An Investigation of Affective Factors Influencing Computational Thinking and Problem-Solving

Hyunchang Moon and Jongpil Cheon*

***Abstract*—This study investigated the effects of affective factors on computational thinking and problem-solving. Computer science subjects are becoming part of the regular curricula in K-12 and higher education to enhance computational problem-solving skills. However, affective factors influencing computational thinking skills and computational thinking components predicting problem-solving skills have yet to be fully explored. This paper proposed a conceptual model to predict (a) four affective factors that influence computational thinking self-efficacy and (b) six computational thinking components that affect problem-solving self-efficacy. Structural equation modeling was used to analyze self-report data from college students to examine the direct relationships among study variables. The findings showed that two affective factors (i.e., programming self-efficacy and computer science usefulness) significantly predicted computational thinking self-efficacy and influenced problem-solving self-efficacy. Also, two computational thinking components (i.e., algorithm and debugging) were the significant determinants of problem-solving self-efficacy. The results validate the importance of affective factors in computer science education and suggest specific computational thinking activities that should be emphasized in computer science curricula to facilitate problem-solving skills.**

***Index Terms*—Affective factors, computational thinking, computer science education, block-based programming**

I. Introduction

Today's pervasiveness and rapid evolvement of computer technology are demanding new competencies indispensable to success in our digital society. The ability to enhance Problem-Solving (PS) with computing has become an integral part of our daily lives and tasks. Beyond the "4Cs" for 21st-century skills (i.e., critical thinking, creativity, collaboration, and communication [1–5]), Computational Thinking (CT) has been argued to be another core skill or the "fifth C." Also, Computer Science (CS) education has been emphasized to provide students with more CT learning opportunities that will enable them to be active in the digital world [6–9]. CT can be fostered by introducing CT competencies (i.e., knowledge, skill, and attitude) in CS subjects and incorporating them into cross-curricular disciplines so that CT can be leveraged across disciplines and everyday life [10–14]. As a result of the recent transition in CS education, the Next Generation Science Standards (NGSS) and Common Core State Standards (CCSS) were restructured to embrace CT as an interdisciplinary approach that can promote synergy with other existing core standards [15–17]. These shifts in CS education have led researchers and educators to focus more on defining and assessing CT competency, which can evaluate evidence of the learning process and enhance the effectiveness of instruction [18–22]. Although much effort has been devoted to developing CT through programming, robotics, gaming, and unplugged activities in various educational environments [23, 24], very little research has been conducted on CT's affective factors as a critical predictor of CT competency, and the relationship of CT's underlying competencies with problem-solving skills.

There remains a need for studies to investigate the affective aspects towards CS, CT, and PS for promoting an active learning experience in computer science education. One way to move this area of research forward was to examine what affective factors influence computational thinking and what CT components contribute to PS. The present study, therefore, intended not only to identify affective factors influencing CT competency but also to examine the structural relationship between CT components and PS in a higher education course. To this end, the following two questions guided this study:

1) What affective factors influence computational thinking self-efficacy?
2) What computational thinking components affect problem-solving self-efficacy?

II. Related Literature

A. Computer Science Education

In recent years, attention and efforts in CS education around the world have rapidly increased to prepare them to develop the knowledge and skills effectively used to not only benefit future career and educational opportunities but also deal with the new challenges of the 21st century [2, 6, 8, 15]. According to the Association for Computing Machinery, Computer Science (CS) is defined as "the study of computers and algorithmic processes, including their principles, their hardware and software designs, their applications, and their impact on society" [4] and is distinct from computer literacy, which merely focuses on using computer technologies. Educators, researchers, and policymakers all have increasingly recognized that CS is a new drive necessary for today's technology innovation, economic opportunity, and social mobility [1–3, 12, 13, 25, 26]. High-quality CS education enables all students to acquire skills for the future, including analytic thinking, creativity, and teamwork [1, 25, 26]. In this context, it has been agreed that K-12 and higher education should prepare learners in CS education to develop

Manuscript received April 15, 2022; revised July 7, 2022; accepted November 1, 2022.

Hyunchang Moon is with the Department of Pediatrics, Medical College of Georgia, Augusta University, Augusta, USA. E-mail: hymoon@augusta.edu (H.M.)

Jongpil Cheon is with the Department of Curriculum and Instruction, Texas Tech University, Lubbock, USA.

*Correspondence: jongpil.cheon@ttu.edu (J.C.)

doi: 10.18178/ijiet.2023.13.10.1956

CT by incorporating the underpinnings of CS competency and new approaches to solving a problem [11, 13, 14, 27]. Therefore, it is crucial to ensure that students of various demographics (e.g., age, gender, race, ethnicity, and socioeconomic status) have the opportunity to acquire these skills.

