International Journal of Instruction
e-ISSN: 1308-1470 ● www.e-iji.net

April 2025 ● Vol.18, No.2
p-ISSN: 1694-609X
pp. 245-260

Article submission code: 20240826071110

Received: 26/08/2024
Revision: 17/11/2024

Accepted: 25/11/2024
OnlineFirst: 03/01/2025

Assessing Instructors' Perceptions of Critical Skills in Computational Thinking and Block-Based Programming: A Needs Assessment Approach

Arhit Aroonsiwagool
King Mongkut's Institute of Technology Ladkrabang, Thailand, *63603116@kmitl.ac.th*

Somkiat Tuntiwongwanich
Corresponding author, King Mongkut's Institute of Technology Ladkrabang, Thailand, *somkiat.tu@kmitl.ac.th*

Paitoon Pimdee
King Mongkut's Institute of Technology Ladkrabang, Thailand, *paitoon.pi@kmitl.ac.th*

Chontawat Meedee
King Mongkut's Institute of Technology Ladkrabang, Thailand, *chontawat.me@kmitl.ac.th*

Sangutai Moto
King Mongkut's Institute of Technology Ladkrabang, Thailand, *sangutai.mo@kmitl.ac.th*

This study investigates the discrepancy between instructors' perceptions of students' computational thinking (CT) and block-based programming (BBP) skills and the actual competence of the students. The sample comprised 47 instructors from five Rajabhat Universities in Southern Thailand, teaching programming-related courses at undergraduate and graduate levels during the 2023 academic year. Data were collected via online questionnaires, with reliability scores of 0.89 and 0.88 for the CT skills assessment, and 0.77 and 0.76 for the BBP skills assessment, for both desired (I) and actual (D) states. Analysis using means, standard deviations, the modified Priority Needs Index (PNImodified), and t-tests for dependent groups revealed significant gaps between desired and actual skill levels. The most critical needs were identified in algorithm design and decomposition for CT skills, and function writing for BBP skills. These results highlight the need for targeted interventions to address the discrepancy between instructors' expectations and students' abilities.

Keywords: block-based programming, computational thinking skills, needs assessment, programming instruction, Thailand

INTRODUCTION

In today's era of complex technology, effective decision-making, and systematic problem-solving are essential (Mohaghegh & Furlan, 2020). This has led to a growing emphasis on developing critical thinking skills to address intricate challenges, particularly in computational thinking (Su & Yang, 2023; Sukkamart et al., 2024). Computational thinking, coupled with block-based programming (BBP), has gained popularity as a method to enhance cognitive abilities due to its effectiveness in fostering structured problem-solving skills (Cheng et al., 2023; Roungrong et al., 2018).

While research on developing computational thinking and block-based programming (BBP) skills exists (Chen & Chung, 2024), this study uniquely contributes by reporting on instructors' assessments of students' current competence levels in these areas using standardized evaluation tools. Some areas may not require development or intervention, as they do not pose significant challenges. Research on programming skills often highlights computational thinking as the core of developing programming algorithms (Wing, 2006). Currently, BBP is widely used to teach algorithm development, with many countries promoting its use through game-based learning (Broza et al., 2023).

A review of the academic performance of students in the Computer Education program at Nakhon Si Thammarat Rajabhat University, following the 2022 academic year (Thongkum et al., 2023), revealed suboptimal results in advanced programming courses (Sharov et al., 2023). Despite efforts by instructors to improve teaching methods and incorporate various tools, students struggled with programming tasks. Interviews with instructors indicated that the students lacked computational thinking skills, which hindered their ability to design algorithms and solve programming problems (Kite & Park, 2024). This deficiency also led to a lack of confidence in programming (Amnouychokanant et al., 2021), particularly in traditional coding, and issues with syntax correctness.

Needs assessment is a valuable tool for analyzing and identifying problems and prioritizing solutions. This process aids instructors in designing curricula, teaching methods, and content that align with students' needs (Tobua et al., 2018). It helps in identifying the actual issues and facilitates the design of targeted interventions that meet the specific requirements of computational thinking. Computational thinking involves systematic, step-by-step problem-solving using efficient algorithms to achieve the best outcomes (Roungrong et al., 2018). This skill, which is fundamental to computer science, is included in the curricula of many countries and is crucial for solving today's complex problems. Without effective thinking strategies, problem-solving becomes challenging.

Developing computational thinking can be difficult if the specific areas needing improvement are not identified. Inappropriate content or tools can lead to ineffective problem-solving. Therefore, understanding the precise issues allows for designing content and tools tailored to address specific challenges in advanced programming courses.

Block-based programming, which involves dragging and dropping blocks to design algorithms, is suitable for beginners without programming experience. It alleviates concerns about syntax errors, making it an ideal tool for algorithm development. Currently, BBP is made more engaging through game-based learning, unplugged activities, board games, web-based learning, and creative projects. Tools are being developed to make it easier for learners to enhance their computational thinking and programming skills (Kalelioğlu, 2015; Rich et al., 2019). Additionally, most BBP environments are designed as games to attract learners, making it a popular method for developing computational thinking and programming skills (Carlborg et al., 2019).

