University Students' Attitude and Anxiety Towards Computer Science Education through Computational Thinking

Youngseok Lee[1], Jungwon Cho[*2]

[1]Professor, KNU College of Liberal Arts and Sciences, Kangnam University, 40 Gangnam-ro, Giheung-gu, Yongin-si, Gyeonggi-do, 16979, South Korea
[*2]Professor, Department of Computer Education, Jeju National University, 102 Jejudaehak-ro, Jeju-si, Jeju-do 63243, South Korea
yslee38@kangnam.ac.kr[1], jwcho@jejunu.ac.kr[*2]
Corresponding author[*] : Jungwon Cho.

Article History:Received:11 november 2020; Accepted: 27 December 2020; Published online: 05 April 2021

Abstract: Background/Objectives: In the 21st century, communication and collaboration between people is an important element of talent. As artificial intelligence (AI), the cutting edge of computer science, develops, AI and collaboration will become important in the near future.
Methods/Statistical analysis: To achieve this, it is necessary to understand how artificial AI based on computer science works, and how problem-based programming education is effective in computer science education. In this study, 177 college students who received programming education focused on problem-solving learning were identified with computational thinking (CT) at the beginning of the semester, and their satisfaction and post-education satisfaction survey showed that their attitudes and interests influenced their education.
Findings: To pretest the learners, they were diagnosed using a measurement sheet. The learners' current knowledge statuses were checked, and the correlation between the evaluation results, based on what was taught according to the problem-solving learning technique, was analyzed according to the proposed method. The analysis of the group average score of the learners showed that the learning effect was significant. The results of the measures of the students' CT at the beginning of the semester were correlated with problem-solving learning, teaching method, lecture satisfaction, and other environmental factors. The ability to solve a variety of problems using CT will become increasingly important, so if students seek to improve their satisfaction with problem-solving learning techniques for computer science education, it will be possible for universities to develop convergence talent more efficiently.
Improvements/Applications: if you pursue a problem-solving learning technique and a way to improve students' satisfaction, it will help students improve their problem-solving skills. If the method of deriving and improving computational thinking ability in this paper is applied to computer education, it will induce student interest, thereby increasing the learning effect.

Keywords: Computational thinking, Problem-solving learning, Learning techniques, Learning satisfaction, Educational programming language

1. Introduction

Efforts to create new values by integrating computing technologies with various fields have been emphasized in modern society. Software is expected to be the center of innovation among various information and communication technologies [1]. Computer science education, including the ability to design and build software, is now a universal social education that must be made available to anyone, regardless of their field. Many educational institutes are attempting to offer comprehensive software education to improve problem-solving skills using computing technologies, including to non-computer-science majors [1].The general trajectory of computer science education is to obtain ideas for solving problems based on computational thinking (CT) and to cultivate creativity [2]. Through general computer science education, computer non-specialists can use computing technology in their major fields to shape ideas. CT ability for this needs to be computer-based thinking, so it can be developed and improved through software education based on educational programming language [1, 2].In reality, non-computer-science majors students are unfamiliar with computers and have difficulty grasping software education. It is not easy to attract these students to software and improve their problem-solving skills using CT and computing skills [3]. To develop problem-solving abilities, students are presented with real-world problems, and through exploration, learning is conducted in a series of processes to implement problem-solving solutions using educational programming languages. These languages should be thus configured to solve real-world problems [1, 3].The method of conducting problem-solving learning begins with introducing students to basic programming content and then presenting them. Students then identify what problems need to be solved, if they can be solved, and if so, what is required to solve them [1]. If students fail to understand the problem, the process returns to the basic understanding stage. If the problem can be identified and judged, learning contents or additional matters to be solved are derived. Each problem is placed in consideration of lecture objectives, problem content, and time. At the beginning of the lecture, the problem is presented in a very simple manner so that it can be solved easily and enables students to continuously reach learning goals. Students gradually learn through the process of repeated steps. The basic syllabus is similar to

