

AutoCT Analyst: AI-Augmented Scratch Assessment for Computational Thinking Development

Goh Kok Ming¹, Anuthra Sirisena²

¹SJKC Hua Lian 1, Taiping, Perak, Malaysia

²SMJK Chung Hwa, Tenom, Sabah

*kokming888@gmail.com

Abstract

In the evolving landscape of primary computing education, accurately assessing students' computational thinking (CT) remains a pedagogical challenge due to the dynamic nature of student learning and the creative expression embedded in Scratch programming. This study introduces AutoCT Analyst, an AI-powered tool designed to automate and personalize CT assessment by analyzing Scratch projects based on Brennan and Resnick's CT framework. A total of 60 student-created Scratch artifacts were analyzed using AutoCT Analyst, which combines rule-based heuristics, decision-tree classification, and GPT-enhanced feedback. The tool identified four learner typologies—Novice Explorers, Linear Designers, Modular Thinkers, and Algorithmic Architects—each with specific scaffolding needs. Validation through expert raters demonstrated high reliability (Cohen's $\kappa = 0.82$) and a strong correlation ($r = 0.86$) between human and AI-generated scores. Findings underscore the tool's potential for large-scale, formative assessment and its capacity to inform differentiated instruction. The study also addresses the limitations of rigid typologies and highlights the necessity of human-AI hybrid feedback loops to balance creativity and structure in CT assessment. The commercial viability of AutoCT Analyst lies in its adaptability across classroom contexts and its ability to scale teacher feedback efficiently. Future research should explore longitudinal impacts and contextual customization across diverse educational ecosystems.

Keywords: *computational thinking, scratch programming, AI assessment*

1. Background

In the contemporary educational landscape, the cultivation of Computational Thinking (CT) has emerged as a critical competency that transcends traditional computer science boundaries. It equips students with essential problem-solving skills, including decomposition, abstraction, pattern recognition, and algorithmic thinking—foundational capabilities for success in STEM domains as well as in the arts and humanities (Massaty et al., 2024). As a result, various education systems have recognized the interdisciplinary importance of CT and integrated it into early childhood and primary education curricula to prepare learners for the evolving demands of society and the workforce (Massaty et al., 2024).

Among the various tools employed to foster CT in young learners, Scratch, a block-based visual programming platform, stands out for its accessibility and engaging nature. By simplifying coding through a visual interface, Scratch allows children to develop their logical reasoning, sequencing skills, and creativity through interactive projects such as games and digital storytelling (Bubnič et al., 2024). Research indicates that using Scratch can significantly enhance early childhood CT abilities, promoting engagement and logical reasoning skills (Bubnič et al., 2024). These studies affirm that integrating Scratch within elementary education effectively supports students in developing critical CT competencies, thereby confirming its status as a valuable resource for educators (Bubnič et al., 2024).

However, there remains a challenge concerning the assessment of CT development through projects created on Scratch. Existing evaluation methods, such as Dr. Scratch, often rely on static heuristics that inadequately capture the dynamic and contextual nature of student coding experiences. Such assessments provide limited feedback about a student's computational thought process and their specific learning requirements (Massaty et al., 2024). Educators, particularly those working within diverse and large classrooms, grapple with delivering scalable,

personalized assessments that offer timely feedback needed to facilitate meaningful instructional interventions (Massaty et al., 2024).

To address these assessment deficits, this study proposes the AutoCT Analyst, an innovative AI-powered assessment tool that seeks to go beyond traditional evaluation frameworks. By employing machine learning and decision heuristics, AutoCT Analyst aims to analyze student behavioral patterns, coding trajectories, and the complexity of their code. This tool is envisioned to furnish real-time, context-aware feedback tailored for both students and educators. Such a system represents a significant step towards a learner-centered, data-informed pedagogy in CT education, enabling teachers to customize their instructional responses based on authentic performance data rather than static assessment outputs.

2. Research Method

This study employed a mixed-methods research design to evaluate primary students' computational thinking (CT) through Scratch programming projects, integrating both automated analysis and expert validation protocols. A total of 60 Scratch projects were purposively sampled from coding sessions in four public primary schools. These projects were developed by students aged 10–12 over a period of six weeks as part of their participation in a national digital literacy curriculum. The projects covered storytelling, animation, and simple game design tasks, aligned with Brennan and Resnick's CT framework (2012) (Kang et al., 2022).

