Toward Developing a Real-World Computational Thinking Test Tool from Existing Models

Youngseok Lee, Jungwon Cho

***Abstract:** Various researches are always being carried out to measure the effectiveness of software education. We analyzed previously developed computational thinking tools and studied their practical application and verification methods. Using this information, we developed a 20-item questionnaire to categorize the tools by the abilities they measured: analysis, design, implementation, and reasoning. We surveyed college freshman and 204 students in computer programming subjects in liberal arts and then conducted an exploratory factor analysis to analyze the validity and reliability of our questionnaire test tool. Our test showed that previously used computational testing tools lacked the ability to measure problem-solving processes based on computational thinking. To solve this problem, we revised the questionnaire items to consider the problem-solving process based on computational thinking and proposed a tool that can check the computational thinking through the material of real life using the students' empirical knowledge. The statistical analysis was as follows: analysis ability (reliability α = .895); design ability (reliability α = .727); implementation ability (reliability α = .745), and reasoning ability (reliability α = .833). To measure computing errors, you need a testing tool that can address real-world problems. We aimed to develop a research tool for measuring computational thinking based on the case of applying and revising existing test tools.*

***Keywords :** Computational Thinking, Testing Tools, Exploratory Factors Analysis, Software Education.*

I. INTRODUCTION

Although the methods and models in software education presented by various countries differ, the main competencies in software education are generally listed as computational thinking, information culture literacy, and cooperative problem solving ability.

Computational thinking is solving problems through selecting and applying calculations. Since computational thinking has become one of the most important competencies of future talents in the knowledge information society, it is critical to know the most effective ways to teach it. Accordingly, many countries around the world have created and applied public education courses, and IT companies and nonprofit organizations are developing various content and teaching examples [1, 2].

As education concerned with computing errors (and their serious potential to cause accidents) progresses, there is an increasing need to know how to measure computational thinking ability, leading to various studies being conducted [3]. Computational thinking is an important foundation necessary to support increasingly technical layers of learning and to set appropriate educational goals for the learners.

Although the definition and concept of computational thinking differs by country and scholar, in recent years, various terms and concepts have been proposed and various testing tools have been developed to measure computational thinking ability [4]. The purpose of this study was to analyze the computer-aided versions of those thinking tools that have been developed to measure the effectiveness of information education and to revise and supplement the test tools to develop a real-world tool to test computational thinking in a form suitable for computer programming courses for university liberal education.

II. RELATED WORKS

This study analyzed the components of computational thinking ability and the study of computational thinking to verify the development and application examples of computing technology.

A. Components of Computational Thinking

In computational thinking, we draw a big picture to solve a problem in a way that can be computed by a computer, and it is an abstract method of thinking. In 2006, Janet Wing defined computational thinking as a concept involving problem solving, system design, and understanding human behavior based on the concept of computer science when we first encounter a problem to be solved [2, 5]. In recent years, we have emphasized abstraction and automation ability as core competencies, and these are skills that everyone should learn about computational thinking for in the future [6, 7].

Thanks to considerable research on computational thinking, sub-elements of computational thinking are expressed as concepts or components. This is because the sub-elements of computational thinking are similar to the process of using computers to solve complex and data-intensive problems [8, 9, 10]. Table- I summarizes the results of the problem-solving process using each country's conceptual style of

Revised Manuscript Received on July 22, 2019.

Youngseok Lee, KNU College of Liberal Arts and Sciences, Kangnam University, Yongin-si, South Korea. Email: yslee38@kangnam.ac.kr

Jungwon Cho, Department of Computer Education, Jeju National University, Jeju-si, South Korea. Email: jwcho@jejunu.ac.kr

computational thinking [9, 10].

Table- I: Problem-Solving Process Using Country-Specific Computational Thinking

United States of America ***CSTA & ISTE***	**United Kingdom**		**Korea**
	Concept	***Approach***	***Ministry of Science and Technology***
Data Collection	Logic	Tinkering	Data Collection
Data analysis	Algorithm	Creating	Data analysis
Data presentation	Decomposition	Debugging	Data presentation
Problem Decomposition	Pattern	Persevering	Problem Decomposition
Abstraction	Abstraction	Collaborating	Abstraction
Algorithms & procedures	-	-	Algorithms & procedures
Automation			Automation
Simulation			Simulation
Parallelization			Parallelization

The United States presented the core concepts of computational thinking in terms of computer science, mathematics, science, sociology, and linguistics. After that, the Computer Science Teachers Association (CSTA) and the International Society for Technology in Education (ISTE) defined nine kinds of operational thinking under the computational thinking concept. The Computing at School (CAS), a nonprofit in the United Kingdom, defined six concepts and five approaches to computational thinking that are emphasized in computing subjects. In Korea, with its emphasis on information education, computational thinking has emerged as a major area of competence, and it has been defined variously according to the nature of institutions or organizations. Although there have been many discussions and definitions about computational thinking, this study defined computational thinking as ability to discover and solve real-life problems with the assumption that doing so will require knowledge and concepts related to computational thinking and the use of computing power.