B. Computational Thinking

Since CT encompasses broad topics across disciplines, it has been defined in a variety of contexts [11−14]. Wing [28] began to publicize the discourse regarding the role of CT across all disciplines and how it contributes to PS in the digital age. She underlined the CT processes of "formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" [29]. The National Research Council [10] underlined the role of programming in constructing a sequence of steps for solving a problem. Barr and Stephenson [27] specified an operational definition of CT for K-12 education, which they described as a PS process and a series of dispositions and attitudes: (a) formulating problems in a way that enables us to use a computer and other tools to help solve them; (b) logically organizing and analyzing data; (c) representing data through abstractions such as models and simulations; (d) automating solutions through algorithmic thinking; (e) identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources; (f) generalizing and transferring this PS process to a wide variety of problems [30]. Many other researchers have elaborated various definitions to suit their research situation [24, 31–33]. As the definition of CT evolves, the three-dimensional framework of CT suggested by Brennan and Resnick [34] has been widely accepted in the context of learning CT through programming [20, 21]. Its three fundamental dimensions are as follows: (a) seven computational concepts (i.e., sequences, loops, events, parallelism, conditionals, operators, and data), (b) four computational practices (i.e., experimenting and iterating, testing and debugging, reusing and remixing, and abstracting and modularizing), and (c) three computational perspectives (expressing, connecting, and questioning) [34].

C. Programming

Advances in computing technology allow people to come up with new PS approaches and examine virtual solutions in the real world. Resnick [35] argued that CT competency requires not just the ability to use computing tools but also the ability to design, deliver, and communicate with new solutions. While CT can be applied to various challenges that are not directly related to programming tasks, programming is a fundamental way to develop CT effectively [3, 5, 9, 18, 36]. The ability to program plays an essential role in enhancing the power of the CT process [18]. For example, programming provides a new perspective for PS strategies—such as decomposition, pattern recognition, abstract, and algorithm—that can be applied to non-programming domains. Also, since programming involves developing thought processes for solving problems, it can provide people with opportunities to reflect on their learning [18, 37]. Due to the close relationship of programming with CT, several studies found that programming is frequently adopted to teach CT skills [3, 5, 9, 18, 36]. However, at the same time, programming was perceived to be highly challenging, such that many people misunderstood computer programming as a limited realm used only by a particular population [3−5, 19]. This may be because early programming learning did not make it easy for many beginners to learn text-based programming languages. In addition, programming was often introduced without sufficient guidance when learners needed help with their codes or in activities that were not relevant to learners' interests and prior knowledge [38]. In recent years, to overcome these obstacles, there have been different attempts to introduce beginners to online block-based programming platforms, such as Scratch, MIT App Inventor, MakeCode, and Lego WeDo [5, 39].

D. Problem-Solving

What are the traits of an effective PS process? There have been several approaches to defining the PS process. Lester and Charles [40] defined PS as the process of coordinating cognitive ability, affective ability, and experience to determine a solution. Schoenfeld [41] outlined four factors that affected students' PS process in mathematics: (a) resources (domain knowledge), (b) heuristics (PS strategy), (c) monitoring and control (self-regulation), and (d) beliefs and affects (perspectives of individuals towards themselves and their environment). According to the problem-solving behavioral theory proposed by Tallman *et al*. [42], PS involves a set of sequences, including (a) defining the problem, (b) gathering information, (c) developing solutions, and (d) evaluating results. Mayer [43] proposed that the success of PS depended on three factors: (a) skill (domain-specific knowledge), (b) meta-skill (metacognition), and (c) will (motivational aspects of cognition). In other words, to solve a problem, students should acquire the relevant skill, meta-skill, and will. The meta-skill is central in PS because it coordinates the cognitive process and other components. The three parts are aligned with the knowledge, skill, and attitude model [44] to measure PS competencies. Similarly, Bloom [45] categorized the three domains of learning as cognitive (knowledge acquisition), psychomotor (physical skills), and affective (feelings, emotions, and attitudes). Jonassen [46] emphasized the cognitive and affective requirements needed to solve different problems. As noted previously, affective factors have a significant role in developing PS ability.

E. Affective Factors

In addition to being a critical factor in PS, many studies revealed that affective factors significantly impact the PS process. For instance, Mason [47] noted that as students' beliefs about PS increased, their PS achievements were enhanced accordingly. In contrast, anxiety and fear towards subjects were a factor that negatively affected students' PS achievements. Likewise, Hoffman and Schraw [48] confirmed a positive relationship between students' PS self-efficacy and PS efficiency. In the present study, these affective factors included attitudes towards CS, perceived usefulness of CS, and self-efficacy regarding Programming,

CT, and PS. These emotional factors have been widely investigated in various fields and defined by distinctive elements. Self-efficacy is a person's belief about their capability to perform a behavior that deals with a particular problem and to produce designated levels of performance [49]. Attitude is a person's evaluative judgment, which depends on their prior knowledge and experience [50]. Usefulness refers to the degree to which a person believes that using a tool will help them to attain gains in job performance [51]. To summarize, affective factors are a set of emotions and attitudes people have about themselves or their surroundings. These factors influence student performance in either positive or negative ways.

F. Research Model and Hypothesis Development

The aim of this study was to investigate the relationships among learners' affective factors—computer science attitude, computer science usefulness, programming self-efficacy, programming attitude, computational thinking self-efficacy, and PS self-efficacy—in the context of an online block-based programming environment. Many studies confirmed that domain-specific knowledge and metacognition skills could promote computational thinking [52–56]. Also, other studies discovered a strong link between CT and PS [36, 57–59]. However, there is a lack of knowledge regarding how affective factors interact to influence CT, as well as how CT sub-competencies affect PS, particularly in higher education and online learning environments. In addition, the effects of CT components such as decomposition, pattern recognition, planning, algorithm, reusing, and debugging on PS self-efficacy were explored. Thus, the direction and degrees of structural relationships were examined among affective factors, CT, and PS. The research model is illustrated with the related hypotheses in Fig. 1, and the hypotheses are presented in Table I.