[*]**Corresponding author:** Jungwon Cho
Professor, Department of Computer Education, Jeju National University, 102 Jejudaehak-ro, Jeju-si, Jeju-do 63243, South Korea . jwcho@jejunu.ac.kr

programming education for proposed computer non-majors, but contents of instruction are modified to apply the problem-solving technique.Currently, most colleges and universities offer software education in the form of liberal arts, but software education goals and methods for non-major students are not optimized and vary considerably depending on the situations of schools and students [4]. Typical software education is a form of programming that uses a programming language to enter commands on a computer and think about the results. There are two types of software education which can be divided into professional programming education for the software-related industry, and universal programming education to improve CT and problem solving [4]. The basic goal of software education is to design and complete thoughts on a computer, resolve problems in the process of solving a given problem, and repeat and combine the process of finding a solution. Various abilities such as logical and/or procedural thinking ability for programming, information processing using computers, and problem-solving ability are improved. Computer non-major students face a host of challenges because they need to use computer skills to solve problems from a completely new perspective. In order to solve these difficulties, research into software education has been actively conducted. In the case study of non-major programming learning, a study that analyzes the difficulty in education of CT for non-majors is typical. In the case of university software education, the study was conducted by applying the class based on problem-solving learning to measure and improve CT ability in the form of knowledge representation [5].Problem-solving learning is said to improve the ability to recognize the problem situation and solve the problem in the process of deriving such a solution. Problem-solving learning differs from traditional teaching methods in many areas, and process-based rather than result-based assessment is carried out. Process-based evaluation focuses on the learning process, i.e., 'how did you get to know' as opposed to 'what did you learn'. Attempting such problem-solving learning can induce learners' interest and imagination, and lead to changes in learners' attitudes [1]. Through this process, we will analyze how attitudes and interests of students are related to actual CT ability and analyze the relationship between satisfaction of the problem-solving learning process and CT ability and use it in various applications of future problem-solving learning.

2. Materials and Methods

2.1. Research Background

In this study, we developed a CT assessment tool and applied it to diagnose the initial state of students. We prepared a variety of problem-oriented learning cases and provided computer education based on problem-solving learning. Based on the concept of CT, the model of learning strategy was applied in order to analyze the CT elements and framework. At the university level, in order to cultivate CT ability in the study of software education, the teaching and learning model for the improvement of CT ability was studied and the project learning model was proposed [6]. Existing studies show that substantial research related to CT and software education has been conducted [7, 8].Analysis confirms that while Python text coding language is highly applicable in CT, non-computer major students find it difficult to operate and thus serves as an entry barrier to programming education. We found that problem-solving learning extracted by analyzing CT ability and learner's attitude through problem-solving learning based software education can raise students' interest and improve their CT in order to facilitate software education in universities [1, 9, 10]. This study aims to analyze correlations and effects of four factors—problem-solving learning, teaching method, lecture satisfaction, and other environmental factors, pedagogical methods, lectures, and other environmental factors—on attitudes towards computer science education and CT.

A CT test was developed for college students, and their attitudes and satisfaction levels obtained through surveys were used as measuring tools [1]. The CT questionnaire consisted of 20 questions and the satisfaction questionnaire consisted of 23 questions. The questionnaire consisted of four sub-factors: problem-solving learning, teaching methods, lecture satisfaction, and other environments. The CT test awarded one point for every correct answer for each question and 0 points if it was wrong. The lecture attitude and satisfaction survey showed that respondents expressed opinions about each item using a 5-point Likert scale. The Cronbach α coefficient was .928 for estimating the internal consistency reliability that analyzed learners' responses to items in this test.Development of the questionnaire occurred in steps: previous research was analyzed and a questionnaire was created, validated, and administered online, following which, results were analyzed [1]. Based on existing previous research, the questionnaire was developed by modifying and supplementing the previous questionnaire [1, 8, 11, 12]. In order to ensure the questionnaire's validity, it was verified in consultation with two computer education professors, two computer science professors, and two liberal arts professors. Efforts were made to avoid technical jargon as much as possible while organizing the questionnaire and its items to make it easier for respondents.