B. Research on Computational Thinking

In software education, computational thinking is a core competency of the curriculum. A variety of computational thinking skills are being studied to measure the overall effectiveness and performance of software education [10, 11].

Specific inspection. A specific inspection tool measures the degree of improvement in a defined sill or topic area using a computational cognitive test tool or the computational thinking tool after the software education.

Survey. Since the implementation of software education, there has been research to measure computational thinking ability in the form of self-assessments through surveys.

Observation. This method of direct before-and-after (software education) monitoring confirms the improvement of computational thinking ability of learners.

Most domestic and foreign studies have measured computational thinking through surveys. Although there are forms that directly measure computational thinking, most of the direct-evaluation questions about measurement have not been disclosed.

III. ANALYSIS OF COMPUTATIONAL THINKING TOOLS

A. Components of Computational Thinking

The Ministry of Education, the Ministry of Science and Technology (ICT) of the Republic of Korea conducted research on software education and surveyed the factors affecting the effectiveness of software education [12]. Thereafter, we developed a thinking-test tool. More than ten information education experts participated in the development of the computing tool, including field teachers and information education professors [9, 10]. The validity of the test tool was confirmed for the content presented in software education.

In this study, we analyzed the computer athletic ability test tool developed for junior high school, revised it to suit college students, and further developed and applied it. It was composed of 20 questions and was developed considering four different computational thinking abilities: analysis, design, implementation, and reasoning. The details of the developed tools are shown in Table- II [9, 13].

Table- II: Composition of the Computational Thinking Testing Tool

Software Competency	Analysis				Design		Implementation	Reasoning	
Problem-solving process based on computational thinking	Data collection	Data analysis	Data representation	Problem decomposition	Abstraction	Algorithm	Automation	Simulation	Parallelization
Number of Item	1	1	1	2	4	4	2	3	2

Analysis comprises skills in data collection, data analysis, data representation, and problem decomposition; design is abstraction and algorithm abilities; implementation is the ability to set the calculation in motion (automation); and reasoning is simulation and parallelism. The existing test tools were revised and supplemented; they consisted of five items of analysis ability, eight items of design ability, two items of implementation ability, and five items of reasoning ability, keeping the total at 20.

The characteristics of the

modified items were as follows. First, each item measured one problem-solving process based on computational thinking. Second, the 20 items were not interrelated; each item was individual and was developed as one theme. Third, because there was no link between the questions, the flow of thinking power was not measured.

B. Validation of Test Tool Items

The proposed test tool was submitted to 204 college students who were not related through software education experience. The validity of the items was verified through the validity and reliability of the items. This tool is relevant to software education contents and curricula. However, the tool was not intended to measure aspects of software, but to measure the problem-solving process based on computational thinking.

There were 60 students in the science and engineering field and 144 students in the humanities and social sciences. We used the same test tool and measurement tools for all. The descriptive statistical results are shown in Table- III.

Table- III: Descriptive Statistics According to Groups of Educated Students

Group	N	Mean	Standard Deviation
A	34	16.56	2.513
B	30	14.07	3.832
C	36	13.72	4.431
D	36	14.64	3.914
E	34	17.44	1.972
F	34	16.97	2.680
Total	204	15.57	3.623

The validity verification is to check whether a test tool is made up of contents suited to the purpose to be measured. Ours was divided into the four areas of analysis ability, design ability, implementation ability, and reasoning ability. Second, we concentrated questions requiring related abilities, although many overlapped. Problem-solving processes using computational thinking are generally a mix of analysis, design, and reasoning ability within in one student.

Table- IV shows the problem-solving process based on computational thinking in analysis capability, design ability, implementation ability and reasoning ability among the software-related competencies presented in Table- II [9, 15].