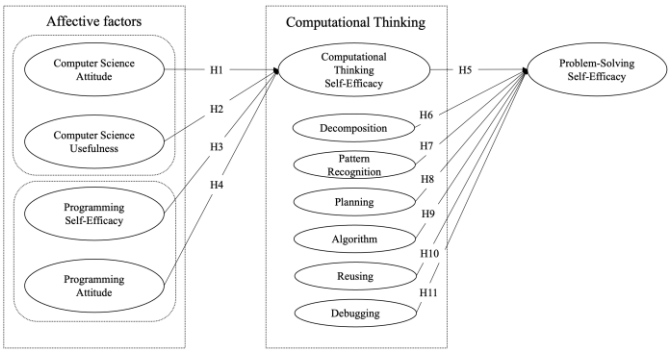

Fig. 1. Research model with hypotheses.

TABLE I: RESEARCH HYPOTHESES

#	Hypotheses
H1	*Computer Science (CS) Attitude* directly and positively influences learner's *Computational Thinking (CT) Self-Efficacy*.
H2	*Computer Science (CS) Usefulness* directly and positively influences learner's *Computational Thinking (CT) Self-Efficacy.*
H3	*Programming Self-Efficacy* directly and positively influences learner's *Computational Thinking (CT) Self-Efficacy*.
H4	*Programming Attitude* directly and positively influences learner's *Computational Thinking (CT) Self-Efficacy*.
H5	*Computational Thinking (CT) Self-Efficacy* directly and positively influences learner's *Problem-Solving (PS) Self-Efficacy*.
H6	*Decomposition* directly and positively influences learner's *Problem-Solving (PS) Self-Efficacy*.
H7	*Pattern recognition* directly and positively influences learner's *Problem-Solving (PS) Self-Efficacy*.
H8	*Planning* directly and positively influences learner's *Problem-Solving (PS) Self-Efficacy*.
H9	*Algorithm* directly and positively influences learner's *Problem-Solving (PS) Self-Efficacy*.
H10	*Reusing* directly and positively influences learner's *Problem-Solving (PS) Self-Efficacy*.
H11	*Debugging* directly and positively influences learner's *Problem-Solving (PS) Self-Efficacy*.

III. METHOD

A. Participants

Two types of sampling techniques were used for this population. These included convenience sampling to quickly reach a targeted sample, and identical sampling to simultaneously produce quantitative and qualitative phases [60]. Participants were a total of 69 college students who were enrolled in an online course, Computing and Information Technology, at a large public university. The participants were enrolled in varied majors, were of various ages, and were both male and female (age: M = 25.26 years, range = 19–47 years, female: 54%). They had little-to-no previous programming experience prior to taking the online course. The sample size was considered adequate, providing ≥ 80% power to reject a non-closely fitting model (i.e., root mean squared error of approximation [RMSEA] = 0.10) in favor of a closely fitting model (i.e., RMSEA = 0.05) at 0.05 alpha level.

B. Research Setting

The course was intended to provide students with basic CS knowledge and programming experiences using Scratch block-based programming language and environment to create and share their programming projects easily. A web-based learning management system was used to administer and provide online learning during a semester. Scratch programming was adopted because it is suitable as an introductory programming language for people of all ages. The participants were asked to perform a programming task using Scratch Version 3.0. There was a total of 13 modules, including six modules related to programming. The main tasks were to complete semi-finished Scratch projects, and the final project was designing and developing a platform game in which a player could make a character jump or climb to avoid obstacles and reach a destination or perform a mission.

C. Instrument

This study used an online survey questionnaire to measure the perceived level of affective factors. The survey consisted of demographic questions and 18 closed-ended questions (three questions per construct) asking about the six constructs: (a) computer science attitude, (b) computer science usefulness, (c) programming self-efficacy, (d) programming attitude, (e)

computational thinking self-efficacy, and (f) problem-solving self-efficacy. Responses to the closed-ended questions were scored on a 5-point Likert scale ranging from "strongly disagree" to "strongly agree." Survey items were adapted from the CS Attitude [61], Programming Self-Efficacy [62], and CT/Problem-Solving Self-Efficacy [63]. Two professionals independently reviewed the survey items. Based on their feedback, we revised the items to enhance clarity and accessibility. The internal consistency of the survey items was confirmed (Cronbach's $\alpha = 0.92$).

D. Data Collection and Analysis

An online form of the survey was linked in the last module, and participants received an extra point for voluntarily taking the survey (71% response rate). The answers to the survey items were stored in the database and statistically analyzed using RStudio [64]. In a preliminary analysis, assumptions of normality, linearity, and homoscedasticity were met. Descriptive statistics were also calculated to summarize the demographic characteristics of the participants, as shown in Table II.