2.2. Research Methodology

We surveyed 177 university students in Gyeonggi-do, Korea using questionnaires on problem-solving learning satisfaction, teaching methods, lecture satisfaction, and students' grades. Correlation analysis was performed. The actual problem-solving learning examples used in lectures conducted in 2019 are described

below.

◎ Example : Temperature and Precipitation Histogram Production Program

♣ Lesson objectives: The goal of the program is to represent temperature and precipitation in three steps: (1) The weather office collects temperature and precipitation data or receives an input statement with this information. (2) Graphs are produced by converting algorithms that are easy to understand. (3) A graphical representation using Python turtle expresses the desired form.

♣ Problem Description and Progress : Average monthly rainfall collected by the Meteorological Administration is [67, 58, 49, 41, 53, 36, 24, 22, 20, 28, 52, 43] and should be treated as a list. Write a program that depicts monthly average rainfall above 60 in red, 51 to 60 in yellow, 31 to 50 in green, and 1 to 30 in blue. Graphs are recommended in the form of bar graphs. Other forms are acceptable.

This is a continuous course that runs for a duration of ten to fourteen weeks, with two weeks of list processing and two weeks of graph grit processing after mid-term exams. An additional week was provided to help resolve the problem.

In this paper, we surveyed 177 university students in Korea using problem sampling to determine their satisfaction with problem-solving learning, teaching methods, lecture satisfaction and their own grades. The CT test and satisfaction survey were measured by self-report method through questionnaire. A total of six classrooms were surveyed and 98% of 177 respondents completed the survey. Frequency analysis was performed to confirm the general characteristics of the study subjects and the surveyed subjects for descriptive statistical analysis are presented in Table 1.

Table 1. Distribution of the research group according to class, gender and year grade.

			N	%			N	%
Class	Social Welfare (Specialization)	2	29	16.4	Gender	Female	110	62.1
	Non-science & Engineering	4	28	15.8		Male	67	37.9
		6	27	15.3	Year grade	1st	159	89.8
		11	31	17.5		2nd	4	2.3
	Science & engineering	12	32	18.1		3rd	7	4.0
		14	30	16.9		4th	7	4.0
Total			177	100				

The demographic break-up of the sample was as follows: Of the 177 students surveyed, there 67 or 37.9% males and 110 or 62.1% were females. In terms of distribution by class, 89.8% or 159 students were in the first grade (or freshmen), 2.3% were in the second grade, 4.0% were in the third grade, and 4.0% were in the fourth grade. In computer science education, there are a majority of first grade students due to the nature of liberal arts. Examining the class level, the science and engineering division is divided into 12 and 14. Diagnostic results were obtained; class 4, 6, and 11 were composed of students of Global Studies, Music Studies, Korean Language, and Content Studies.

3. Results and Discussion

A CT test was administered at the beginning of the semester, and a questionnaire relating to satisfaction was conducted before the end of the semester. The questionnaire was analyzed and processed using SPSS Statistics 18.0 program and significance level was verified at $p<.05$. The results for the validity of questions are shown in Table 2. KMO and Bartlett's test scores were high at .928, and the significance probability was .000, which ensured the validity of the question.

Table 2. KMO and Bartlett's Test.

Kaiser-Meyer-Olkin Measure of Sampling Adequacy.		.928
Bartlett's Test of Sphericity	Approx. Chi-Square	2360.073
	df	253
	Sig.	.000

Table 3 shows results of the ANOVA test based on results of CT ability to analyze characteristics of 177 students in six classes. Since engineering students were divided into two classes, the homogeneity of each class

in the result of the CT test were not the same.

Table 3. The results of ANOVA analysis.

