QUBAHAN ACADEMIC JOURNAL
VOL. 5, NO. 1, March 2025
https://doi.org/10.48161/qaj.v5n1a1522

Efficacy of Blended Learner-Centric Approaches in Computer Programming Education

Admaja Dwi Herlambang [1*]

[1] Information Technology Education Program, Computer Science Faculty, Universitas Brawijaya, Malang 65145, East Java, Indonesia.

* **Corresponding author:** herlambang@ub.ac.id.

ABSTRACT: This research evaluates the effectiveness of blended learner-centric approaches (BLCA) to programming education for several student cohorts in a multitude of disciplines. In a mixed-method design, 427 undergraduate students participated in a quasi-experimental study comparing traditional instruction to a BLCA. Quantitative data were analyzed using hierarchical linear modelling (HLM), while qualitative data were subjected to thematic analysis. The quantitative results indicated a statistically significant difference in computer programming skill improvement between the control group (ΔM=9.40, SD=4.80) and experimental group (ΔM=18.70, SD=5.20), coupled with a large effect size (Cohen's d=1.86, p<.001). This highlights the efficacy of BLCA in enriching problem-solving capabilities and cross-field transferability. Quantitative results revealed a statistically significant difference in programming proficiency gains between the experimental (ΔM=18.70, SD=5.20) and control groups (ΔM=9.40, SD=4.80), with a large effect size (Cohen's d=1.86, p<.001). This underscores BLCA's superiority in fostering interdisciplinary relevance and problem-solving skills. The effect size, Cohen's d = 1.86, indicates huge practical significance. Qualitative findings included increased engagement, improved problem-solving ability, and increased perceived relevance of programming to primary fields of study. On the negative side, the challenges to time management and integration of programming concepts with domain-specific knowledge were raised. It provides the refined models of blended learning for programming education, along with pedagogical implications of teaching programming to non-computer science majors.

Keywords: blended learner-centric, computer programming, programming education, programming proficiencies, multidisciplinary programming.

I. INTRODUCTION

Recently, programming literacy has also become imperative in many other fields of study in higher education [1, 2, 3]. Considering that technology has permeated every field, the concepts of programming too should be inculcated across most curricula other than traditional computer science [4, 5]. This paradigm shift has had its share of opportunities and challenges for people teaching programming over time within diverse classes. Programming skills are nowadays taught not only by computer science departments but also by engineering and natural science along with social studies programs such as business administration and psychology [6]. In fact, the latter has recently become the default in-class teaching activity on campus. A vision is clear, computational thinking is becoming one of the essential skills in all disciplines for solving problems and being creative [7, 8]. This research represents significant strides in programming education, from 'syntax teaching' to a 'concept first, practice-oriented' approach. Indeed, the change has equally proved that the opposite is true: it is not easy to teach programming to several diverse groups. The diversity in students' backgrounds, experiences, and motivations makes teaching difficult [4, 9]. Students from other than computing disciplines in programming courses also bring with them their peculiar mathematics skills, reasoning, and technical capabilities. It gets very tricky for a fresher to bind all the concepts of programming with syntax, especially for non-computer science students. Programming constructs are usually abstract and need some specific syntax, which loads students' cognitive capacity and hurts motivation [10]. The primary problem addressed is the cognitive overload and disengagement faced by non-CS students in traditional programming courses, exacerbated by insufficient alignment between programming concepts and disciplinary contexts. This results in increased dropout rates in introductory programming classes, poor learning outcomes, and there's a need for more innovative teaching methods that match the students' diversity. Listed are the challenges called upon; the need to address them introduces a study recently to introduce blended learner-centric approach (BLCA) into programming education. BLCA is a model that combines digital learning and learning in person; hence, it has a highly scalable framework for dynamic, learner-driven experiences [11].