Table- IV: Exploratory Factor Analysis of Computational Thinking-Ability Testing

Item	Factor			
	Group	N	Mean	Standard Deviation
NO.2	.620	.095	-.241	-.128
NO.9	.610	-.097	.126	.103
NO.12	.580	.001	.292	.255
NO.10	.574	.169	.042	.001
NO.20	.522	.301	.074	-.089
NO.8	.464	.230	.364	.197
NO.16	.450	.172	.297	.119
NO.11	.405	.405	.061	.177
NO.6	.318	.313	.132	.226
NO.15	.089	.663	-.060	.042
NO.19	.043	.626	.151	-.128
NO.14	.063	.561	.167	.106
NO.13	.214	.332	.226	.174
NO.4	.109	-.002	.731	-.228
NO.17	.184	.388	.480	.254
NO.7	.000	.091	.479	.074
NO.18	.300	.231	.324	.134
NO.1	.212	.089	-.116	.764
NO.5	.313	.372	-.243	.527
NO.3	.303	.132	.038	.509

Estimation of the reliability coefficient is the most conservative, and Cronbach α, which estimates the minimum value of objectively, was used. The reliability of each item was verified by showing the change in the reliability coefficient (alpha if item deleted) that estimates the reliability coefficient based on only the remaining items after eliminating one item in each set in turn.

Table- V: Exploratory Factor Analysis of Computational Thinking-Ability Testing

Item	Cronbach α	Cronbach α of Factor	
NO.2	.674	.727	Factor 1 (design)
NO.9	.681		
NO.12	.810		
NO.10	.609		
NO.20	.659		
NO.8	.887		
NO.16	.933		
NO.11	.659		
NO.6	.648		
NO.15	.671	.745	Factor 2 (implementation)
NO.19	.736		
NO.14	.808		
NO.13	.783		
NO.4	.962	.895	Factor 3 (analysis)
NO.17	.898		
NO.7	.920		

Item	Cronbach α	Cronbach α of Factor	
NO.18	.812		
NO.1	.933	.833	Factor 4 (reasoning)
NO.5	.744		
NO.3	.834		
Total	.794	-	-

The internal consistency of the total and subscales shown in Table- V was 0.609–0.962 for the total and sub-factors of the computational thinking-ability test tool. Thus, the internal consistency between all the items was more than 0.60.

C. Analysis of Results

The purpose of this study was to analyze the computational thinking-ability test tool developed to measure various problem-solving processes related to computational thinking abilities. In factor analysis, factor loading shows the degree of correlation between each item and the corresponding factor—in other words, the sub-factors for competency should be tied to the same factors, depending on the factor load. Our analysis results showed that it did not measure the problem-solving process based on the computational thinking ability to be measured.

As a result of the competency and factor analysis of the computational thinking testing tool, we found that the items of the proposed testing tool could not measure the competency accurately, even though it did measure the problem-solving process based on computational thinking ability. Therefore, an item that includes all the algorithms, automation, and simulation should be developed. To develop an item that will measure the competency accurately, the problem-solving process or contents must be clarified. However, in the case of this test tool, it can be interpreted that the exact competence or the problem-solving process cannot be measured due to the hierarchical and mixed use of terms.

In terms of item composition, the placement of items according to each competency should be appropriately controlled so as not to be devoted to the measurement of specific sub-competencies. It is necessary to construct the items according to the flow of thought on the assumption that computational thinking means analyzing a problem using a computing system to understand and solve it.

However, the newly developed computational thinking test tools have limitations in that they that not only measure the problem-solving process and capacity based on fragmentary computational thinking but also enable students to solve simple questions. In other words, the related sub-questions in one large topic can be generated or expanded on in a large topic, but this will measure the computational thinking within the problem-solving process by handling the core principle or knowledge contained in the question. In terms of the content of the question, we need to develop a test tool with friendly, real-life material that does not draw on knowledge of computer science to solve the problem with students' empirical knowledge.

It is possible to solve problems through various ways of thinking such as logical thinking, and computational thinking. The abilities of students reflect the characteristic of the computational thinking that confronts real-life problems and solves them using IT. We need to be able to lead the flow of thinking.

Therefore, there is a need for a tool that can test learners' abilities step by step by creating a method and procedure—in addition to the test tool—that can solve the problem with empirical knowledge. For the step-by-step test, it is impossible to measure with the same test tool. Therefore, it is necessary to prepare measures for various cases and judge whether each item has other problem items or captures problem-solving ability. In other words, it is necessary to construct an appropriate test tool for users with the same test tool for the general test and to carry out a correlation and influence analysis related thereto in the future.