TABLE II: DESCRIPTIVE STATISTICS

Demographics	Item	Frequency	%
Gender	Female	32	46.4
	Male	37	53.6
Age	Below 30 years	52	75.4
	31–40 years	13	18.8
	41 years and above	4	5.8
Grade	Freshman	7	10.1
	Sophomore	15	21.7
	Junior	20	29.0
	Senior	27	39.1

Confirmatory Factor Analysis (CFA) was employed to examine the hypothesized factors of the research model, which illustrates the directions and links between the latent variables (i.e., factors) and their observed variable (i.e., closed-ended questions) as well as the associations among the factors. The CFA model included the following parameters: (a) item loadings on six affective factors—computer science attitude, computer science usefulness, programming self-efficacy, programming attitude, computational thinking self-efficacy, problem-solving self-efficacy, (b) covariances among the six factors, and (c) residual variances/covariances and intercepts of the items.

Next, Structural Equation Modeling (SEM) was conducted to examine predictive relationships among the latent variables. The SEM model involved the same measurement part as the CFA model, but the structural relationships were modified with the regression paths: (a) from computer science attitude, computer science usefulness, programming self-efficacy, programming attitude (predictors) to computational thinking self-efficacy (outcome); (b) from computational thinking self-efficacy (predictor) to problem-solving self-efficacy (outcome); and (c) from decomposition, pattern recognition, planning, algorithm, reusing, and debugging (predictors) to problem-solving self-efficacy (outcome). The model parameters were estimated via robust maximum likelihood [65]. The overall model fit was evaluated with multiple indicators: chi-square statistic (χ^2), RMSEA, Standardized Root Mean Square Residual (SRMR), and two comparative fit indices [66–69].

IV. RESULTS

Descriptive statistics and bivariate correlations were calculated and are presented in Table III and Table IV, respectively. The overall model fit was evaluated using chi-square statistic (χ^2), RMSEA, SRMR, Comparative Fit Index (CFI), and Tucker-Lewis index (TLI) because various indicators may show different features of model fit [70]. The chi-square statistic and its *p*-value should not be significant at the 0.05 level if there is a good model fit. The chi-square ($\chi^2 = 181.65$, df = 124) was statistically significant ($p < 0.001$); however, the chi-square statistic is very sensitive to sample size and is weak to use as a single criterion of acceptance or rejection [71, 72]. To overcome this issue, other indices are recommended to determine the fit [65, 68]. The model fit indices indicated a good fit between the observed data and the structural causal model, as shown in Table V. The RMSEA indicates the absolute fit of the hypothesized model in the population covariance matrix. The RMSEA estimate was 0.07, and its associated 90% Confidence Interval (CI) was (0.03, 0.09), which also indicates a good model fit [73]. The difference between the residuals of the sample covariance and the hypothesized covariance model is represented by the SRMR. Since the lower SRMR values (≤ 0.08) are generally considered a good fit [66], the SRMR estimate in the model obtained 0.05, indicating a good model fit. Lastly, the CFI calculates how well the hypothesized model fits the data compared to a null model of uncorrelated variables. The comparative indices of CFI and TLI were 0.95 and 0.94, respectively, showing a good and acceptable model fit [65, 68]. Although the results of the chi-square test and RMSEA's CI are not sufficient, the rest of the indices indicated that the fit of the model deems adequate and validated the hypotheses of the research model.

TABLE III: BIVARIATE CORRELATIONS OF VARIABLES

	Affective Factors				CT	PS
	1	2	3	4	5	6
1. CS Attitude	1					
2. CS Usefulness	0.831**	1				
3. Programming Self-Efficacy	0.637**	0.602**	1			
4. Programming Attitude	0.563**	0.545**	0.555**	1		
5. CT Self-Efficacy	0.642**	0.641**	0.754**	0.586**	1	
6. PS Self-Efficacy	0.465**	0.550**	0.429**	0.441**	0.639**	1

	Computational Thinking						CT	PS
	1	2	3	4	5	6	7	8
1. Decomposition	1							
2. Pattern Recognition	0.802**	1						
3. Planning	0.670**	0.761**	1					
4. Algorithm	0.624**	0.682**	0.745**	1				
5. Reusing	0.896**	0.786**	0.730**	0.689**	1			
6. Debugging	0.724**	0.723**	0.810**	0.702**	0.823**	1		
7. CT Self-Efficacy	0.887**	0.895**	0.885**	0.831**	0.926**	0.899**	1	
8. PS Self-Efficacy	0.536**	0.521**	0.521**	0.634**	0.597**	0.597**	0.639**	1

** Correlation is significant at the 0.01 level (2-tailed).

TABLE IV: UNSTANDARDIZED AND STANDARDIZED ESTIMATES OF THE PREDICTIVE EFFECTS ON COMPUTATIONAL THINKING

Parameter	*b*	*SE*	*β*	*p*
Regression Path to *CT*				
Computer Science Attitude	0.09	0.23	0.11	0.280
Computer Science Usefulness	0.20	0.06	0.29	0.000
Programming Self-Efficacy	0.48	0.08	0.58	0.000
Programming Attitude	0.11	0.29	0.17	0.622
Regression Path to *PS Self-Efficacy*				
Decomposition	0.11	0.20	0.12	0.378
Pattern Recognition	0.03	0.02	0.04	0.789
Planning	–0.04	0.17	–0.04	0.830
Algorithm	0.41	0.14	0.42	0.023
Reusing	0.10	0.21	0.12	0.489
Debugging	0.27	0.11	0.30	0.002