The challenges identified in this research highlight that promoting CT skills is difficult without understanding the specific needs and urgent areas for improvement. This aligns with the promotion of BBP skills, which is an effective tool for developing programming algorithms. However, the content of programming courses varies in complexity. If instructors do not design content that meets students' needs, the promotion of programming skills may not be effective.

Statement of the Problem/Research Gap

In the current era of rapid technological advancement, the ability to think computationally and understand programming concepts has become a critical skill set, particularly within the field of computer science education. Computational thinking, which involves problem-solving using systematic approaches such as decomposition, pattern recognition, abstraction, and algorithm design, is increasingly recognized as essential for students to navigate and excel in a technology-driven world. Block-based programming, a method that simplifies coding through the use of visual blocks representing code structures, has emerged as an effective tool for teaching programming, especially to beginners. By reducing the cognitive load associated with syntax, BBP allows students to focus on the logical flow and structure of algorithms, thereby enhancing their computational thinking skills.

Despite the recognized importance of these skills, there remains a significant gap in the literature regarding the effective development and assessment of computational thinking and BBP skills among students. While numerous studies have explored general approaches to teaching these concepts, few have specifically addressed the distinct needs and challenges faced by students in developing these skills. Existing research often fails to consider the specific areas within CT, such as algorithm design and decomposition, that are critical yet challenging for students to master. Moreover, there is a lack of comprehensive needs assessments that identify the specific gaps between students' current capabilities and the desired proficiency levels in these areas.

This study seeks to address these gaps by conducting a detailed needs assessment of CT and BBP skills among students. By comparing the actual skill levels of students with the levels that are deemed necessary for success in computer science, this research aims to identify the key areas where students struggle and to provide actionable insights for educators. The findings will not only contribute to the existing body of knowledge but also offer practical guidance for curriculum development and instructional strategies, ensuring that students are better equipped to meet the demands of the modern technological landscape.

Research Hypothesis

This study hypothesizes that key components of computational thinking, such as algorithm design and decomposition, as well as BBP skills like function writing, are critical for students in computer science education. Furthermore, it is expected that the current levels of these skills among students are significantly lower than the desired levels, indicating a need for targeted educational interventions to better develop these competencies within the context of Thailand's educational system.

Research Objectives

RO1: To study the opinions of instructors regarding students' CT and BBP skills.
RO2: To analyze and rank the needs related to CR and BBP skills, comparing descriptive and inferential statistical methods.

Research Questions

RQ1: What is the current level of CT skills among students as perceived by instructors?
RQ2: What is the current level of BBP skills among students as perceived by instructors?
RQ3: How do the actual levels of CT and BBP skills compare to the desired levels needed for success in computer science education?
RQ4: Which specific areas within CT and BBP require the most significant development based on the Priority Needs Index ($PNI_{modified}$)?

LITERATURE REVIEW

Needs Assessment

Needs assessment is a critical tool for analyzing the gap between the current state and the desired state, often used in the planning phase before implementing problem-solving strategies. PNI originated with the United Nations Development Programme (UNDP) in the 1990s as a tool designed to identify and prioritize the needs of vulnerable populations in developing countries. Initially developed to support the allocation of resources in development projects, the Priority Needs Index (PNI) was instrumental in guiding efforts to address critical areas of need in these regions.

The methodology introduced by the UNDP was later adapted for use in Thailand, where it gained significant traction. Wongwanich and Wiratchai (2005) played a pivotal role in this adaptation, refining the PNI to suit the specific needs and context of Thailand better. Although they were not the original creators of the PNI, their contributions were crucial in modifying the tool to enhance its applicability in Thai development planning. Since their work, the modified PNI ($PNI_{modified}$)has become widely used in both Thailand and Asia for identifying priority areas for development and guiding resource allocation decisions. It remains an important tool in the country's development planning framework, helping to ensure that resources are directed to where they are most needed.

$PNI_{modified}$ has become a useful tool in the identification of the precise processes or methods required to address the needs of stakeholders (Wongwanich, 2019). This approach is widely employed to enhance operational efficiency by pinpointing areas that require improvement and development (Mirsa, 2024; Prasittichok & Klaykaew, 2022; Yurayat & Seechaliao, 2021). For instance, Riese et al. (2023) utilized needs assessment to enhance academic services in libraries for graduate students, aligning the services with the needs of both students and faculty. The research findings enabled the library to improve its offerings, thereby supporting the research efforts of graduate students and faculty members. Tobua et al. (2018) also applied needs assessment to design professional development programs for vocational educators, specifically focusing on enhancing their project evaluation skills.