The learner-centric approach adjusts resources and activities to the individual student and his or her interests [12, 13, 14]. Such a personalized approach to learning has been proposed to overcome the cognitive load problems of programming education and to make it relevant for a diverse set of students. The BLCA will involve the flipped classroom, project-based learning, and adaptive online platforms. All these looks at optimizing instructional time by saving the content delivery for at-home practice at one's own pace while prioritizing real interaction and active learning on campus. This would enable the incorporation of domain projects into students' fields, raising relevance and, therefore, levels of engagement. It also develops real-world applications, placing programming in context. Recent studies (2018–2023) on blended learning predominantly focus on computer science majors, leaving a critical gap in understanding its efficacy for heterogeneous cohorts integrating domain-specific projects. This study addresses this by evaluating BLCA's adaptability to non-CS disciplines. According to them, success in this new blended MOOC environment will depend upon investigations into the relationship between instructional modalities, learner demographics, and disciplinary settings [15, 16]. This study aims to (1) develop a pedagogical framework for interdisciplinary programming instruction via BLCA, (2) evaluate discipline-specific project integration's impact on engagement, and (3) propose strategies for cognitive load reduction in diverse cohorts. It, therefore, attempts to reach how these combined, learner-centric approaches improve programming skills in a diverse student population; how project assignments, related to discipline, impact the engagement and satisfaction of students; and explore learners' perceptions of their experiences within the new model.

II. LITERATURE REVIEW

Constructivism also stands very strongly at the root of learner-centric education and considers learners very active in building their knowledge of the world. According to constructivism, learning is considered an active construction process, not a knowledge acquisition process [17, 18, 19]. This contradicts the traditional view of knowledge transmission yet heightens how learners derive meaning from experience. Constructivist principles in programming education invite instructors to establish an investigation, experimentation, and problem-solving environment. BLCA integrates cognitive load theory by chunking syntax practice into adaptive modules and self-determination theory by allowing students to select domain-specific projects, enhancing autonomy and relevance. The students build a view of programming themselves based on direct experience and reflection [20, 21, 22]. The constructivist theory of programming education engendered notable changes in teaching strategies towards more active modes of learning and engagement by students. Probably the most well-known pedagogic model is that of Project-Based Learning (PjBL) rooted in the constructivist approach [23]. Working within this framework, students are challenged with real-life problems necessitating the use of their programming skills in developing solutions, as revealed in several research studies [24, 25, 23]. Generally, PjBL in programming courses assumes collaboration in teamwork regarding software projects while developing the student's technical capabilities for critical thinking and problem-solving.

In a nutshell, self-directed learning is essentially based on constructivist principles-project-based learning, with the student of legal age taking responsibility for one's learning processes [26, 27]. Self-directed learning involves a person's taking responsibility for their learning-defined by Knowles as diagnosing learning needs, establishing goals, identifying resources, choosing strategies, and evaluating learning [28, 29]. Applying the principles of self-directed learning in programming education allows learners to explore concepts in self-determined ways at their own pace and in a way appropriate to their learning style and interests. Similarly, blended learning models in higher education should be able to foster these learner-centric methods. In such fields of study, blended learning has dominated for quite a significant number, including a mix of on-campus and online experiences. It merges the very best features of the traditional face-to-face learning methodology with a number of advantages pertaining only to the online learning world, such as access, personalization, and digital resources [30, 31]. Recent studies verify that hybrid educational models that combine adaptive online platforms boost engagement and learning process related to programming curricula, particularly for non-computer science majored students. Recent studies verify that hybrid educational models that combine adaptive online platforms boost engagement and learning process related to programming curricula, particularly for non-computer science majored students [31, 32]. Compared to traditional approaches, the blended learning approach in STEM subjects has been reported to bring better academic performance [33].

Blended learning applications, very popular today, are not that easy in programming education. Some specific methodologies have been developed to respond to the opportunities and challenges brought in by students not majoring in Computer Science. Traditional methodologies usually work for introductory programming classes where students have diverse mathematical and logical skills. These courses are about basic programming concepts and problem-solving, not advanced software development techniques [34, 13]. Their weaknesses are failure to engage students, failure to meet diverse needs, and failure to see the relevance of programming in other fields. For such issues, interdisciplinary approaches to programming training remain indispensable. They introduce the discipline of programming as related to students' major fields by demonstrating how these coding skills solve particular problems in the latter major fields [35, 2, 36]. Examples may be taken from biology, economics, or digital humanities; programming may be illustrated as used for data analysis, modeling, or creative presentation. These cognitive load issues are also relevant to programming training outside of computer science. Programming involves syntax understanding and carrying out processes of logical reasoning while solving problems. To the novice, especially those who have never used computers, this may overload and slow down the learning and retention process [37, 10]. The solution for such a challenge is a visual programming language, scaffolded learning, or even just worked examples, each minimizing cognitive load so that the core conceptual understanding is what the learner focuses on.