SE = Standard Error

TABLE V: SUMMARY OF MODEL FIT INDICES AND CUT-OFF THRESHOLDS

Measure	Cut-off	Result
χ^2 *p*-value	> 0.05	0.001
RMSEA	0.06–0.08 (acceptable), < 0.06 (good)	0.07 [0.03, 0.09]
SRMR	0.05–0.08 (acceptable), < 0.05 (good)	0.05
CFI	0.90–0.95(acceptable), > 0.95 (good)	0.95
TLI	0.90–0.95 (acceptable), > 0.95 (good)	0.94

For the first research question concerning affective factors influencing CT, the results suggested that computer science usefulness and programming self-efficacy were positively and significantly linked to computational thinking self-efficacy. The rest of the latent variables—i.e., computer science attitude and programming attitude—were not significant. As hypothesized in Table I, programming self-efficacy and computer science usefulness significantly predicted computational thinking self-efficacy (Hypotheses 2 and 3). 71% of computational thinking self-efficacy variance was explained by the two latent variables, among which programming self-efficacy was the stronger predictor with path loading (β = 0.58, p < 0.001). The direct effects of computer science usefulness had slightly lower influences with path loading (β = 0.29, p < 0.01). However, two attitudinal factors (i.e., computer science attitude and programming attitude) had no significant influence on computational thinking self-efficacy (Hypotheses 1 and 4) (p = 0.41–0.59).

Computational thinking self-efficacy was also positively related to problem-solving self-efficacy. For the second research question, it was found that the effects of two CT components, algorithm and debugging, were positively significant on problem-solving self-efficacy. In contrast, the rest of the components— decomposition, pattern recognition, planning, and reuse—were not significant. The analysis results of the final structural model are illustrated in Fig. 2 and detailed in Table IV. Computational thinking self-efficacy was a strong predictor of problem-solving self-efficacy (Hypothesis 5) (β = 0.64, p < 0.01). Algorithm and debugging were moderate predictors of problem-solving self-efficacy (Hypotheses 9 and 11) (p = 0.02), but decomposition, pattern recognition, planning, and reuse did not predict problem-solving self-efficacy (Hypotheses 6, 7, 8, and 10) (p = 0.04–0.30).

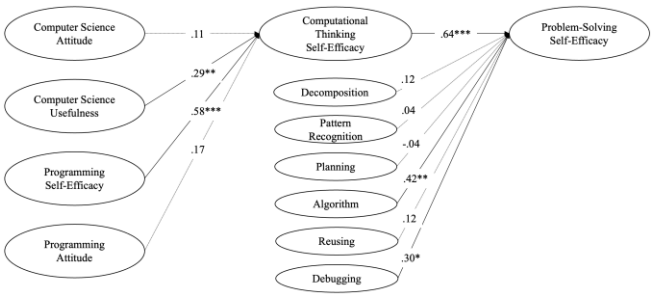

Fig. 2. Standardized coefficients of structural equation model (* p < 0.05, ** p < 0.01, *** p < 0.001).

V. DISCUSSION AND IMPLICATIONS

This study aimed to explore the structural relationships among learner affective factors, CT self-efficacy, and PS self-efficacy in the context of CT learning via block-based programming. These relationships could have meaningful implications for evaluating the affective factors, thereby deriving insights on instructional designs to facilitate computational thinking.

According to the model analysis findings, the overall theoretical framework was statistically established, and two affective determinants (i.e., programming self-efficacy and computer science usefulness) explain students' CT development. Additionally, for the significant effect of CT self-efficacy on problem-solving self-efficacy, the theoretical assumption from previous studies was statistically confirmed. Some previous studies have revealed a strong link between CT and PS by examining CT knowledge and skills [36, 57–59]. For this reason, the current research has implications for contributing to the validation of affective factors predicting CT and the predictive relationship between CT and PS.

Interestingly, program self-efficacy was the most vital affective factor promoting computational thinking. Program self-efficacy represents how well students can plan and write blocks of code and explain a Scratch program they created. In other words, the ability to create a Scratch program has been confirmed to be closely related to computational thinking, which is consistent with the findings of previous studies [14, 36–38]. The National Research Council [10] also highlighted the role of programming in constructing a sequence of steps for solving a problem. Hence, it can be concluded that programming experience is critical to promoting computational thinking skills. Also, the perceived usefulness of CS exerted positive effects on computational thinking. However, the present study found that attitude toward CS was not a significant factor. Since the attitude toward computer science asked to what extent the participants were interested in CS, computer science usefulness was about the perceived benefits of computing skills in their lives, meaning that fostering the value of CS usefulness in curricula may facilitate computational thinking skills.