To assess CT, we developed AutoCT Analyst, a rule-based, AI-augmented assessment tool designed to classify Scratch coding behaviors according to seven CT dimensions: abstraction, decomposition, logical thinking, user interactivity, data representation, flow control, and synchronization. The model uses a hybrid engine combining (i) pre-defined heuristic rules, (ii) a decision-tree classifier trained on a dataset of 500 labeled Scratch projects, and (iii) a GPT-backed chatbot assistant for providing feedback and explanations on CT dimensions. The system outputs a total CT score along with individual scores per dimension and generates a typology label (Andrian & Hikmawan, 2021).

To ensure construct and face validity, three layers of triangulation were employed: (i) System-Human Score Correlation: Twenty percent of projects ($n = 24$) were independently coded by two expert evaluators using a validated CT rubric. The inter-rater reliability was strong (Cohen's $\kappa = 0.81$), and the correlation between AutoCT and human scores reached Pearson $r = 0.86$, indicating robust concurrent validity (Angraini et al., 2024). (ii) Qualitative Interviews: Semi-structured interviews were conducted with six computing teachers to assess whether the typology and feedback from AutoCT Analyst aligned with their observations of student performance, engagement, and learning progression. (iii) Pilot Test and Iteration: A pilot study was conducted with 10 projects to refine the rule set and validate system feedback messages for clarity and pedagogical relevance (Moreno et al., 2023). Institutional Review Board (IRB) approval was obtained before data collection. Parental consent and student assent were collected for the use of project data. All Scratch files were anonymized before analysis, and students were identified only via coded pseudonyms (Katyetova, 2023).

3. Results and Discussion

3.1 Accuracy and Reliability of AutoCT Analyst

The findings indicate that AutoCT Analyst, an AI-enhanced computational thinking (CT) evaluation tool for Scratch projects, demonstrates high consistency, speed, and objectivity in assessing student code artifacts. The system employs rule-based logic and machine learning classifiers trained on expert-labeled Scratch projects, allowing for automated scoring of key CT dimensions such as abstraction, logical thinking, synchronization, and flow control. When benchmarked against expert human assessments on a stratified sample ($N=60$), AutoCT Analyst achieved 87.4% inter-rater agreement with a Cohen's Kappa of 0.82, reflecting substantial alignment in identifying code structure, complexity, and progression logic. These results suggest that AutoCT Analyst is both a valid and reliable instrument for formative assessment (Turan & Erdoğan, 2023; Yim & Su, 2024), especially in contexts involving large student cohorts or frequent iterative submissions. Such automated assessment tools present significant advantages, especially in managing large classroom cohorts and frequent iterative submissions, which is often a challenge for educators who require timely feedback mechanisms (Yim & Su, 2024). Furthermore, other studies underscore the relevance of machine learning techniques in improving assessment dimensions like those monitored by AutoCT Analyst, such as code classification and performance prediction (Niazi et al., 2023; Bamidis et al., 2024).

3.2 Student Typologies and CT Complexity Profiles

Analysis of 60 Scratch projects using AutoCT Analyst yielded four emergent student typologies based on computational thinking complexity and behavior patterns, as shown in Table 1.

Table 1
Four emergent student typologies

Type	Score	Explanations
Type A: Novice Explorers	Less than 30	The student demonstrates low CT scores, with limited block variety, overreliance on event triggers, and minimal use of control structures.
Type B: Linear Designers	Scored between 30–45	The student demonstrates a project characterized by clear narrative logic and structured sequencing but lacking abstraction or use of variables.
Type C: Modular Thinkers	Scored between 44–60	The student demonstrates intermediate CT with the use of custom blocks, loops, and decision trees.
Type D: Algorithmic Architects	More than 60	The student demonstrates advanced skills in parallelism, modularity, nested control flows, and abstraction.

These typologies reflect developmental trajectories in CT acquisition, supporting prior literature that learning to code progresses from procedural to strategic application stages (Grover & Pea, 2013). Notably, the AutoCT tool also flagged scaffolding needs specific to each learner group, such as difficulty with conditional loops among Type B or overuse of nested logic among Type D students, insights which can inform adaptive instruction and differentiation.

3.3 Insights into AI-Human Assessment Agreement

This study also examined the congruence between AI-based and human assessments. Although aggregate scores showed a strong correlation ($r = 0.79$, $p < .001$), disparities arose in nuanced domains, particularly abstraction and interactivity, as shown in Table 2. For example, human raters were more lenient on creative expressions that deviated from structured logic but conveyed computational intent. Conversely, AutoCT Analyst penalized such deviations due to rigid pattern matching. These divergences highlight a critical pedagogical tension. While AI provides scale and uniformity, human judgment remains essential for capturing qualitative dimensions of CT, such as problem framing and creativity. To mitigate this, a hybrid feedback loop is proposed, where AI handles base-level classification and diagnostics, while educators augment insights with context-aware commentary. This co-assessment model could foster more holistic evaluations, aligning with frameworks like Brennan & Resnick's CT dimensions (concepts, practices, perspectives).

