A Study on the Design and Effect of Computational Thinking and Software Education

Jungin Kwon[1] and Jaehyoun Kim[2]
[1] College of General Studies, Sangmyung University
Hongimun 2-Gil, Jongno-Gu, Seoul - ROK
[e-mail: jikwon@smu.ac.kr]
[2] Computer Education, Sungkyunkwan University
Sungkyunkwan-Ro, Jongno-Gu, Seoul - ROK
[e-mail: jaekim@skku.edu]
*Corresponding author: Jaehyoun Kim

Received February 28, 2018; revised July 4, 2018; accepted July 28, 2018; published August 31, 2018

Abstract

The software centered world following the fourth industrial revolution is rapidly approaching us. Countries around the world attach importance to software's ability as one of the key elements for training future human resources. In order to train software centered human resources, each university has designated Software Education as an essential curriculum for not only major but also non-majors. In the past Software Education was an education for a major, but recent Software Education was changed to the essential education that is necessary for all living in the software centered world. In the past the curriculum was focused on software development and implementation-oriented education, but recent curriculum emphasizes sequential arranging and thinking of problem solving. In order to reflect trends in recent Software Education in detail, we integrate Software Education with major concept of Computational Thinking. In this paper, we analyzed the effect of the main concept of Computational Thinking on Software Education for non-majored learners who received Software Education based on Computational Thinking (here refers to learners who major in humanities, social sciences and arts). In addition, research models of satisfaction, self-efficacy, and occupational change was established as the elements of Software Education, and it was found that there was a relation between Computational Thinking and Software Education.

Keywords: Computationl Thinking, Software Education, Algorithms and Procedures, Learner Satisfaction, Self-Efficacy

A preliminary version of this paper was presented at ICONI 2017, and was selected as an outstanding paper.
This research was supported by the MIST(Ministry of Science and ICT), Korea, under the National Program for Excellence in SW supervised by the IITP(Institute for Information & communications Technology Promotion)"(2015-0-00914)

1. Introduction

As the interest in the future society of the knowledge base is heightened, the software centered world is rapidly approaching us due to the fourth industrial revolution. Countries around the world attach to software capability as one of the key elements of training future human resources [1]. Software capability refers to software centered communication ability, problem solving ability, information utilization technology, and is a core ability for knowledge based training of training future human resources [1]. In accordance with the change of the ability of this education, the problem solving ability is improved based on Software Education in Korea, USA, Japan, Israel, India and UK etc [1].

In recent years Software Education has proposed an Thinking centered curriculum that can learn the principles of computer science. In the past Software Education was an education for a major, but recent Software Education was changed to the essential education that is necessary for all living in the software centered world. At each university, Software Education has become a necessary education course for not only major but also non-majors. In the past the curriculum was focused on software development and implementation-oriented education, but recent curriculum emphasizes sequential arranging and thinking of problem solving. In order to expand the scope of education and access new Software Education, we are pursuing a new change in the education field which moved away from existing software curriculum and methodology.

In order to reflect trends in recent Software Education in education, we combine main concept of Computational Thinking and Software Education which is generalized by Wing(2017) [2, 3, 4]. Computational Thinking is a process of thinking about problem solving ability. It suggests a new direction of Software Education, claiming that everyone should accustomed and learn to read, write, and calculate as well as learn Computational Thinking [2, 3, 4].

Therefore, Computational Thinking and curriculum of Software Education should be convergence in order to constantly discovering future human resources. In order to reflect this, each educational institution has recently introduced a new vision of Software Education and is attempting a new change of creative human resources required by future society. As a part of this change, the methodology of Software Education based on Computational Thinking is attracting attention. Computational Thinking based Software Education should utilize a wide range of knowledge from various fields of study and fusion. It should also improve the ability to solve complex problems. Computational Thinking based Software Education means comprehensive education that can break down the division of disciplines and develop integrated insight.