Another finding was the significant relationship between computational thinking components and problem-solving self-efficacy. Although every CT component has a positive relationship with problem-solving self-efficacy, the current

study found that out of six CT components, algorithmic thinking and debugging are significant determinants of problem-solving self-efficacy. In other words, the levels of the two CT components positively predict problem-solving levels. Thus, these two activities should be emphasized in K-12 and higher education curricula to improve problem-solving skills. For instance, algorithmic thinking can be developed by building ways of thinking of a step-by-step solution. Debugging can also be taught by providing students with various testing and debugging experiences.

VI. Conclusion

The present study provides insights allowing us to better understand learners' affective factors (i.e., attitude, self-efficacy, and usefulness) and CT sub-components for problem-solving skills with a limited sample size. The results thus need to be interpreted with caution and may not be generalizable to all settings or other age groups. Further research on different cultures, institutions, and populations is needed for them to be verified. Moreover, since only three emotional factors were considered in this study, the effects of other affective factors not included in the study were not investigated. Future research may need to incorporate different affective factors to obtain a holistic picture. Lastly, complementary data (e.g., CS knowledge test, Scratch programming artifacts assessment, problem-solving performance test, open-ended questions about CS and CT) should be considered to establish the relationships in the research model to evaluate CT knowledge and skill domains comprehensively. With these limitations in mind, most of the model fit indices supported that the fit of the model was adequate, and the research model was validated. The findings of future studies will contribute to a better understanding of the affective factors of learners as well as new ways to redesign course curricula for teaching CT skills effectively.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Jongpil and Hyunchang designed the study. Hyunchang collected and analyzed the data. Both wrote the initial manuscript, and Hyunchang revised the manuscript. All authors approved the final manuscript.

References

[1] F. Buitrago-Flórez, G. Danies, S. Restrepo, and C. Hernández, "Fostering 21st century competences through computational thinking and active learning: A mixed method study," *International Journal of Instruction*, vol. 14, no. 3, pp. 737–754, 2021.

[2] S. Grover and R. Pea, "Computational thinking: A competency whose time has come," *Computer Science Education: Perspectives on Teaching and Learning in School*, vol. 19, pp. 1257–1258, 2018.

[3] J. Nouri, L. Zhang, L. Mannila, and E. Norén, "Development of computational thinking, digital competence and 21st century skills when learning programming in K-9," *Education Inquiry*, vol. 11, no. 1, pp. 1–17, 2020.

[4] C. Angeli and M. Giannakos, "Computational thinking education: Issues and challenges," *Computers in Human Behavior*, vol. 105, pp. 106–185, 2020.

[5] H. Montiel and M. G. Gomez-Zermeño, "Educational challenges for computational thinking in K–12 education: A systematic literature review of "Scratch" as an innovative programming tool," *Computers*, vol. 10, no. 6, p. 69, 2021.

[6] A. Lamprou and A. Repenning, "Teaching how to teach computational thinking," in *Proc. the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, pp. 69–74, 2018.

[7] D. J. Ketelhut, K. Mills, E. Hestness, L. Cabrera, J. Plane, and J. R. McGinnis, "Teacher change following a professional development experience in integrating computational thinking into elementary science," *Journal of Science Education and Technology*, vol. 29, no. 1, pp. 174–188, 2020.

[8] T. Šiaulys, "Developing interactive visualizations focusing on computational thinking in K-12 computer science education," in *Proc. the 26th ACM Conference on Innovation and Technology in Computer Science Education*, vol. 2, pp. 680–681, 2021.

[9] Y. T. Lin, M. T. Wang, and C. C. Wu, "Design and implementation of interdisciplinary STEM instruction: Teaching programming by computational physics," *The Asia-Pacific Education Researcher*, vol. 28, no. 1, pp. 77–91, 2019.

[10] National Research Council, *Report of a Workshop on the Scope and Nature of Computational Thinking*, National Academies Press, 2010.

[11] I. Lee, S. Grover, F. Martin, S. Pillai, and J. Malyn-Smith, "Computational thinking from a disciplinary perspective: Integrating computational thinking in K-12 science, technology, engineering, and mathematics education," *Journal of Science Education and Technology*, vol. 29, no. 1, pp. 1–8, 2020.

[12] S. Psycharis, "STEAM in education: A literature review on the role of computational thinking, engineering epistemology and computational science. Computational steam pedagogy (CSP)," *Scientific Culture*, vol. 4, no. 2, pp. 51–72, 2018.

[13] P. J. Denning and M. Tedre, "Computational thinking: A disciplinary perspective," *Informatics in Education*, vol. 20, no. 3, pp. 361–390, 2022.

[14] J. Olmo-Muñoz, R. Cózar-Gutiérrez, and J. A. González-Calero, "Computational thinking through unplugged activities in early years of primary education," *Computers & Education*, vol. 150, pp. 103–832, 2020.

[15] S. Grover and R. Pea, "Computational thinking in K–12: A review of the state of the field," *Educational researcher*, vol. 42, no. 1, pp. 38–43, 2013.

[16] S. Grover, K. Fisler, I. Lee, and A, Yadav, "Integrating computing and computational thinking into K-12 STEM learning," in *Proc. the 51st ACM Technical Symposium on Computer Science Education*, pp. 481–482, 2020.

[17] K. M. Rich, A. Yadav, and C. V. Schwarz, "Computational thinking, mathematics, and science: Elementary teachers' perspectives on integration," *Journal of Technology and Teacher Education*, vol. 27, no. 2, pp. 165–205, 2019.