Item	Class	N	Mean	Standard deviation	Sum of squares	df	F	Sig.
Sum of Computational Thinking	2	29	16.72	2.576	.478	5	6.881	.000
	4	28	13.75	3.903	.738			
	6	27	14.04	4.407	.848			
	11	31	14.97	3.656	.657			
	12	32	17.59	1.915	.339			
	14	30	16.77	2.979	.544			
Total		177	15.70	3.576	.269			

The average grades of science and engineering students (12, 14) were significantly higher than those of non-engineering students, showing a difference of 1.75 points to 3.84 points. This shows that there is a difference in students' basic CT ability. When comparing the quarter average (13.75) and the six quarter average (14.04), the mean t-value of the two non-processing segments (4, 6) showed no significant difference (t=-.256, p=.799) at the $p<0.05$ level. In other words, homogeneity between groups of non-major students was verified. Similarly, the mean of the two science and engineering divisions (12, 14) also showed no significant difference (t=1.309, p=.196) at the $p<0.05$ level. In other words, homogeneity among groups of science and engineering students was also secured. Homogeneity of each class was verified according to characteristics of each series. Results of analyzing the correlation between the results of problem-solving learning, teaching method, lecture satisfaction, and others (attendance, environment, etc.) and CT ability diagnosis according to the factor analysis of the questionnaire are shown in Table 4.

Table 4. The results of correlation analysis.

	Problem-solving learning	Teaching method	Lecture satisfaction	Others	Computational thinking ability Total
Problem-solving learning	1				
Teaching method	.745**	1			
Lecture satisfaction	.675**	.604**	1		
Other environment	.293**	.350**	.190*	1	
Computational thinking ability Total	.439**	.319**	.434**	.168*	1

****. Correlation is significant at the 0.01 level(2-tailed)**
***. Correlation is significant at the 0.05 level(2-tailed)**

The correlation between CT test scores and problem-solving learning was high, at 0.439. The correlation with teaching method was .319, and the correlation with lecture satisfaction was .434. Attendance environment was correlated with .168. Problem-solving learning showed a very high correlation with teaching method,

at .745, and satisfaction with lectures at .675, and correlation with other environments at .293. According to these results, students' satisfaction with the result of software education based on problem-solving based on results of CT ability is high and correlated.

Table 5 depicts results of the multiple regression analysis to determine the impact of the correlations.

Table 5. The results of multiple regression analysis.

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	beta		
1	(Constant)	7.344	1.907		3.851	.000
	Class	.048	.056	.060	.8561	.393
	Problem-solving learning	1.199	.465	.292	2.580	.011
	Teaching method	-.507	.637	-.084	-.796	.427
	Lecture satisfaction learning	1.375	.483	.265	2.847	.005
	Other environment	.252	.309	.059	.815	.416

As the correlation analysis showed meaningful results, problem-solving learning ($t=2.580$, $p<.05$) and lecture satisfaction ($t=2.847$, $p<.05$) had significant effects on actual students' academic attitudes and satisfaction. There were no significant results obtained from analyzing both science and non-science students due to differences in academic content and preferences.

4. Conclusions

The society of the future will need talented people who can solve problems based on CT. Therefore, in this paper, the solution-oriented Python-based software education is based on problem-solving and finds a solution that is suitable for the proposed problem situation and learns and utilizes it through mutual cooperation. Robots and AI are expected to cause a great social transformation in the near future. Human resources with excellent CT and problem-solving skills will be needed to create new values by combining computing technologies with various fields based on humanities. Computer education as a general programming education should be configured to make the problem-solving experience using computing technology. Limitations of programming language education, such as difficulties associated with learning the languages, are yet to be overcome.

