meta-analysis investigating moderators. Educational Research Review, 26, 71–81.

[3] Cohen, J. (1988). Statistical power analysis for the behavioral sciences (2nd ed.). Lawrence Erlbaum.

[4] Deci, E. L., & Ryan, R. M. (2000). The "what" and "why" of goal pursuits: Human needs and the self-determination of behavior. Psychological Inquiry, 11(4), 227–268.

[5] Garrison, D. R., & Vaughan, N. D. (2008). Blended learning in higher education: Framework, principles, and guidelines. Jossey-Bass.

[6] Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. Educational Researcher, 42(1), 38–43.

[7] Hattie, J., & Timperley, H. (2007). The power of feedback. Review of Educational Research, 77(1), 81–112.

[8] Hsu, T.-C., Chang, S.-C., & Hung, Y.-T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. Computers & Education, 126, 296–310.

[9] Keller, J. M. (2010). Motivational design for learning and performance: The ARCS model approach. Springer.

[10] Kirschner, P. A., Sweller, J., & Clark, R. E. (2006). Why minimal guidance during instruction does not work. Educational Psychologist, 41(2), 75–86.

[11] Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? Computers in Human Behavior, 41, 51–61.

[12] Means, B., Toyama, Y., Murphy, R., & Baki, M. (2013). The effectiveness of online and blended learning: A meta-analysis of the empirical literature. Teachers College Record, 115(3), 1–47.

[13] Mishra, P., & Koehler, M. J. (2006). Technological pedagogical content knowledge: A framework for teacher knowledge. Teachers College Record, 108(6), 1017–1054.

[14] Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. Basic Books.

[15] Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., … Kafai, Y. (2009). Scratch: Programming for all. Communications of the ACM, 52(11), 60–67.

[16] Román-González, M., Pérez-González, J.-C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Computers in Human Behavior, 72, 678–691.

[17] Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. Educational Research Review, 22, 142–158.

[18] Sweller, J., van Merriënboer, J. J. G., & Paas, F. (2019). Cognitive architecture and instructional design: 20 years later. Educational Psychology Review, 31, 261–292.

[19] Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. Education and Information Technologies, 20(4), 715–728.

[20] Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33–35.

[21] Zhang, L., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. Computers & Education, 141, 103607.