The purpose of this study is to analyze the effect of the main concept of Computational Thinking for non-majors who received Software Education based on Computational Thinking majoring in humanities, social sciences, and arts.

In addition, we set up a research model for the satisfaction of Software Education, self-efficacy, and change of occupation of non-majors and we found that there is a relation between Computational Thinking and Software Education. We then conducted a hierarchical regression analysis that influenced Software Education among the core elements of Computational Thinking. Based on the results, we propose an educational model of Software Education based on Computational Thinking for non-majors.

2. Theoretical Background

2.1 Related Research on Computational Thinking

Computational Thinking was first introduced to us in 1996 when Seymour Papert was first used as an approach to creating geometric ideas and later became known by Wing. Wing(2017) claim that "Computational Thinking should be defined as thinking for problem solving, system design, and understanding of human behavior according to the basic concepts and principles of computer science. Everyone should learn and accustomed Computational Thinking as well as learn to read, write, and calculate" [2, 3, 4].

CSTA (Computer Science Teachers Association, 2011) defined Computational Thinking as a process of problem solving [5]. Computational Thinking includes the following characteristics [5].

First, we define the problem so that we can use the computer and other tools to solve the problem. Second, the data necessary for problem solving are logically composed and analyzed. Third, data is re-expressed through abstractions such as models and simulations. Fourth, we automate the solution based on procedural steps and algorithmic thinking. Fifth, we identified analyzed and implemented all possible solutions to select the most efficient of problem solving procedures. Sixth, we generalize the solution to various problems.

CSTA and ISTE(International Society for Technology in Education) presented Data Collection, Data Analysis, Data Representation, Problem Decomposition, Abstraction, Algorithms & Procedures, Automation, Simulation, Parallelization as a key element of Computational Thinking based on the results of David Barr, John Harrison, & Leslie Conery (2011) [4, 5, 6].

The nine core concepts of Computational Thinking are shown in **Table 1** [4, 5, 6].

Table 1. Computational Thinking Main Concepts

Concept	Definition
Data Collection	Problem understanding, analysis and collect data based on analysis to solve the problem
Data Analysis	Carefully sorting and analyzing the data collected and data provided in the problem
Data Representation	Express data in problem using graphs, charts, words and images
Problem Decomposition	Dividing and analyzing the problem to solve the problem
Abstraction	Defining the main concepts to reduce the complexity of the problem
Algorithm and Procedures	Expressing the steps required to solve the problem until now
Automation	Creating an algorithm of the solution procedure for a computing machine to carry it out
Simulation	Creating an experimental model to solve the problem
Parallelization	Coming up with a common objective to solve a problem

2.2 Computational Thinking based Software Education

In order to spread software society, software centered universities are selected and the training of software talent with problem solving ability is strengthened [8]. It also includes software basic education for non-majors and encourages the development of software curriculum design and support programs centering on non-majors. However, it is somewhat unreasonable to think that the curriculum or the curriculum provided so far is the development of the curriculum or the curriculum for the non-majors. In addition, there is no education content in the Software Education course based on Computational Thinking, which has recently been recognized as important.

Therefore, this study aims to design the development of curriculum and curriculum for Software Education based on Computational Thinking, which is focused on non-majors. The design goals of the Computational Thinking based Software Education course for non-majors are as follows [7, 8, 9].

First, it aims to understand the principles of computer science and recognize the importance of software by real life and academic field. Understands the principles of computer science and recognizes the importance of software based on practical examples of real life computer science principles applied [9].

Second, it reflects the characteristics of major field of non-majors. The course focuses on improving the problem solving ability based on Computational Thinking and focuses on the Software Education that reflects the characteristics of each major so that non-majors can solve the problems through software in each major field [8, 9].

Third, we aim to educate students with procedural problem solving skills based on Computational Thinking. Non-majors are not coding programs to solve problems. It is to improve the ability that the process of solving problems based on their prior knowledge, experience, and thinking. This suggests a practical methodology to implement learner 's sequential thinking directly with software [10].