Suadang et al. (2020) used a comprehensive needs assessment to develop distance learning materials for undergraduate students, aiming to identify and address the essential needs in communication and media for distance education. Wuttikamonchai et al. (2024) applied needs assessment in evaluating students' skills in mobile web development. The findings highlighted the critical skills that industry professionals and educators prioritized, enabling a focused approach to skill development. Similarly, Ussarn et al. (2022) used needs assessment to promote digital literacy in community colleges, identifying the specific areas where students needed improvement.

Computational Thinking

Computational thinking, a concept that has gained widespread attention since the early 2000s, was first introduced by Jeannette M. Wing (Wing, 2006). She defined it as a problem-solving process that involves breaking down complex issues into manageable parts and abstract thinking. Computational thinking is not limited to computer scientists; it can be applied to everyday problem-solving by anyone. It is considered a crucial skill for the 21st century (Hou et al., 2020) because it involves systematic, step-by-step thinking to arrive at efficient solutions. Computational thinking is an advanced cognitive process that generates innovative solutions (Sittikhetkron & Sawangmek, 2021).

Songkhram et al. (2020) echoed these views, emphasizing that computational thinking involves using skills and techniques to solve problems systematically. It is a structured and efficient thought process that includes four key components: 1) decomposition, 2) pattern recognition, 3) abstraction, and 4) algorithm design (Rueangrong et al., 2018; Rueangrong & Phitthayasenee, 2021; Sittikhetkron & Sawangmek, 2021; Özmutlu et al., 2021). Although the specific steps may vary depending on the definition, the core approach remains consistent—systematic thinking to solve problems efficiently.

Cheng et al. (2023) focused on enhancing students' computational thinking skills through self-generated questioning strategies on a game-based learning platform. This approach aimed to develop higher-order thinking skills by encouraging students to decompose problems and engage in abstract thinking during the learning process. The study found that the questioning strategy effectively promoted advanced problem-solving skills, motivation, and confidence in students. Broza et al. (2023) also contributed to this field by designing a curriculum that uses BBP to develop CT skills in teacher education students.

Block-Based Programming (BBP)

Block-based programming is an accessible tool for programming, especially suitable for those developing algorithms without worrying about syntax errors (Rich et al., 2019; Kalelioğlu, 2015). This method involves selecting and connecting blocks of code through drag-and-drop actions to solve programming problems by designing algorithms.

Block-based programming is particularly helpful for novice programmers or those who want to develop algorithms, as it allows for the creation of both simple and complex algorithms.

Currently, BBP is increasingly integrated with educational theories to make learning programs more enjoyable. For example, Chowdhury et al. (2024) utilized BBP to develop a digital game-based English vocabulary game, combining game-based learning with constructivist theory. This research aimed to have students create the vocabulary game themselves, thereby enhancing their CT and algorithmic skills at the elementary level. Similarly, Uğraş et al. (2022) used BBP to develop collaborative game design techniques for students and teachers, leveraging children's innate creativity and interests to stimulate their imagination. Videnovik et al. (2024) researched game-based learning approaches in computer science education for elementary students and determined that most games run on web platforms using BBP, where students learn while playing through the drag-and-drop programming approach.

METHOD

Research Design

This study employs a descriptive survey design combined with a needs assessment approach. The descriptive survey aims to gather data on instructors' perceptions of their students' computational thinking (CT) and block-based programming (BBP) skills, while the needs assessment helps identify the gap between the actual skill levels and the desired proficiency. This mixed-method approach is appropriate for understanding both the current state and the priority areas for intervention, offering practical guidance for improving curriculum and instruction.

Population and Sample

The population for this study consisted of instructors teaching programming or related courses at undergraduate and graduate levels in computer science or related fields during the 2023 academic year. Five Rajabhat Universities in Southern Thailand were targeted, encompassing a total population of 139 instructors. Using Yamane's formula with a 10% margin of error, a target sample size of 60 participants was calculated (Mirsa, 2024). Simple random sampling was used to ensure a representative sample (Table 1).

Table 1
Population and sample of instructors from Rajabhat Universities in Southern Thailand

Rajabhat University	Population (Total Instructors)	Sample Group		
		Target	Collected	
			Collected	%
Nakhon Si Thammarat	40	17	13	77
Surat Thani	22	9	9	100
Songkhla	23	10	8	80
Yala	28	12	9	75
Phuket	26	11	8	73
Total	139	60	47	78

Research Instruments

Two primary tools were employed to assess both the actual and expected levels of CT and BBP skills. Each used a 5-point Likert scale and was validated for content consistency by five experts:

1. Computational Thinking Skills Assessment: This tool evaluated four key aspects: decomposition, pattern recognition, abstraction, and algorithm design, based on established frameworks (Palts & Pedaste, 2020; Pewkam & Chamrat, 2022). The tool consisted of 24 items, with content validity tested using the Index of Item Objective Congruence (IOC), yielding scores from 0.80 to 1.00. Pilot testing was conducted with 30 instructors, and reliability coefficients are presented in Table 2.