IV. CONCLUSION

The purpose of this study was to verify the validity of computational thinking ability using tools for measuring and analyzing computational thinking ability. The reliability of the items was confirmed and the content validity was secured. However, the results of the analysis of the problem-solving process based on the computational thinking ability showed that the tool was not suitable.

Through the pilot application and analysis results, we have once again grasped the difficulties of developing a tool for evaluation abilities in computational thinking. When developing tools for measuring computer thinking, it is necessary to first agree on definitions of terms (e.g., computational thinking) and to develop questions considering the problem-solving process according to those definitions. In addition, we need to measure the process of solving problems with real-life topics without directly revealing computer-related knowledge. Finally, it is necessary to confirm the purpose of the evaluation and establish the appropriateness of the test items and the degree of difficulty.

In the medium to long term, the method of checking the competency measurement related to computational thinking is performed in a personalized computer-based test (CBT) environment to identify problems and solve application cases based on basic examples. For this process, it is necessary to develop a computational thinking measurement tool suitable for users through various problem development and evaluation system building, and to conduct efficient testing suitable for the level of users.

REFERENCES

1. Grover S, Pea R, "Computational thinking in K–12: A review of the state of the field" Educational researcher, 42(1), 38-43, 2013. https://doi.org/10.3102/0013189X12463051
2. Wing JM, "Computational thinking" Communications of the ACM, 49(3), 33-35, 2006.
3. Grover, Shuchi. "Systems of Assessments for deeper learning of computational thinking in K-12" In Proceedings of the 2015 annual meeting of the American educational research association (pp. 15-20), 2015.
4. Zhong B, Wang Q, Chen J, Li Y., "An exploration of three-dimensional

integrated assessment for computational thinking" Journal of Educational Computing Research, 53(4), 562-590, 2016. http://dx.doi.org/10.1177/0735633115608444

5. Relkin E. "Assessing Young Children's Computational Thinking Abilities," PhD Thesis, Tufts University, 2018.
6. Romero M, Lepage A, Lille B., "Computational thinking development through creative programming in higher education" International Journal of Educational Technology in Higher Education, 14(1), 42, 2017. https://doi.org/10.1186/s41239-017-0080-z
7. González MR, "Computational thinking test: Design guidelines and content validation" In Proceedings of EDULEARN15 conference 2015 (pp. 2436-2444), 2015.
8. Brennan K, Resnick M., "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 2012 Apr 13 (Vol. 1, p. 25), 2012.
9. Youngseok Lee, Jungwon Cho, "Factor Analysis of Computational Thinking for Software Education Based on Problem-Solving Learning" International Journal of Pure and Applied Mathematics, 120(6), 4953-4967, 2018. https://acadpubl.eu/hub/2018-120-6/4/359.pdf
10. Kwon J, Kim J., "A Study on the Design and Effect of Computational Thinking and Software Education" Thinking, 2(3), 4, 2018. http://doi.org/10.3837/tiis.2018.08.028
11. Kim, J. A., & Ko, D. Y., "Survey of On-Line & Block Programming Language-Scratch: On Perspective of Educational Achievements" In Advances in Computer Science and Ubiquitous Computing (pp. 35-40). Springer, Singapore, 2017. https://doi.org/10.1007/978-981-10-7605-3_6
12. Czerkawski, B. C., & Lyman, E. W., "Exploring issues about computational thinking in higher education" TechTrends, 59(2), 57-65, 2015. https://doi.org/10.1007/s11528-015-0840-3
13. Chen G, Shen J, Barth-Cohen L, Jiang S, Huang X, Eltoukhy M., "Assessing elementary students' computational thinking in everyday reasoning and robotics programming" Computers & Education, 109, 162-175, 2017. https://doi.org/10.1016/j.compedu.2017.03.001
14. Kline RB, Principles and practice of structural equation modeling, Guilford publications, 2015.

AUTHORS PROFILE

Youngseok Lee is a Professor at the KNU College of Liberal Arts and Sciences, Kangnam University. He is studying about Computer education, Data science, Intelligent information system, Liberal education and so on.

JungwonCho is a Professor at the Department of Computer Education, Jeju National University. He is the director of 'Research Institute of Education Science' and 'Center of Intelligent Computing Education' in Jeju National University. He is studying about Computing education, Intelligent information ethics, Intelligent information system and so on.