Fourth, programming can be applied for efficient selection among problem solving methods. Making software programming language to be applied in order to improve the efficiency of sequential problem solving [11].

3. Research Method and Procedure

3.1 Course Development

After the design of Software Education based on Computational Thinking for non-majors, we developed two subjects: ‘Computational Thinking and Software Coding’ and ‘Problem Solving and Algorithm’.

The development period of the curriculum is from October 2015 to July 2016 and the overall schedule for the curriculum development is shown in **Table 2**.

Table 2. Course Development Schedule

Time	Schedule
2015. 10.	Course development decision
2015. 11.	Prepare the first data and collect expert opinions
2015. 12.	Conducting expert reviews after preparing secondary data
2016. 1.	Use it as teaching materials for preparatory college
2016. 2.	After revision expert evaluation conducted

3.1.1 Development of 'Computational Thinking and Software Coding' Course

'Computational Thinking and Software Coding' is a curriculum to raise awareness of the principles of computer science, trends in the latest IT technologies and importance of information society for non-majoring students who are new to software. The main concept of Computational Thinking was used as a sequential step of the teaching and learning process, and focused on improving the learners logical thinking ability [12, 13]. The main concept of Computational Thinking was used as a sequential step of the teaching and learning process and focused on improving the learners logical thinking ability. In addition, the learner 's thinking for problem solving was designed as a core concept of Computational Thinking, and it was constructed to be implemented as a block-based programming language to implement it [12, 13].

'Computational Thinking and Software Coding' is designed to learn the basic concepts of software through the theoretical education to understand the importance of computer science and Software Education and the principles of computer operation in the beginning of learning for Software Education of the non-majors. In the middle, the problem solving procedure and application process based on the concept were constructed.

The curriculum design is shown in **Table 3**.

3.1.2 Development of 'Problem Solving and Algorithm' Course

'Problem Solving and Algorithms' include three educational contents. First, learners are not simply coding the program, but solving the problems presented in the story. Second, learners make problem solving procedural based on experience, knowledge and thinking. Third, learners can choose a more efficient methodology [14].

The experience of presenting problem solving procedures in sequence is a great help to improve the learner's logical thinking ability. Therefore, problem-solving procedure is expressed in pseudo-code and then configured to be able to be algorithms and learn how to implement more efficient algorithms [15].

Python languages were used to implement algorithms as tools. 'Problem Solving and Algorithm' is designed to learn how to select the problem solving procedures learned in 'Computational Thinking and Software Coding' more efficiently.

The curriculum design is shown in **Table 3** [8].

Table 3. Design of Development Subject

Main concepts	Subject name	Theoretical education	Practical training
Understanding the principles of Computer Science	Computational Thinking and Software Coding	Computer science and Software education	unplugged activity
		Principle of computer operation and data representation	Introduction and usage of Entry
		The development of the latest IT technology	Entry basic example
Concept and Application of Computational Thinking		The concept of Computing Thinking	Sequence / Event / Signal Example
		Problem Solving based on Computational Thinking	Entry Application Example
Procedures for Problem Solving		Understanding Software principles	Variables and example of lists
		Variables and Lists	Operator example
		Understanding Logic Operations	Selection example
		Understanding the control statements	Loop logic example 1
		Understanding Loop Logic	Loop logic example 2
		Differentiating the differences between loops	Function Example
		Understanding the concept of Functions	Search and Sort
		Understanding Search and Sort	Advanced Example
	Problem Solving and Algorithm	Necessity and process of Problem Solving	Concept of Physical Computing
		Step-by-step procedure for Problem Solving	Use of physical Computing
		Concept and purpose of data structure	Python data type
Various Problem Solving Methods		Solution Techniques for Puzzle Problems	Python Data Handling
		Techniques for solving logical problems	Program flow control
		Solution Techniques for Computer Science Problems	Working with graphics
Choosing Efficient Problem Solving Methods		Data sorting Algorithm	Program flow control
		Data search Algorithm	Functions
		Brute Force Algorithm	Sorting
		Divide and conquer Algorithm	Object-Oriented Program Concepts

3.2 Study Subjects and Model Setting

This study is aimed at 205 students attending 'Computational Thinking and Software Coding' and 'Problem Solving and Algorithm' courses at S university in 2017.