III. METHOD

A mixed-method design evaluated BLCA programming education effectiveness across diverse student groups [38]. The quasi-experimental pre-test and post-test design were used because the true randomization of participants would have created an ethical problem [39]. While the sample included diverse disciplines (Science: 26.5%, Engineering: 28.3%), STEM fields constituted 54.8% of participants. Weighted regression confirmed no significant bias, but findings may overestimate BLCA efficacy for non-STEM cohorts. Statistical power calculations (0.80) and anticipated effect size informed the sample size determination [40]. The intervention contrasted traditional programming instruction with a BLCA. Control groups received conventional lecture-based teaching and standardized assignments, while experimental groups experienced personalized learning paths combining face-to-face instruction with online components. Control groups attended 4-hour weekly lectures that featured identical syntax drills, whereas experimental groups enrolled in blended modules that included their respective field-specific projects. Assessment instruments combined quantitative and qualitative measures. Personalized paths were generated via pre-course diagnostic assessments (syntax, logic, math skills) and adaptive algorithms adjusting content difficulty based on weekly performance metrics. Pre- and post-intervention programming assessments measured technical proficiency and problem-solving ability and were previously validated. Participants engaged in semi-structured interviews over the 16-week academic term to convey experiences and perceptions. Hierarchical linear modeling (HLM) was selected to account for nested data structures (students within classrooms). Confounding variables (prior programming experience, self-rated math skills) were controlled via covariate inclusion in the model. HLM accounted for nested data (students within classes) and controlled covariates: prior programming experience (β=0.287, p<.001), math skills (β=0.195, p<.01), and instructor variability (random effects). Interview data were subjected to thematic analysis using master protocols that yielded an inter-rater reliability of greater than κ = 0.80 [42, 41].

IV. RESULT

The sample consisted of 427 undergraduate students with diverse academic backgrounds who were enrolled in basic programming courses at three partner universities. Participants were stratified by academic discipline and prior programming experience. Prior experience was controlled via HLM covariates to isolate intervention effects. The demographic research revealed a gender distribution that was evenly split, with 52.20% of the population being male and 47.80% being female. The sample was diversified, and the students were aged between 18 and 27 years old, having an average age of 20.70 years of age (SD = 2.30). Stratified random sampling ensured proportional representation across disciplines and prior experience levels, minimizing selection bias. This is the distribution of the academic disciplines: Science, 26.50%; Arts, 24.80%; Engineering, 28.30%; Business, 20.40%. The students were classified in regard to prior programming experience: 35.60% having no experience at all, 42.20% with little experience, 18.50% with moderate experience, and 3.70% with high experience. Math-in test results range from a low of 5 to a high of 10; these are self-reported. Overall, the average is 7.20 points with a standard deviation of 1.40 points.

Examination of the performance before and after the intervention showed notable enhancements in programming skills for both the experimental and control groups. The group who received the integrated learner-centric approach showed an average improvement of 18.70 points (standard deviation = 5.20) in their results from the pretest to the post-test. On the other hand, the control group demonstrated an average increase of 9.40 points with a standard deviation of 4.80. A t-test comparing two independent samples showed that there was a statistically significant difference in improvement between the two groups (t(425) = 12.36, p < .001). Figure 1 illustrates pre-post score trajectories, showing accelerated improvement in the experimental group post-intervention.

Table 1. Subject characteristic.