[18] F. J. Agbo, S. S. Oyelere, J. Suhonen, and S. Adewumi, "A systematic review of computational thinking approach for programming education in higher education institutions," in *Proc. the 19th Koli Calling International Conference on Computing Education Research*, pp. 1–10, 2019.

[19] S. Grover, G. Biswas, A. Dickes *et al*., "Integrating STEM and computing in PK-12: Operationalizing computational thinking for STEM learning and assessment," in *the Interdisciplinarity of the Learning Sciences, 14th International Conference of the Learning Sciences (ICLS)*, vol. 3, 2020.

[20] H. İ. Haseski, U. Ilic, and Tugtekin, U, "Defining a new 21st century skill-computational thinking: Concepts and trends," *International Education Studies*, vol. 11, no. 4, pp. 29–42, 2018.

[21] N. Zhang and G., Biswas, "Defining and assessing students' computational thinking in a learning by modeling environment," *Computational Thinking Education*. Springer, Singapore, pp. 203–221, 2019.

[22] Y. Allsop, "Assessing computational thinking process using a multiple evaluation approach," *International Journal of Child-Computer Interaction*, vol. 19, pp. 30–55, 2019.

[23] V. J. Shute, C. Sun, and J. Asbell-Clarke, "Demystifying computational thinking," *Educational Research Review*, vol. 22, pp. 142–158, 2017.

[24] A. Tucker, D. McCowan, F. Deek, C. Stephenson, J. Jones, and A. Verno, *A Model Curriculum for K–12 Computer Science: Report of the ACM K–12 Task Force Curriculum Committee*, 2nd ed. New York, Association for Computing Machinery, p. 2, 2006.

[25] National Science Foundation, *Computer Science for All (CSforAll: RPP)*, 2016.

[26] A. Yadav, S. Gretter, S. Hambrusch, and P. Sands, "Expanding computer science education in schools: understanding teacher experiences and challenges," *Computer Science Education*, vol. 26, no. 4, pp. 235–254. 2016.

[27] V. Barr and C. Stephenson, "Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community?" *Inroads*, vol. 2, no. 1, pp. 48–54, 2011.

[28] J. M. Wing, "Computational thinking," *Communications of the ACM*, vol. 49, no. 3, pp. 33–35, 2006.

[29] J. M., Wing, "Research notebook: Computational thinking—What and why?" *The Link Magazine*, Spring, 2011.

[30] CSTA & ISTE, *Operational Definition of Computational Thinking for K-12 Education*, 2011.

[31] S. Atmatzidou and S. Demetriadis, "Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences," *Robotics and Autonomous Systems*, vol. 75, pp. 661–670, 2016.

[32] M. Berland and U. Wilensky, "Comparing virtual and physical robotics environments for supporting complex systems and computational thinking," *Journal of Science Education and Technology*, vol. 24, no. 5, pp. 628–647, 2015.

[33] M. Israel, J. N. Pearson, T. Tapia, Q. M. Wherfel, and G. Reese, "Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis," *Computers and Education*, vol. 82, pp. 263–279, 2015.

[34] K. Brennan and M. Resnick, "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, vol. 1, p. 25, 2012.

[35] M. Resnick, "All I really need to know (about creative thinking) I learned (by studying how children learn) in kindergarten," in *Proc. the 6th ACM SIGCHI Conference on Creativity & Cognition*, pp. 1–6, ACM, 2007.

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

[37] A. A. DiSessa, *Changing Minds*, MIT Press, 2000.

[38] I. Cetin, "Preservice teachers' introduction to computing: Exploring utilization of scratch," *Journal of Educational Computing Research*, vol. 54, no. 7, pp. 997–1021, 2016.

[39] S. Sentance and A. Csizmadia, "Teachers' perspectives on successful strategies for teaching computing in school," *IFIP TCS*, 2015.

[40] F. Lester and R. Charles, "Teaching problem solving: What, why & how," *Dale Seymour Publications*, 1982.

[41] A. H. Schoenfeld, "Learning to think mathematically: Problem solving, metacognition, and sense making in mathematics," *Handbook of Research on Mathematics Teaching and Learning*, pp. 334–370, 1992.

[42] I. Tallman, R. K. Leik, L. N. Gray, and M. C. Stafford, "A theory of problem-solving behavior," *Social Psychology Quarterly*, pp. 157–177, 1993.

[43] R. E. Mayer, "Cognitive, metacognitive, and motivational aspects of problem solving," *Instructional Science*, vol. 26, no.1–2, pp. 49–63, 1998.

[44] K. N. Palmer, D. E. Ziegenfuss, and R. E. Pinsker, "International knowledge, skills, and abilities of auditors/accountants: Evidence from recent competency studies," *Managerial Auditing Journal*, vol. 19, no. 7, pp. 889–896, 2004.

[45] B. S. Bloom, *Taxonomy of Educational Objectives. Vol. 1: Cognitive Domain*, New York: McKay, pp. 20–24, 1956.

[46] D. H. Jonassen, *Learning to Solve Problems: A Handbook for Designing Problem-solving Learning Environments*, Routledge, 2010.