We investigated self-efficacy, satisfaction with educational effectiveness and change of occupation based on Computational Thinking after learning ‘Computational Thinking and Software Coding’ and ‘Problem Solving and Algorithm’.

A total of 40 questionnaires were divided on a 6 points scale. The effects of software coding education on learning satisfaction, self-efficacy, and change of occupation were investigated based on nine core elements of Computational Thinking.

The research model of this study is shown in **Fig. 1**.

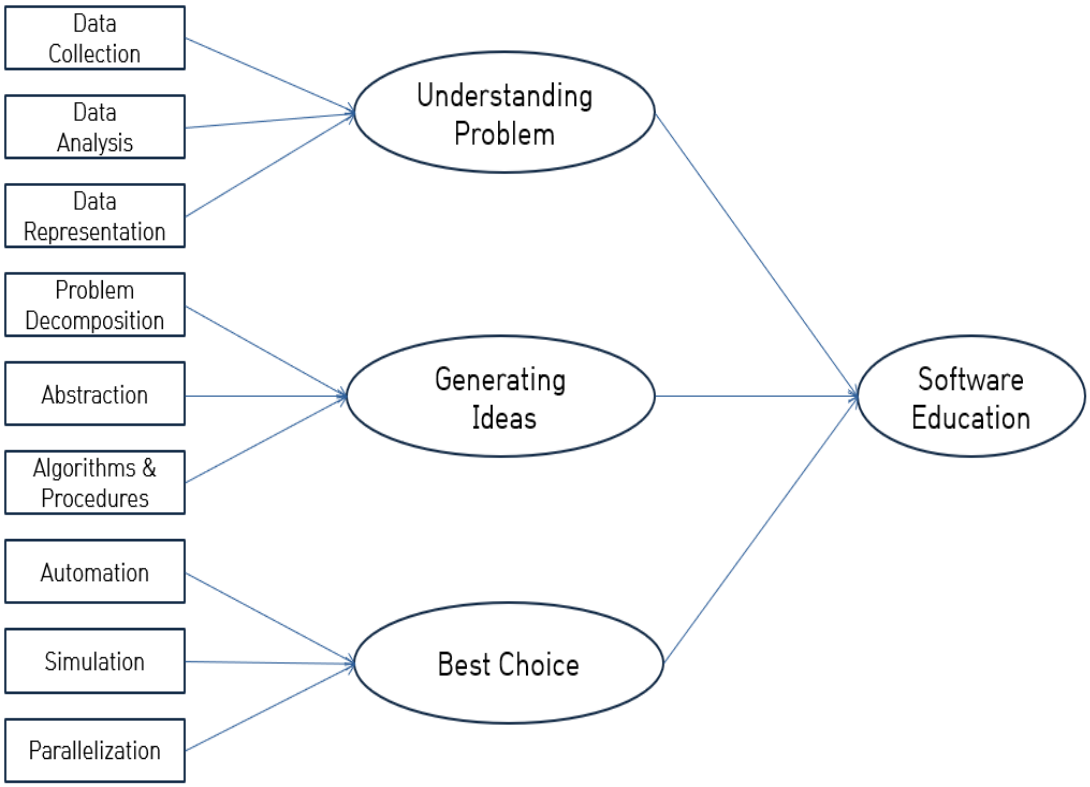

Fig. 1. Research Model

3.3 Research Result

In this study, the Cronbach α coefficient was used to measure the internal consistency relia bility of each factor. The measurement reference value was set to 0.6 or more. **Table 4** is th e result of analysis about each average, standard deviation and credibility.