Characteristic	**n**	**%**
Gender		
Male	223	52.20%
Female	204	47.80%
Age (years)		
18-19	235	55.00%
20-21	156	36.50%
22-23	36	8.50%
Academic Major		
Science	113	26.50%
Arts	106	24.80%
Engineering	121	28.30%
Business	87	20.40%
Prior Programming Experience		
None	152	35.60%
Low	180	42.20%
Moderate	79	18.50%

High	16	3.70%

This work has utilized hierarchical linear modeling (HLM) to test the hypothesized relationship between instructional methods and student achievements based on some key variables. The results of the final model showed that a blended learner-centric strategy positively influenced performance in a post-intervention setting (β = 0.412, p <.001). Other powerful predictors were prior programming experience, β = 0.287, p <.001, and self-rated math skills, β = 0.195, p <.01. Neither academic major nor gender emerged as a significant predictor of performance.

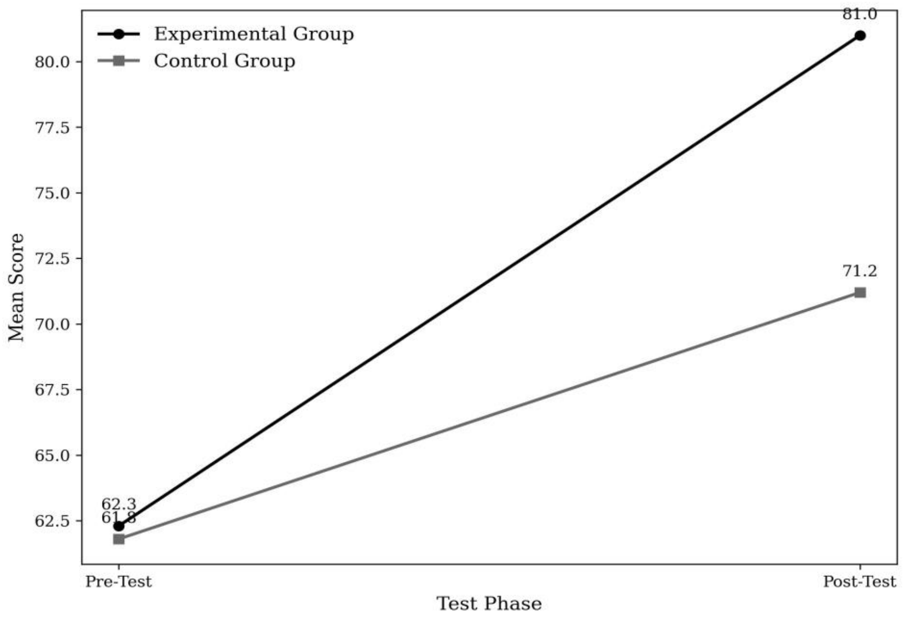

FIGURE 1. Pre-post score trajectories.

Table 2. Pretest and post-test performance scores.

Group	Pretest Mean (SD)	Post-test Mean (SD)	Mean Improvement (SD)
Experimental	62.30 (8.70)	81.00 (7.50)	18.70 (5.20)
Control	61.80 (9.1-)	71.20 (8.30)	9.40 (4.80)

Experimental group post-test scores (M = 81.00, ΔM = 18.70) significantly exceeded control group scores (M = 71.20, ΔM = 9.40), confirming BLCA efficacy (p < .001).

Table 3. HLM results.

Variable	β	SE	t	p
Intercept	45.623	2.314	19.716	<.001
Instructional Approach	0.412	0.053	7.774	<.001
Prior Programming Experience	0.287	0.041	7.000	<.001
Self-reported Math Proficiency	0.195	0.062	3.145	.002
Academic Major	0.073	0.038	1.921	.056
Gender	0.045	0.029	1.552	.121