2. Block-Based Programming Skills Assessment: This tool focused on three core aspects: conditional logic, iteration (looping), and function writing, all critical to programming education. The IOC ranged from 0.80 to 1.00, and reliability testing was conducted using the same pilot group. The reliability results are detailed in Table 2.

Table 2
IOC and reliability for CT and BBP skills assessment

Skill/Aspect	Items	IOC	Reliability (α)	
			Actual condition	Expected Condition
Computational Thinking Skills	24	-	0.89	0.88
Decomposition	6	1.00	0.85	0.83
Pattern Recognition	6	1.00	0.86	0.83
Abstraction	6	0.80-1.00	0.86	0.83
Algorithms	6	1.00	0.85	0.83
Block Programming Skills	17	-	0.77	0.76
Conditional Programming	5	0.80-1.00	0.81	0.79
Iterative Programming	6	0.80-1.00	0.77	0.76
Functional Programming	6	1.00	0.83	0.78

Data Collection

Data were collected through questionnaires distributed to instructors teaching programming or related courses at the undergraduate or graduate levels in computer science or related fields. The data collection occurred in August 2023 and targeted 60 participants from the five Rajabhat Universities in Southern Thailand. The questionnaires were administered online via Google Forms, with 47 responses received, representing a 78% response rate (Table 1).

Data Analysis

The analysis of instructors' opinions on CT and BBP skills was conducted using descriptive statistics, including means, standard deviations (SD), and percentages. A 5-point Likert scale was used to gauge agreement on each item, with the scale levels interpreted as follows: 5 = very high (4.50-5.00), 4 = high (3.50-4.49), 3 = moderate (2.50-3.49), 2 = low (1.50-2.49), and 1 = very low (1.00-1.49) (Mirsa, 2024).

The needs assessment for CT and BBP was undertaken using the Priority Needs Index ($PNI_{modified}$) method (Nguyen & Leksansern, 2024). The formula for the modified PNI is:

$$PNI_{modified} = (I - D)/ D \quad (1)$$

PNI = priority needs index

I = Mean for the *intended* or *desired* outcome

D = Mean for the *actual* results or success

Research hypothesis testing was performed to compare the overall and specific aspects between the actual condition (D) and the intended or desired condition (I) using a t-test for dependent groups. This test determined whether there was a significant difference between the desired outcomes and the actual results.

FINDINGS

Analysis of Computational Thinking Skills

The analysis of computational thinking (CT) skills, as presented in Table 4, reveals significant gaps between the desired (I) and actual (D) skill levels among students. Instructors perceive students' desired CT skills to be at a high level, but their actual skills are only moderate. This discrepancy underscores the need for targeted educational interventions, particularly in algorithm design and decomposition.

Table 4
CT skills need assessment

CT Skills	State	Mean	Level	SD	$PNI_{modified}$		t-test		
					(I-D)/D	Rank	t	Sig.	Rank
Decomposition	Desired (I)	4.28	High	0.66	0.31	2	8.65**	<.00	2
	Actual (D)	3.27	Moderate	0.66					
Pattern Recognition	Desired (I)	4.26	High	0.72	0.21	3	5.77**	<.00	4
	Actual (D)	3.53	High	0.62					
Abstraction	Desired (I)	4.41	High	0.61	0.20	4	6.98**	<.00	3
	Actual (D)	3.67	High	0.54					
Algorithm Design	Desired (I)	4.42	High	0.62	0.33	1	9.95**	<.00	1
	Actual (D)	3.32	Moderate	0.65					
Overall	Desired (I)	4.34	High	0.62	0.25	-	8.93**	<.00	-
	Actual (D)	3.45	Moderate	0.51					

Note: $p < 0.05$, significant

Algorithm design emerged as the most critical skill needing development, with a $PNI_{modified}$ score of 0.33 and the highest t-test value ($t = 9.95$, $p < 0.01$). This finding is crucial because algorithm design is foundational to computational thinking, directly impacting students' ability to solve complex problems systematically (Clarke-Midura et al., 2023; Stephens & Kadijevich, 2020). The significant gap between the desired and actual levels suggests that students may struggle with understanding and applying algorithms in their programming tasks, aligning with previous studies that emphasize the importance of algorithm design in higher-order problem-solving (Sittikhetkron & Sawangmek, 2021).

Decomposition, which breaks down complex problems into smaller, manageable parts, ranked second in terms of need, with a PNI$_{modified}$ score of 0.31 and a significant t-test value (t = 8.65, p < 0.01). Decomposition is fundamental in developing efficient solutions, but the moderate level of students' actual skills suggests that while students can identify problems, they may struggle with effectively breaking them down into smaller tasks.

Pattern recognition and abstraction also showed significant gaps between the desired and actual states, but their PNI$_{modified}$ scores were slightly lower, indicating these areas, while important, may not be as urgent as algorithm design and decomposition. The results suggest that targeted interventions focusing on these critical skills could significantly enhance students' computational thinking abilities, better equipping them to tackle complex programming challenges (Sittisak et al., 2022).