Table 4. Reliability Verification Results

Concept variable	Average	Standard Deviation	N	Cronbach α
Data Collection	1.9928	.55892	205	0.674
Data Analysis	1.8342	.51558	205	0.649
Data Representation	1.8886	.66710	205	0.790
Problem Decomposition	1.7899	.67294	205	0.696
Abstraction	1.8995	.55938	205	0.791
Algorithms & Procedures	1.9837	.57317	205	0.850
Automation	1.9644	.63127	205	0.691
Simulation	1.8916	.61683	205	0.873
Parallelization	1.7779	.61438	205	0.714
Understanding Problem	1.9029	.60107	205	0.618
Generating Ideas	1.8342	.51557	205	0.757
Best Choice	2.0659	.56845	205	0.749
Software Education	1.9837	.55382	205	0.603

Automation variables have very weak correlation with most variables or show no correlation. On the other hand, the Software Education variables are correlated with most variables and the degree of correlation is very high. The correlation coefficient was highest 0.866 with Algorithms & Procedures and Software Education. The correlation between Abstraction and Software Education was 0.691.

Thus, Algorithms & Procedures, Abstraction can be considered to have a strong correlation with Software Education. Correlation analysis result is shown in **Table 5**.

Table 5. Correlation Coefficients between Variables

P.C.C.= Pearson Correlation Coefficient / P-value = Means both sides / n=205

		Data Collection	Data Analysis	Data Representation	Problem Decomposition	Abstraction	Algorithms & Procedures	Automation	Simulation	Parallelization	Understanding Problem	Generating Ideas	Best Choice	SW Education
Data Collection	P.C.C.	1												
	P-value													
Data Analysis	P.C.C.	.445''	1											
	P-value	.000												
Data Representation	P.C.C.	.341''	.319''	1										
	P-value	.000	.000											
Problem Decomposition	P.C.C.	.331''	.481''	.305''	1									
	P-value	.000	.000	.000										
Abstraction	P.C.C.	.420''	.345''	.356''	.252''	1								
	P-value	.000	.000	.000	.000									
Algorithms & Procedures	P.C.C.	.748''	.370''	.247''	.302''	.329''	1							
	P-value	.000	.000	.000	.000	.000								
Automation	P.C.C.	.023	.028	.108	.046	.016	.002	1						
	P-value	.662	.591	.038	.376	.757	.977							
Simulation	P.C.C.	.853''	.383''	.267''	.275''	.613''	.406''	.016	1					
	P-value	.000	.000	.000	.000	.000	.000	.759						
Parallelization	P.C.C.	.657''	.347''	.228''	.240''	.416''	.607''	.025	.671''	1				
	P-value	.000	.000	.000	.000	.000	.000	.723	.000					
Understanding Problem	P.C.C.	.491''	.456''	.418''	.384''	.288''	.347''	.034	.378''	.343''	1			
	P-value	.000	.000	.000	.000	.000	.000	.511	.000	.000				
Generating Ideas	P.C.C.	.343''	.389''	.391''	.418''	.444''	.473''	.023	.376''	.349''	.451''	1		
	P-value	.000	.000	.000	.000	.000	.000	.666	.000	.000	.000			
Best Choice	P.C.C.	.269''	.382''	.274''	.324''	.351''	.368''	.396''	.480''	.488''	.354''	.385''	1	
	P-value	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000		
SW Education	P.C.C.	.481''	.408''	.317''	.304''	.691''	.866''	.061	.876''	.678''	.387''	.409''	.458''	1
	P-value	.000	.000	.000	.000	.000	.000	.282	.000	.000	.000	.000	.000	

**. The correlation coefficient is at 001 level (both sides)

Hierarchical regression analysis was used to analyze which independent variables had the greatest effect on dependent variables except intervening variables [16]. Hierarchical regression analysis is one of the types of multiple regression analysis. Hierarchical regression analysis is an analysis that set the order in accordance with degree which influenced on dependent variable among several independent variable. The following hypothesis was set up for hierarchical regression analysis.