[47] L. Mason, "Personal epistemologies and intentional conceptual change," *Intentional Conceptual Change*, Routledge, pp. 204–241, 2003.

[48] B. Hoffman and G. Schraw, "The influence of self-efficacy and working memory capacity on problem-solving efficiency," *Learning and Individual Differences*, vol. 19, no. 1, pp. 91–100, 2009.

[49] A. Bandura, *Self-efficacy: The Exercise of Control*. Macmillan, 1997.

[50] I. Ajzen, *Attitudes, Personality, and Behavior*, McGraw-Hill Education, 2005.

[51] V. Venkatesh, M. G. Morris, G. B. Davis, and F. D. Davis, "User acceptance of information technology: Toward a unified view," *MIS Quarterly*, pp. 425–478, 2003.

[52] S. Basu, G. Biswas, and J. S. Kinnebrew, "Learner modeling for adaptive scaffolding in a computational thinking-based science learning environment," *User Modeling and User-Adapted Interaction*, vol. 27, no. 1, pp. 5–53, 2017.

[53] I. Fronza, N. E. Ioini, and L. Corral, "Teaching computational thinking using agile software engineering methods: A framework for middle schools," *ACM Transactions on Computing Education (TOCE)*, vol. 17, no. 4, p. 19, 2017.

[54] J. Moreno-León, G. Robles, and M. Román-González, "Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking," *Revista de Educación a Distancia (RED)*, vol. 46, pp. 1–23, 2015.

[55] D. Weintrop and U. Wilensky, "To block or not to block, that is the question: Students' perceptions of blocks-based programming," in *Proc. the 14th International Conference on Interaction Design and Children*, ACM, pp. 199–208, 2015.

[56] B. Zhong, Q. Wang, J. Chen, and Y. Li, "An exploration of three-dimensional integrated assessment for computational thinking," *Journal of Educational Computing Research*, vol. 53, no. 4, pp. 562–590, 2016.

[57] D. Barr, J. Harrison, and L. Conery, "Computational thinking: A digital age skill for everyone," *Learning & Leading with Technology*, vol. 38, no. 6, pp. 20–23, 2011.

[58] M. Román-González, J. C. Pérez-González, and C. Jiménez-Fernández, "Which cognitive abilities underlie computational thinking? Criterion validity of the computational thinking test," *Computers in Human Behavior*, vol. 72, pp. 678–691, 2017.

[59] M. G. Voskoglou and S. Buckley, "Problem solving and computational thinking in a learning environment," *arXiv preprint arXiv*:1212.0750, 2012.

[60] K. M. Collins, A. J. Onwuegbuzie, and Q. G. Jiao, "A mixed methods investigation of mixed methods sampling designs in social and health science research," *Journal of Mixed Methods Research*, vol. 1, no. 3, pp. 267–294, 2007.

[61] E. Wiebe, L. A. Williams, K. Yang, and C. S. Miller, "Computer science attitude survey," North Carolina State University, Dept. of Computer Science, 2003.

[62] M. Baser, "Attitude, gender and achievement in computer programming," *Middle-East Journal of Scientific Research*, vol. 14, no. 2, pp. 248–255, 2013.

[63] V. Kukul and S. Karatas, "Computational thinking self-efficacy scale: Development, validity and reliability," *Informatics in Education*, vol. 18, no. 1, pp. 151–164, 2019.

[64] R. C. Team, *A Language and Environment for Statistical Computing*, 2013.

[65] H. W. Marsh, Z. Wen, K. T. Hau, and B. Nagengast, "Structural equation models of latent interaction and quadratic effects," *Structural Equation Modeling: A Second Course*, pp. 225–265, 2006.

[66] L. T. Hu and P. M. Bentler, "Cutoff criteria for fit indexes in covariance structure analysis: Conventional criteria versus new alternatives," *Structural Equation Modeling: A Multidisciplinary Journal*, vol. 6, no. 1, pp. 1–55, 1999.

[67] J. H. Steiger and J. C. Lind, "Statistically based tests for the number of common factors," *Psychometric Society*, Iowa City, IA, 1980.

[68] P. M. Bentler, "Comparative fit indexes in structural models," *Psychological Bulletin*," vol. 107, no. 2, p. 238, 1990.

[69] R. B. Kline, *Methodology in the Social Sciences*, 2005.

[70] D. Hooper, J. Coughlan, and M. Mullen, "Structural equation modelling: Guidelines for determining model fit," *Articles*, vol. 2, 2008.

[71] P. M. Bentler and D. G. Bonett, "Significance tests and goodness of fit in the analysis of covariance structures," *Psychological Bulletin*, vol. 88, no. 3, p. 588, 1980.

[72] K. Schermelleh-Engel, H. Moosbrugger, and H. Müller, "Evaluating the fit of structural equation models: Tests of significance and descriptive goodness-of-fit measures," *Methods of Psychological Research Online*, vol. 8, no. 2, pp. 23–74, 2003.

[73] R. C. MacCallum, M. W. Browne, and H. M. Sugawara, "Power analysis and determination of sample size for covariance structure modeling," *Psychological Methods*, vol. 1, no. 2, p. 130, 1996.

Copyright © 2023 by the authors. This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited (CC BY 4.0).