Analysis of Block-Based Programming Skills

Table 5 provides insights into students' block-based programming (BBP) skills, with instructors rating the desired level (I) at the highest level, while the actual skills (D) were rated as high but still showed noticeable gaps.

Function writing was identified as the top priority for improvement, with a PNImodified score of 0.21 and the highest t-test value (t = 9.90, p < 0.01). Function writing is crucial in programming as it enables code modularization, making programs more efficient and easier to manage. This finding suggests that while students are familiar with BBP, they struggle with advanced concepts such as function writing, consistent with the work of Rich et al. (2019), who noted the need for careful scaffolding to help students progress to more complex tasks.

Table 5
BBP skills need assessment

BBP Skills	State	Mean	Level	SD.	Index of essential needs		Hypothesis testing		
					(I-D)/D	Rank	t	Sig.	Rank
Conditional Programming	Desired (I)	4.66	Very High	0.39	0.16	2	7.61 **	<.00	3
	Actual (D)	4.02	High	0.43					
Iterative Programming	Desired (I)	4.73	Very High	0.39	0.14	3	9.08 **	<.00	2
	Actual (D)	4.16	High	0.37					
Functional Programming	Desired (I)	4.77	Very High	0.36	0.21	1	9.90 **	<.00	1
	Actual (D)	3.95	High	0.42					
Overall	Desired (I)	4.72	Very High	0.34	0.17	-	10.44 **	<.00	-
	Actual (D)	4.04	High	0.32					

Note: $p < 0.05$, significant

Conditional programming and looping were also identified as important areas but with slightly lower PNImodified scores, indicating these skills are relatively better developed but still require attention. The gaps in these skills suggest that students may not fully understand how to apply these concepts in more complex programming scenarios. This aligns with Kalelioğlu (2015), who highlighted the importance of these skills in building a strong foundation for programming, particularly when transitioning from block-based to text-based coding.

Overall, the study's findings suggest that while BBP is effective in developing foundational programming skills, there is a critical need for more advanced instruction to help students move beyond the basics. Integrating more complex programming tasks into the curriculum and providing opportunities for students to practice these skills in varied contexts, such as game-based learning environments, could enhance engagement and learning outcomes (Videnovik et al., 2024).

DISCUSSION

The findings of this study reveal notable discrepancies between instructors' perceptions of the desired and actual computational thinking (CT) skills among students. Specifically, instructors rated students' desired CT skills at a high level (mean = 4.34), but their actual skills were moderate (mean = 3.45). This gap underscores the need for more targeted educational interventions to bridge the divide, particularly in algorithm design and decomposition—two key areas where the discrepancies were most pronounced.

Instructor Perception of CT Skill Gaps

The finding that instructors perceive students' desired CT skills to be high while their actual skills remain moderate aligns with similar challenges identified in previous research. For instance, Stephens and Kadijevich (2020) noted that while students often demonstrate some understanding of computational principles, applying these principles to more complex tasks—especially in algorithm design—remains a significant hurdle. This gap suggests that current teaching methods may not adequately prepare students for real-world problem-solving, where algorithm design is crucial for breaking down complex problems into logical sequences.

In terms of teacher assessment, this discrepancy could be partially attributed to the types of rubrics used to evaluate CT skills. Traditional rubrics may focus more on conceptual understanding rather than practical application, as Clarke-Midura et al. (2023) highlighted the importance of aligning assessment tools with authentic problem-solving tasks. This calls for rubrics that not only evaluate students' theoretical knowledge but also their ability to apply CT principles to realistic programming challenges, especially in algorithm design and decomposition.

Implications for Educational Interventions

To address these gaps, several **targeted interventions** can be implemented:

1. Algorithm Design Workshops: Given the high priority placed on algorithm design (PNImodified = 0.33), focused workshops that emphasize this skill would be beneficial. These workshops should use hands-on activities, such as coding challenges that require students to break down problems and design algorithms. Research suggests that

scaffolding students through increasingly complex problems can significantly improve algorithmic thinking (Sittikhetkron & Sawangmek, 2021).

2. Decomposition Practice through Project-Based Learning: Since decomposition was also identified as a key area for improvement (PNImodified = 0.31), project-based learning (PBL) could be an effective intervention. In PBL, students are tasked with larger projects that they must decompose into smaller tasks, reinforcing the decomposition process. Projects could range from building simple applications to more complex systems, helping students practice breaking down problems in a structured manner, as recommended by Sittisak et al. (2022).

3. Gamified Learning Environments: To boost engagement and enhance learning outcomes in both CT and BBP, incorporating gamified environments, such as game-based learning (GBL), can be an effective strategy. Videnovik et al. (2024) emphasized that GBL, where students are tasked with solving puzzles or programming games, can improve both algorithmic thinking and decomposition by immersing students in challenging, yet enjoyable, tasks that require systematic problem-solving.