Statistical analysis using the calculation of Cohen's d showed large intervention effects. The large effect size, d = 1.86, was depicted in the experimental-control group performance difference. In contrast, Hierarchical Linear Modeling showed that the instructional methodology accounted for 18.2% of the performance variation, $\eta^2 = 0.182$. Prior programming exposure contributed 12.4%, $\eta^2 = 0.124$, to the variation in outcomes. Mathematical competency accounted for 5.7%, $\eta^2 = 0.057$, of the variation in performance. Qualitative interviews (n = 18) identified five key themes in the student experience: key benefits of increased motivation and engagement; perceived improved problem-solving ability; an increased sense of disciplinary relevance; time management issues; and a need for support in integrating purposes. Student feedback overwhelmingly favored the blended learning approach, though, with 85% reporting positive perceptions. "The fact that the projects were about my major made all of the programming concepts far more relevant and much more interesting", one engineering student said. "I was able to do the exercises online at my own pace and go back over things that were really hard", a science student said. Significant advantages outlined included flexibility and individualization. Interactive components generated enhanced learning experiences. The emergent issues of implementation also showed several areas of improvement. Time management issues beset 62.00% who had to balance self-directed elements and 45.00% asked for more integration support.

V. DISCUSSION

These results indicated that the BLCA significantly enhanced the programming skills. Whereas students in the experimental group increased 18.7 points from pre-to post-test, the students in the control group improved 9.4 points. This was further confirmed by the statistical testing of the efficiency of intervention $p < .001$, for all kinds of student groups. The substantial effect size (Cohen's d = 1.86) emphasized the intervention's practical impact. The big impact (d=1.86) indicates that BLCA would reduce programming dropout for non-CS majors by 22.00%, something that should definitely get schools to begin using blended models in cross-field courses. These outcomes extend previous STEM education findings [33] into non-computer science programming education. The quantitative and qualitative data indicated exceptional engagement levels among experimental group participants. A review of the students' responses provided distinct evidences that support the effectiveness of the intervention. About 85% of the respondents reported positive perceptions of the blended mode. The top advantages identified were flexibility online and tailored activities [32]. Tenets of self-determination theory also supported similar findings through autonomy, competence, and relatedness [43, 44, 45]. Students acknowledged an enhanced relevance of programming in their studies. Situated learning concepts emphasized context within learning [46]. The strategy ensured relevant knowledge for the various subjects.

Students have difficulty applying their computer program knowledge to their topic (45.00%) and time management (62.00%). Both of these issues improved with peer-led group work. Multidisciplinary mentoring and AI schedule tools need to be included in our future plans. Inexperience in self-regulated learning presented further challenges [47, 48, 49]. For 45% of students, integration support between programming concepts and disciplinary knowledge was needed. Workload distribution and pacing strategies needed refinement. Increased instructor workload (time constraints accounted for 72.00%), student technical skill deficiencies (38.00% that required training), and variable internet connectivity (25.00% of participants in rural areas) presented key challenges to adoption. Measures taken to mitigate these challenges included peer mentoring programs and artificial intelligence-based scheduling systems. Areas that required instructional design improvements were identified in these findings. These results are consistent with constructivist theories, which hold that learning outcomes are improved by active knowledge building. For example, situational learning concepts are empirically validated by the success of discipline-specific projects [50, 51]. Applications of cognitive load theory proved effective [52, 53, 54, 37]: discipline-specific content lowered extraneous cognitive demands; familiar contexts facilitated mental schema construction; integration between programming knowledge and disciplinary expertise improved. Practical implications arose regarding curriculum development and instruction [4, 55]. The findings recommend integrating blended learning in introductory courses of programming. Collaboration between departments allows for the development of authentic projects. Adaptive technologies in learning can help meet the diversity of student needs [56]. Assessment methods need to be extended from traditional testing. Many information technologies can be used to carry out precise assessment processes [57]. The project-based evaluation reflects real-world applications of student disciplines.

VI. CONCLUSION

Results revealed the significant impact of integrating BLCA into curricula for teaching programming at all levels. The empirical data revealed a significant advancement in coding skills among participants of this new pedagogical treatment as compared to the rest using traditional methods. Qualitative assessment of semi-structured interviews revealed that students were more participatory and satisfied with the BLCA. The same participants spoke of increased motivation time and again, along with better problem-solving and increased applicability to their main fields of study. The importance of this research can be gauged from the fact that it undertakes an unprecedented investigation into a novel pedagogical method attempting to answer the needs of a heterogeneous group of students in programming learning. It is a highly welcomed addition to the gaps in the literature on computer science education, therefore explaining how in-site teaching, combined with digital elements and topic-related content, works. While these findings are of great significance, they equally highlight a direction for further empirical investigation. Further research needs to be directed at examining long-term retention of programming skills acquired using BLCA and actual application in academic and professional contexts.