As a result, that there was a correlation between the satisfaction of problem-solving learning and grades, attendance, and actual academic scores. In particular, a high correlation between academic performance and academic grades was noticed after the course, and satisfaction levels of academic grades and problem-solving learning. This correlation analysis confirmed that students' satisfaction had significant effects on their academic performance. This paper analyzed the relationship between college students' attitudes and interests in computer science education and CT. In order to conduct the CT test, we presented a problem-based situation related to daily life, developed a test paper to solve the problem, and applied it to college liberal arts classes. Satisfaction was confirmed by a post-test. Based on Python, the most widely used educational programming language at universities, software education was designed in a problem-oriented form, and students were able to solve their own problems by learning and utilizing solutions for presented problem situations. As a result of analyzing CT results, there was a significant difference according to the overall group characteristics, but no significant results for group characteristics according to series characteristics.

The correlation between the CT test result and problem-solving learning, teaching method, and lecture satisfaction showed significant correlation. The result of CT correlated with the attendance environment. In particular, satisfaction with problem-solving learning was highly correlated with pedagogy and lecture satisfaction, and significantly different with other environments. According to results of the computerized thinking ability diagnosis, lessons were found to have a significant effect on students' academic attitudes and satisfaction, especially factors such as problem-solving learning and lecture satisfaction. It was concluded that students' satisfaction is high and is correlates with software education based on problem-solving in relation to CT ability.

If software education is conducted for more than year to predict the degree of change in CT, it is expected that student satisfaction will have a significant impact on actual academic performance. Therefore, if we construct a problem-solving learning form suitable for university software education, analyze various cases and

share results, we will be able to proceed with desirable software education. We will analyze the relationship between the problem-solving learning process, attendance, and academic achievements performed by actual students, and subsequently investigate methods to improve CT, problem-solving abilities, and applications of real-life problems in each non-major area. The study intends to verify effectiveness after conducting software training on different subjects.

References

1. Lee, Y., & Cho, J. (2020). Knowledge representation for computational thinking using knowledge discovery computing. *Information Technology and Management* 21(1), 15-28.
2. Grover, S., & Pea, R. (2018). Computational Thinking: A competency whose time has come. *Computer science education: Perspectives on teaching and learning in school* 19.
3. González, M. R. (2015). Computational thinking test: Design guidelines and content validation. In Proceedings of EDULEARN15 conference (pp. 2436-2444).
4. Durak, H. Y., Yilmaz, F. G. K., & Yilmaz, R. (2019). Computational Thinking, Programming Self-Efficacy, Problem Solving and Experiences in the Programming Process Conducted with Robotic Activities. *Contemporary Educational Technology* 10(2), 173-197.
5. Basogain, X., Olabe, M. Á., Olabe, J. C., & Rico, M. J. (2018). Computational Thinking in pre-university Blended Learning classrooms. *Computers in Human Behavior* 80, 412-419.
6. Wang, M., & feng Wang, Y. (2016). A Study on Computer Teaching Based on Computational Thinking. *International Journal of Emerging Technologies in Learning (iJET)* 11(12), 72-74.
7. Broks, A. (2016). Systems theory of systems thinking: General and particular within modern science and technology education. *Journal of Baltic Science Education* 15(4), 408-410.
8. Kong, S. C., Chiu, M. M., & Lai, M. (2018). A study of primary school students' interest, collaboration attitude, and programming empowerment in computational thinking education. *Computers & Education* 127, 178-189.
9. Burbaitė, R., Drąsutė, V., & Štuikys, V. (2018, April). Integration of computational thinking skills in STEM-driven computer science education. In 2018 IEEE Global Engineering Education Conference (EDUCON) (pp. 1824-1832). IEEE.
10. Kwon, J., & Kim, J. (2018). A Study on the Design and Effect of Computational Thinking and Software Education. *KSII Transactions on Internet & Information Systems* 12(8).
11. Romero, M., Lepage, A., & Lille, B. (2017). Computational thinking development through creative programming in higher education. *International Journal of Educational Technology in Higher Education* 14(1), 42.
12. Topalli, D., & Cagiltay, N. E. (2018). Improving programming skills in engineering education through problem-based game projects with Scratch. *Computers & Education* 120, 64-74.