Block-Based Programming Skill Gaps

In the context of block-based programming (BBP), the most significant gap was in functional programming ($PNI_{modified}$ = 0.21), where students struggled with writing reusable code blocks. This finding mirrors the challenges noted by Rich et al. (2019), who emphasized the importance of providing structured guidance to help students grasp complex programming concepts like functions. The difficulty students face with function writing is particularly concerning because this skill is essential for creating modular and efficient code—a necessity for more advanced programming tasks.

Targeted BBP Interventions

1. Function Writing Instruction: To address the gap in function writing, instructors could implement step-by-step tutorials that walk students through the process of creating, testing, and refining functions. As noted by Kalelioğlu (2015), scaffolding plays a crucial role in enabling students to move from simple blocks to more complex programming structures. Integrating more function-based tasks into coursework can gradually increase students' comfort and competence with this advanced skill.

2. Advanced Block-Based Challenges: Introducing more advanced block-based challenges that require the use of loops and conditionals in conjunction with functions would help students develop these critical skills. Such tasks should encourage students to apply conditional programming and iteration in more complex scenarios, helping them transition from basic programming to more sophisticated designs.

Broader Implications for Curriculum Development

The overall findings suggest a broader need to revise the current curriculum to incorporate more advanced computational thinking and programming skills, particularly in higher-order problem-solving. Instructors can play a key role in identifying and addressing specific areas where students struggle, using data-driven approaches such as the Priority Needs Index ($PNI_{modified}$) to pinpoint the most pressing gaps.

Moreover, the integration of real-world problems into programming tasks can further enhance students' ability to apply CT and BBP skills in diverse contexts. Encouraging students to tackle open-ended problems that mimic real-world scenarios, such as app development or data analysis projects, could provide them with valuable experience that traditional exercises may not offer. This aligns with the recommendations of Clarke-Midura et al. (2023), who advocate for the use of authentic tasks in programming education.

CONCLUSION

The findings of this study reveal significant discrepancies between instructors' perceptions of students' desired CT and BBP skills and the actual competence demonstrated by students. Instructors rated the desired levels of algorithm design, decomposition, and function writing skills at a high level, while students' actual performance in these areas was moderate, highlighting the need for focused interventions. These gaps underscore the critical importance of refining educational strategies to better align student competence with instructors' expectations.

To bridge this gap, targeted curriculum adjustments are essential. Prioritizing the development of algorithm design, decomposition, and function writing skills will not only enhance students' ability to engage in higher-order computational tasks but also ensure they are better prepared to navigate complex programming challenges. The integration of workshops, project-based learning, and gamified environments could serve as effective interventions, helping students to apply computational thinking more effectively in real-world contexts. Addressing this discrepancy is vital to fostering the next generation of proficient computer science professionals.

LIMITATIONS

This study, while providing valuable insights into the need for developing CT and BBP skills, has some limitations. Firstly, the sample size was limited to instructors from five Rajabhat Universities in Southern Thailand, which may not fully represent the broader educational scope across Thailand or other regions. Additionally, the study relied on self-reported data from instructors, which may introduce bias or inaccuracies in assessing students' actual skill levels. The cross-sectional nature of the study also limits the ability to conclude changes in skills over time.

SUGGESTIONS

Future research could expand on this study by including a wider range of educational institutions across different regions, providing a more comprehensive view of computational thinking and programming skills across Thailand. Additionally, incorporating direct assessments of students' skills, rather than relying solely on instructors' perceptions, would offer a more objective measure of students' abilities. Researchers could also explore the effectiveness of specific instructional strategies or interventions designed to enhance these skills, particularly focusing on areas identified as having the greatest need, such as algorithm design and function writing. Finally, longitudinal studies tracking students' progress over time would provide deeper insights into how these skills develop and the long-term impact of targeted educational interventions.

REFERENCES

Amnouychokanant, V., Boonlue, S., Chuathong, S., & Thamwipat, K. (2021). A Study of First-Year Students' Attitudes toward Programming in the Innovation in Educational Technology Course. *Education Research International*, *2021*(1), 9105342. https://doi.org/10.1155/2021/9105342

Broza, O., Biberman-Shalev, L., & Chamo, N. (2023). "Start from scratch": Integrating computational thinking skills in teacher education program. *Thinking Skills and Creativity, 48,* 101285. https://doi.org/https://doi.org/10.1016/j.tsc.2023.101285

Carlborg, N., Tyrén, M., Heath, C., & Eriksson, E. (2019). The scope of autonomy when teaching computational thinking in primary school. *International Journal of Child-Computer Interaction*, *21*, 130-139. https://doi.org/https://doi.org/10.1016/j.ijcci.2019.06.005

Chen, C. H., & Chung, H. Y. (2024). Fostering computational thinking and problem-solving in programming: Integrating Concept maps into Robot Block-based programming. *Journal of Educational Computing Research*, *62*(1), 406-427. https://doi.org/10.1177/07356331231205052