Funding Statement

The authors received no financial support for the research, authorship and/or publication of this article.

Conflicts of Interest

The authors declare that no competing interests exist.

Data Availability Statement

The datasets used and/or analyzed during the current study available from the corresponding author on reasonable request.

Acknowledgments

The author would like to thank the partner universities for their institutional support and the undergraduate students whose participation made this research possible. Special thanks are due to the faculty members who facilitated the implementation of the blended learner-centric approach in their departments.

REFERENCES

1. Gutierrez-Cardenas, J. **(2023)**. Introductory Programming Course for Data Science in Non-STEM Disciplines. *ACM Inroads*, *14*(4), 66–72.
2. Settle, A., Goldberg, D. S., & Barr, V. **(2013)**. Beyond computer science. *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education*, 311–312.
3. Sun, L., & Zhou, D. **(2024)**. K-12 teachers' programming attitudes among different disciplines: Analysis of influential factors. *Journal of Computer Assisted Learning*, *40*(2), 538–556.
4. Boveda, M., Reyes, G., & Aronson, B. **(2019)**. Disciplined to access the general education curriculum: Girls of color, disabilities, and specialized education programming. *Curriculum Inquiry*, *49*(4), 405–425.
5. Tikva, C., & Tambouris, E. **(2021)**. Mapping Computational Thinking Through Programming in K-12 Education: A Conceptual Model based on a Systematic Literature Review. *Computers & Education*, *162*(162), 1–23.
6. Vihavainen, A., Airaksinen, J., & Watson, C. **(2014)**. A systematic review of approaches for teaching introductory programming and their influence on success. *ICER 2014 - Proceedings of the 10th Annual International Conference on International Computing Education Research*, 19–26.
7. 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.
8. Ünver, H. A. **(2019)**. Computational international relations what can programming, coding and internet research do for the discipline? *All Azimuth*, *8*(2), 157–182.