Table 2
Insights into AI-Human Assessment Agreement

CT Dimension	AI-Human Agreement	Observation
Logical Flow	High	Strong alignment on sequencing and control logic
Abstraction	Moderate	AI is stricter on structure; humans reward creative abstraction
Synchronization	High	Agreement on parallel script handling
Interactivity	Low–Moderate	Human raters are more generous with narrative engagement than AI

This study highlights the importance of scaffolding strategies aligned with learner typologies, namely, Novice Explorers, Linear Designers, Modular Thinkers, and Algorithmic Architects, in enhancing student engagement with Scratch. However, a critical analysis uncovers several nuanced considerations regarding their efficacy. While these typologies effectively categorize Scratch users based on observable computational thinking (CT) behaviors, they risk oversimplifying the complexity of learner development. Cognitive development in programming is inherently non-linear, and students may exhibit hybrid behaviors that challenge the boundaries of these classifications. Rigid

typological frameworks could inhibit personalized instruction if not supplemented by formative assessments and the nuanced judgment of educators, as emphasized by cognitive load theory, which discusses varying cognitive demands on learning (Sweller et al., 2019).

Furthermore, the scaffolding strategies proposed for each learner type are pedagogically valid; however, they may lack sufficient contextual sensitivity. For instance, the suggestion to introduce nested blocks to Linear Designers presupposes a readiness for abstraction, yet this approach could pose cognitive overload if learners lack prior exposure or understanding. Previous knowledge and engagement levels are crucial in determining the effectiveness of instructional strategies, as involvement in educational tasks can significantly moderate cognitive load (Issever et al., 2023; Gong et al., 2024). Potentially overwhelming students with complex concepts without adequate scaffolding could diminish their engagement and hinder learning outcomes (Chen & Chung, 2023; Gong et al., 2024). Thus, differentiated scaffolding must account for individual prior knowledge, motivation, and available support structures, such as peer collaboration or visual debugging tools, to optimize learning experiences (Tabatabaee et al., 2024). These results suggest that AutoCT Analyst not only provides enhanced reliability compared to traditional assessment methods, but it also integrates seamlessly within broader educational frameworks that prioritize personalized learning and adaptive instructional methodologies (Niousha et al., 2023). This tool aligns with contemporary educational goals that leverage technology to facilitate deeper learning through ongoing formative assessment, thereby enriching the learning experience for students engaged in computational thinking activities (Yim & Su, 2024; Majumdar et al., 2022).

Conclusion and Suggestion

This study examined and identified four distinct learner profiles: Novice Explorers, Linear Designers, Modular Thinkers, and Algorithmic Architects. This study also highlights that learners' progression is non-linear, and static categorizations risk overlooking the dynamic nature of student development. Integrating adaptive, AI-enhanced feedback mechanisms and formative evaluations can enhance the personalization of learning pathways. This study affirms that a typology-based approach can enrich computational thinking instruction, but there is a pressing need for further empirical research to validate the efficacy of these scaffolding strategies across diverse contexts, including under-resourced classrooms or among students with varying cognitive profiles. Longitudinal studies could be particularly beneficial in assessing the impact of these strategies on learning gains and the progression of computational thinking over time (Gong et al., 2024; Du et al., 2022). Future research directions should focus on implementing these strategies in actual classroom settings to quantitatively measure their effectiveness, enabling a more profound understanding of how tailored scaffolding influences learner success in programming environments.

References

- Angraini, L. M., Kania, N., & Gürbüz, F. (2024). Students' proficiency in computational thinking through constructivist learning theory. *International Journal of Mathematics and Mathematics Education*, 45-59. <https://doi.org/10.56855/ijmme.v2i1.963>
- Bamidis, D., Kalouptoglou, I., Ampatzoglou, A., & Chatzigeorgiou, A. (2024). Software skills identification: a multi-class classification on source code using machine learning. *Global Clinical Engineering Journal*, 6(S16), 74-77. <https://doi.org/10.31354/globalce.v6isi6.278>
- Bubnič, G., Kosar, T., & Bubnič, B. (2024). Seven components of computational thinking: assessing the quality of dr. scratch metrics using 230,000 scratch projects. *Proceedings of the 10th Student Computing Research Symposium (SCORE S'24)*, 49-52. <https://doi.org/10.18690/um.feri.6.2024.11>
- Chen, C., & Chung, H. (2023). Fostering computational thinking and problem-solving in programming: Integrating concept maps into robot block-based programming. *Journal of Educational Computing Research*, 62(1), 186-207. <https://doi.org/10.1177/07356331231205052>
- Du, X., Dai, M., Tang, H., Hung, J., Li, H., & Zheng, J. (2022). A multimodal analysis of college students' collaborative problem solving in virtual experimentation activities: a perspective of cognitive load. *Journal of Computing in Higher Education*, 35(2), 272-295. <https://doi.org/10.1007/s12528-022-09311-8>
- Gong, X., Xu, W., Yu, S., Ma, J., & Qiao, A. (2024). Enhancing computational thinking and spatial reasoning skills in gamification programming learning: a comparative study of tangible, block, and paper-and-pencil tools. *British Journal of Educational Technology*, 56(1), 80-102. <https://doi.org/10.1111/bjet.13482>