Cheng, Y.-P., Lai, C.-F., Chen, Y.-T., Wang, W.-S., Huang, Y.-M., & Wu, T.-T. (2023). Enhancing student's computational thinking skills with student-generated questions strategy in a game-based learning platform. *Computers & Education*, *200*, 104794. https://doi.org/https://doi.org/10.1016/j.compedu.2023.104794

Chowdhury, M., Dixon, L. Q., Kuo, L.-J., Donaldson, J. P., Eslami, Z., Viruru, R., & Luo, W. (2024). Digital game-based language learning for vocabulary development. *Computers and Education Open*, *6*, 100160. https://doi.org/https://doi.org/10.1016/j.caeo.2024.100160

Clarke-Midura, J., Silvis, D., Shumway, J. F., Lee, V. R., & Kozlowski, J. S. (2023). Developing a kindergarten computational thinking assessment using evidence-centered design: the case of algorithmic thinking. In *Assessing Computational Thinking* (pp. 5-28). Routledge. https://doi.org/10.4324/9781003431152

Hou, H. Y., Agrawal, S., & Lee, C. F. (2020). Computational thinking training with technology for non-information undergraduates. *Thinking Skills and Creativity*, *38*, 100720. https://doi.org/10.1016/j.tsc.2020.100720

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code.org. *Computers in Human Behavior, 52*, 200-210. https://doi.org/https://doi.org/10.1016/j.chb.2015.05.047

Kite, V., & Park, S. (2024). Context matters: Secondary science teachers' integration of process-based, unplugged computational thinking into science curriculum. *Journal of Research in Science Teaching*, *61*(1), 203-227. https://doi.org/10.1002/tea.21883

Mirsa, N. R. P. (2024). Needs Assessment for Professional Competence of Elementary School Teacher in Yogyakarta. *IJCAR: Indonesian Journal of Classroom Action Research*, *2*(2), 26-33. https://doi.org/10.53866/ijcar.v2i2.535

Mohaghegh, M., & Furlan, A. (2020). Systematic problem-solving and its antecedents: a synthesis of the literature. *Management Research Review*, *43*(9), 1033-1062. https://doi.org/10.1108/MRR-06-2019-0284

Nguyen, T. V. D., & Leksansern, A. (2024). A model of continuing professional development for Vietnamese lecturers. *Kasetsart Journal of Social Sciences*, *45*(1), 147-158. https://doi.org/10.34044/j.kjss.2024.45.1.16

Özmutlu, M., Atay, D., & Erdp an, B. h. (2021). Collaboration and engagement-based coding training to enhance children's computational thinking self-efficacy. *Thinking Skills and Creativity*, 100833.

Palts, T., & Pedaste, M. (2020). A model for developing computational thinking skills. *Informatics in Education*, *19*(1), 113-128.

Pewkam, W., & Chamrat, S. (2022). Pre-service teacher training program of STEM-based activities in computing science to develop computational thinking. *Informatics in Education*, *21*(2), 311-329.

Prasittichok, P., & Klaykaew, K. K. (2022). Meta-skills development needs assessment among undergraduate students. *Heliyon, 8* (1), 1–5. https://doi.org/10.1016/j.heliyon.2022.e08787

Rich, P. J., Browning, S. F., Perkins, M., Shoop, T., Yoshikawa, E., & Belikov, O. M. (2019) . Coding in K-8: International Trends in Teaching Elementary/ Primary Computing. *TechTrends*, *63*(3), 311-329. https://doi.org/10.1007/s11528-018-0295-4

Riesen, K. (2023) . Graduate student learning and the development of programmatic instruction: A mixed methods needs assessment. *The Journal of Academic Librarianship*, *49*(1) , 102631. https://doi.org/https://doi.org/10.1016/j.acalib.2022.102631

Roungrong, P., Kaewurai, R., Namoungon, S., Changkwanyeun, A., & Tengkew, S. (2018). Computational thinking with Thai education. *Panyapiwat Journal*, *10*(3), 322-330.

Rueangrong, P., Kaewurai, R., Namoungon, S., Changkwanyeun, A. & Tengkew, S. (2018). Computational Thinking with Thai Education. *Panyapiwat Journal*, *10*(3), 322-330. Retrieved 26 August, 2024 from https://tinyurl.com/2vksbzfy

Rueangrong, P. & Phitthayasenee, M. (2021). Computational concept is combined with the coding learning management model to enhance Collaborative problem-solving skills. *Journal of Graduate Studies in Northern Rajabhat Universities,* 11(1), 1-16. Retrieved 26 August, 2024 from https://tinyurl.com/ym7vt6xf

Sharov, S., Tereshchuk, S., Tereshchuk, A., Kolmakova, V., & Yankova, N. (2023). Using MOOC to Learn the Python Programming Language. *International Journal of*

Emerging Technologies in Learning (Online), *18*(2), 17. https://doi.org/10.3991/ijet.v18i02.36431