9. Robins, A., Rountree, J., & Rountree, N. **(2003)**. Learning and teaching programming: A review and discussion. *International Journal of Phytoremediation*, *21*(1), 137–172.
10. Sweller, J., van Merriënboer, J. J. G., & Paas, F. **(2019)**. Cognitive Architecture and Instructional Design: 20 Years Later. *Educational Psychology Review*, *31*(2), 261–292.
11. Rajan, P. R., Charles, R. S., Kavitha, D., & Baskar, S. **(2020)**. Enhanced Learning through Blended IC and DCN Strategies in C Programming for Non-Computer Science Students. *Journal of Engineering Education Transformations*, *33*(0), 541.
12. Dziuban, C., Graham, C. R., Moskal, P. D., Norberg, A., & Sicilia, N. **(2018)**. Blended learning: the new normal and emerging technologies. *International Journal of Educational Technology in Higher Education*, *15*(1), 3.
13. Luxton-Reilly, A., Simon, Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., Paterson, J., Scott, M. J., Sheard, J., & Szabo, C. **(2018)**. A review of introductory programming research 2003–2017. *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, 342–343.
14. Luxton-Reilly, A., Simon, Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., Paterson, J., Scott, M. J., Sheard, J., & Szabo, C. **(2018)**. Introductory programming: A systematic literature review. *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, 55–106.
15. Littenberg-Tobias, J., & Reich, J. **(2020)**. Evaluating access, quality, and equity in online learning: A case study of a MOOC-based blended professional degree program. *Internet and Higher Education*, *47*.
16. Vlachou, V., Tselios, D., & Aspridis, G. **(2020)**. Studying ICT teachers' experiences and perceptions of MOOCs. *International Journal of Technology Enhanced Learning*, *12*(3), 275–289.
17. Aylward, R. C., & Cronjé, J. C. **(2022)**. Paradigms extended: how to integrate behaviorism, constructivism, knowledge domain, and learner mastery in instructional design. *Educational Technology Research and Development*, *70*(2), 503–529.
18. Olusegun, S. **(2015)**. Constructivism learning theory: A paradigm for teaching and learning. *Journal of Research & Method in Education*, *5*(6), 66–70.
19. Stapleton, L., & Stefaniak, J. **(2019)**. Cognitive Constructivism: Revisiting Jerome Bruner's Influence on Instructional Design Practices. *TechTrends*, *63*(1), 4–5.
20. Braithwaite, D. W., & Sprague, L. **(2021)**. Conceptual Knowledge, Procedural Knowledge, and Metacognition in Routine and Nonroutine Problem Solving. *Cognitive Science*, *45*(10**)**.
21. Li, C., He, J., Yuan, C., Chen, B., & Sun, Z. **(2019)**. The effects of blended learning on knowledge, skills, and satisfaction in nursing students: A meta-analysis. *Nurse Education Today*, *82*, 51–57.
22. Weese, J. L., & Feldhausen, R. **(2017)**. STEM outreach: Assessing computational thinking and problem solving. *ASEE Annual Conference and Exposition, Conference Proceedings*, *2017-June*.
23. Uziak, J. **(2016)**. A project-based learning approach in an engineering curriculum. *Global Journal of Engineering Education*, *18*(2), 119–123.
24. Chang, S. C., & Wongwatkit, C. **(2024)**. Effects of a peer assessment-based scrum project learning system on computer programming's learning motivation, collaboration, communication, critical thinking, and cognitive load. *Education and Information Technologies*, *29*(6), 7105–7128.
25. Kokotsaki, D., Menzies, V., & Wiggins, A. **(2016)**. Project-based learning: A review of the literature. *Improving Schools*, *19*(3), 267–277.
26. Balakrishnan, B., & Long, C. Y. **(2020)**. An Effective Self-directed Personalized Learning Environment for Engineering Students During the COVID-19 Pandemic. *Advances in Engineering Education*, *8*(4), 1–8.
27. Choi, I., & Jie, F. Z. **(2021)**. "Time to Be an Academic Influencer": Peer-to-Peer Learning Enhances Students' Self-Directed Learning with Disparate Knowledge Background in CAD. *Cubic Journal*, *4*, 54–69.
28. Knowles, M. S. **(1975)**. *Self-directed learning: A guide for learners and teachers*. Association Press.
29. Merriam, S. B., & Baumgartner, L. M. **(2006)**. *Learning in adulthood: A comprehensive guide* (4th ed.**)**. Jossey-Bass.
30. Graham, C. R. **(2013)**. Emerging Practice and Research in Blended Learning. In *Handbook of Distance Education* (Vol. 3, pp. 333–350**)**. Routledge.
31. Herlambang, A. D., Ririn, R., & Rachmadi, A. **(2024)**. The flipped-classroom effect on vocational high school students' learning outcomes. *International Journal of Evaluation and Research in Education (IJERE)*, *13*(3), 1807.
32. Herlambang, A. D., & Rachmadi, A. **(2023)**. The Online Learning Interest and Learning Outcomes Through Mobile and Desktop Application Based on the Indonesian Information Technology Majoring Vocational High School Student's Perspective. *ACM International Conference Proceeding Series*, 354–360.
33. Vo, H. M., Zhu, C., & Diep, N. A. **(2013)**. The effect of blended learning on student performance at course-level in higher education: A meta-analysis. *Studies in Educational Evaluation*, *53*, 17–28.
34. Guzdial, M. **(2015)**. Learner-Centered Design of Computing Education: Research on Computing for Everyone. *Synthesis Lectures on Human-Centered Informatics*, *8*(6), 1–165.
35. Flórez, F. B., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. **(2017)**. Changing a Generation's Way of Thinking: Teaching Computational Thinking Through Programming.