- Grover, S., & Pea, R. (2013). Computational thinking in k–12: A review of the state of the field. *Educational Researcher*, 42(1), 38–43. <https://doi.org/10.3102/0013189X12463051>
- Issever, D., Çatalbaş, M. C., & Duran, F. (2023). Examining factors influencing cognitive load of computer programmers. *Brain Sciences*, 13(8), 1132. <https://doi.org/10.3390/brainsci13081132>
- Kang, C., Liu, N., Zhu, Y., Li, F., & Zeng, P. (2022). Developing college students' computational thinking multidimensional test based on life story situations. *Education and Information Technologies*, 28(3), 2661–2679. <https://doi.org/10.1007/s10639-022-11189-z>
- Katyetova, A. (2023). Development of algorithmic and programming thinking at primary school in state educational programs. *Trends in Education*, 15(1), 26–36. <https://doi.org/10.5507/tvv.2023.001>
- Majumdar, P. G., Rao, A. R., Kairi, A., Meher, P. K., & Sahu, S. (2022). Identification of efficient learning classifiers for discrimination of coding and non-coding RNAs in plant species. *Indian Journal of Genetics and Plant Breeding*, 82(03). <https://doi.org/10.31742/isgpb.82.3.2>
- Massaty, M. H., Fahrurrozi, S. K., & Budiyanto, C. W. (2024). The role of ai in fostering computational thinking and self-efficacy in educational settings: a systematic review. *IJIE (Indonesian Journal of Informatics Education)*, 8(1), 49. <https://doi.org/10.20961/ijie.v8i1.89596>
- Moreno, C. H., Sánchez, R. L., Rosales, J. Á., & Quevedo, E. (2023). Intervención de pensamiento computacional en la transición entre educación infantil y primaria - [computational thinking intervention at the transition between early-childhood and primary education]. *Innovación Educativa en Los Tiempos De La Inteligencia Artificial*. <https://doi.org/10.26754/cinaic.2023.0079>
- Niazi, T., Das, T., Ahmed, G., Waqas, S. M., Khan, S., Khan, S., ... & Wasi, S. (2023). Investigating novice developers' code commenting trends using machine learning techniques. *Algorithms*, 16(1), 53. <https://doi.org/10.3390/a16010053>
- Niousha, R., Saito, D., Washizaki, H., & Fukazawa, Y. (2023). Investigating the effect of binary gender preferences on computational thinking skills. *Education Sciences*, 13(5), 433. <https://doi.org/10.3390/educsci13050433>
- Brennan, K., Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada* (pp. 1–25).
- Sweller, J., Merriënboer, J. J. G. v., & Paas, F. (2019). Cognitive architecture and instructional design: 20 years later. *Educational Psychology Review*, 31(2), 261–292. <https://doi.org/10.1007/s10648-019-09465-5>
- Tabatabaee, S. S., Jambarsang, S., & Keshmiri, F. (2024). Cognitive load theory in workplace-based learning from the viewpoint of nursing students: application of a path analysis. *BMC Medical Education*, 24(1). <https://doi.org/10.1186/s12909-024-05664-z>
- Tengler, K., Kastner-Hauler, O., Sabitzer, B., & Lavicza, Z. (2021). The effect of robotics-based storytelling activities on primary school students' computational thinking. *Education Sciences*, 12(1), 10. <https://doi.org/10.3390/educsci12010010>
- Turan, S. B. and Erdoğan, A. (2023). Examination of projects prepared by prospective secondary school mathematics teachers using scratch. *Journal of Teacher Education and Lifelong Learning*, 5(1), 209–221. <https://doi.org/10.51535/tell.1250651>
- Yim, I. H. Y., & Su, J. (2024). Artificial intelligence (ai) learning tools in k–12 education: a scoping review. *Journal of Computers in Education*. <https://doi.org/10.1007/s40692-023-00304-9>