Siriphatcharachot, P., Sukkamart, A., Thongkaw, A., Pimdee, P., & Moto, S. (2024). High School Student Creativity, Innovation, and Teamwork Skills from a Technology Teacher's Perspective: A Second-Order Confirmatory Factor Analysis. *International Journal of Instruction*, *18*(1), 39-60. Retrieved 1 October, 2024 from https://www.e-iji.net/dosyalar/iji_2025_1_3.pdf

Sittikhetkron, W., & Sawangmek, S. (2021). Development of Computational Thinking Skill Through 5ES Inquiry Learning Activities with Board Game and Formula Coding on the Population in Pandemic for Grade 12 Students. *Journal of Education and Innovation*, *23*(3), 286–300. Retrieved 26 August, 2024 from https://tinyurl.com/yucn98kv

Sittisak, R., Sukamart, A., & Kantathanawat, T. (2022). Thai student teacher learning management skills: A stakeholder needs assessment. *Journal of Positive Psychology and Wellbeing, 6*(2), 901-916.

Songkhram, C., Klineam, C. & Supap, W. (2020). The Development of Computational Thinking by using the Problem-Based Learning in Probability for Grade 10th Students. *Silpakorn Educational Research Journal, 12*(1), 203-217. Retrieved 26 August 2024 from https://tinyurl.com/2m8bbnaw

Stephens, M., & Kadijevich, D. M. (2020). Computational/algorithmic thinking. *Encyclopedia of Mathematics Education*, 117-123. https://doi.org/10.1007/978-3-030-15789-0_100044

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

Suadang, O., Yanprechaset, Y. & Paiwithayasitham, C. (2020). Guideline to develop learning media for distance education of undergraduates: A complete needs assessment research, *STOU Journal, 33*(2), 134-151. Retrieved 26 August, 2024 from https://tinyurl.com/3esxkpar

Sukardi, Wildan, & Subhani, A. (2022). Experiential learning in entrepreneurship teaching: An evaluation based on importance-performance analysis. *International Journal of Instruction*, *15*(4), 453-472. https://doi.org/10.29333/iji.2022.15425a

Sukkamart, A., Sermsri, N., Kantathanawat, T., Nakwijit, R., & Meekhobtong, S. (2024). Computer Education Student Teacher Complex Problem-Solving Skills Development using Computational Thinking and Visualization Tools. *Pakistan Journal of Life and Social Sciences, 22*(2), 4915-4923. https://doi.org/10.57239/PJLSS-2024-22.2.00364

Thongkum, S., & Buaraphan, K. (2023). Problems and Needs in Experiential Learning in Mathematics: Teachers' and Students' Perspectives From Thailand. The Asian

Conference on Education 2023. Official Conference Proceedings. Retrieved 26 August, 2024 from https://tinyurl.com/2sjj6mne

Tobua, S., Khemtong, P., & Thammakittipob, V. (2018). Use of Need Assessment for Personnel Development Design in Vocational Education Institution Project Evaluation. *Journal of Research and Curriculum Development*, *2*(8), 69-87. Retrieved 26 August, 2024 from https://tinyurl.com/drrucdnu

Uğraş, T., Rızvanoğlu, K., & Gülseçen, S. (2022). New co-design techniques for digital game narrative design with children. *International Journal of Child-Computer Interaction*, *31*, 100441. https://doi.org/https://doi.org/10.1016/j.ijcci.2021.100441

Ussarn, A., Pimdee, P., & Kantathanawat, T. (2022). Needs assessment to promote the digital literacy among students in Thai community colleges. *International Journal of Evaluation and Research in Education*, *11*, 1278-1284. https://doi.org/10.11591/ijere.v11i3.23218

Videnovik, M., Madevska Bogdanova, A., & Trajkovik, V. (2024). Game-based learning approach in computer science in primary education: A systematic review. *Entertainment Computing*, *48*, 100616. https://doi.org/https://doi.org/10.1016/j.entcom.2023.100616

Wing, J. M. (2006). Computational thinking. *Commun. ACM*, *49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Wongwanich, S., & Wiratchai, N. (2005). A follow-up and evaluation of the government educational reform results based on the state fundamental policy and the national act. *Journal of Research Methodology*, 18(1), 93 – 124. (In Thai)

Wongwanich, S. (2019). *Needs Assessment Research* (4 ed.). Chulalongkorn university press. (In Thai)

Wuttikamonchai, O., Pimdee, P., Ployduangrat, J., & Sukkamart, A. (2024). A needs assessment evaluation of information technology student mobile website design skills. *Contemporary Educational Technology, 16*(1), ep494. https://doi.org/10.30935/cedtech/14173

Yurayat, P., & Seechaliao, T. (2021). Needs assessment to develop online counseling program. *International Education Studies, 14*(7), 59-71. Retrieved 26 August, 2024 from https://tinyurl.com/y4ypty9s