36. Wu, B., Hu, Y., Ruis, A. R., & Wang, M. **(2019)**. Analysing computational thinking in collaborative programming: A quantitative ethnography approach. *Journal of Computer Assisted Learning*, *35*(3), 421–434. https://doi.org/10.1111/jcal.12348
37. Sweller, J. **(1988)**. Cognitive Load During Problem Solving: Effects on Learning. *Cognitive Science*, *12*(2), 257–285.
38. Creswell, J. W., & Clark, V. L. P. **(2017)**. *Designing and conducting mixed methods research* (4th ed.**)**. SAGE Publications.
39. Fraenkel, J. R., Wallen, N. E., & Hyun, H. H. **(2022)**. *How to Design and Evaluate Research in Education*. McGraw Hill LLC.
40. Cohen, J. **(1988)**. *Statistical Power Analysis for the Behavioral Sciences* (2nd ed.**)**. Lawrence Erlbaum Associates.
41. Landis, J. R., & Koch, G. G. **(1977)**. The Measurement of Observer Agreement for Categorical Data. *Biometrics*, *33*(1), 159.
42. Braun, V., & Clarke, V. **(2006)**. Using thematic analysis in psychology. *Qualitative Research in Psychology*, *3*(2), 77–101.
43. Jang, H., Kim, E. J., & Reeve, J. **(2016)**. Why students become more engaged or more disengaged during the semester: A self-determination theory dual-process model. *Learning and Instruction*, *43*, 27–38.
44. Ryan, R. M., & Deci, E. L. **(2000)**. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American Psychologist*, *55*(1), 68–78.
45. Sergis, S., Sampson, D. G., & Pelliccione, L. **(2018)**. Investigating the impact of Flipped Classroom on students' learning experiences: A Self-Determination Theory approach. *Computers in Human Behavior*, *78*, 368–378.
46. Farnsworth, V., Kleanthous, I., & Wenger-Trayner, E. **(2016)**. Communities of Practice as a Social Theory of Learning: a Conversation with Etienne Wenger. *British Journal of Educational Studies*, *64*(2), 139–160.
47. Anthonysamy, L., Koo, A. C., & Hew, S. H. **(2020)**. Self-regulated learning strategies and non-academic outcomes in higher education blended learning environments: A one decade review. *Education and Information Technologies*, *25*(5), 3677–3704.
48. Halverson, L. R., & Graham, C. R. **(2019)**. Learner engagement in blended learning environments: A conceptual framework. *Online Learning Journal*, *23*(2), 145–178.
49. Zimmerman, B. J. **(2002)**. Becoming a Self-Regulated Learner: An Overview. *Theory Into Practice*, *41*(2), 64–70.
50. Piaget, J. **(1976)**. Piaget's Theory. In B. Inhelder, H. H. Chipman, & C. Zwingmann (Eds.), *Piaget and His School* (pp. 11–23**)**. Springer.
51. Vygotsky, L. S. **(1978)**. Social Constructivism - Mind in Society: The Development of Higher Psychological Processes. In *Full-Text. (N.D.)*. Harvard University Press.
52. Duran, R., Zavgorodniaia, A., & Sorva, J. **(2022)**. Cognitive Load Theory in Computing Education Research: A Review. *ACM Transactions on Computing Education*, *22*(4**)**.
53. Kirschner, P. A., Sweller, J., Kirschner, F., & Zambrano, J. R. **(2018)**. From Cognitive Load Theory to Collaborative Cognitive Load Theory. *International Journal of Computer-Supported Collaborative Learning*, *13*(2), 213–233.
54. Leppink, J. **(2017)**. Cognitive load theory: Practical implications and an important challenge. *Journal of Taibah University Medical Sciences*, *12*(5), 385–391.
55. Lai, R. P. Y. **(2022)**. Beyond Programming: A Computer-Based Assessment of Computational Thinking Competency. *ACM Transactions on Computing Education*, *22*(2), 27 Pages.
56. Raj, N. S., & Renumol, V. G. **(2022)**. A systematic literature review on adaptive content recommenders in personalized learning environments from 2015 to 2020. *Journal of Computers in Education*, *9*(1), 113–148.
57. Herlambang, A. D., & Rachmadi, A. **(2024)**. Student's Perception of Technology-Rich Classrooms Usage to Support Conceptual and Procedural Knowledge Delivery in Higher Education Computer Science Course. *Procedia Computer Science*, *234*, 1500–1509.