H : The nine key components of Computational Thinking will have a positive impact on Software Education. Hierarchical Regression model is shown in **Fig. 2**.

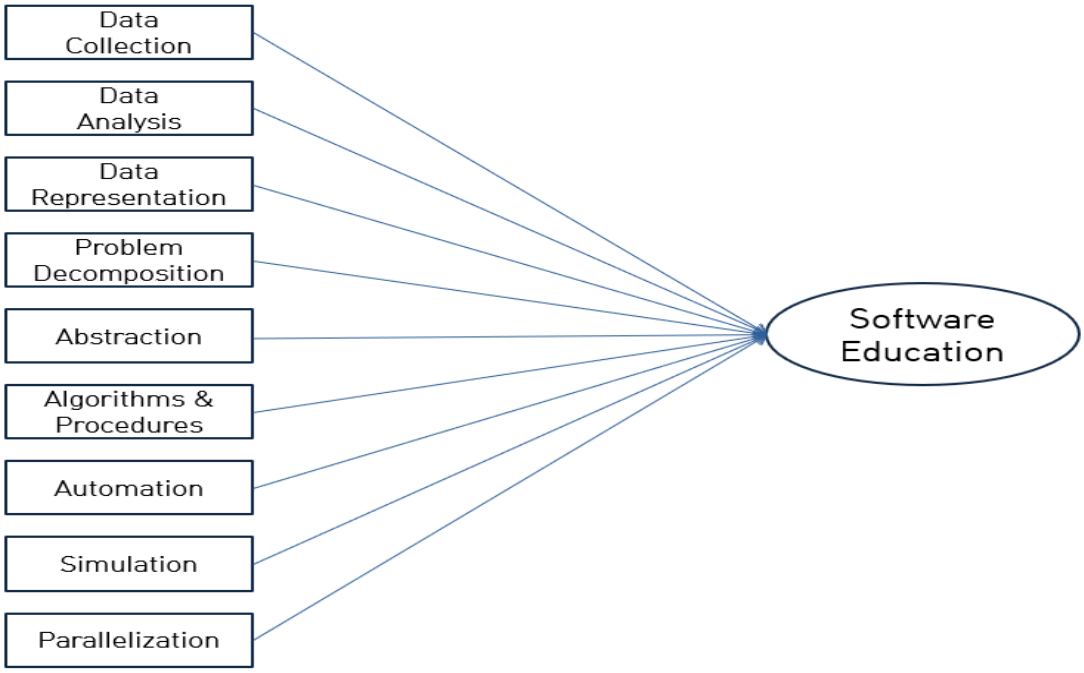

Fig. 2. Hierarchical Regression model

Table 6 is a hierarchical regression model of variables affecting Software Education. In Model 1, Data Collection (t = 96.761, p = .000) explains Software Education to 97.2%, and Data Collection affects Software Education. Model 2 was further regressed by Data Analysis (t = -3.165, p = .000) in addition to Data Collection. It showed 0.1% more influence on Software Education than Model 1. Model 3 was further regressed by Data Representation (t = 2.322, p = .034) and was found to affect Software Education. The problem decomposition of model 4 (t = 1.978, p = .135) showed no effect at statistical significance level. Model 5's Abstraction (t = -2.877, p = .013) and Model 6's Algorithms & Procedures (t = 6.877, p = .000) were found to affect Software Education. Model 7's Automation (t = -1.122, p = .263) did not have any effect at statistical significance levels. Model 8 simulation (t = 6.980, p = 0.000) and model 9 parallelization (t = 2.590, p = .010) were found to affect Software Education.

The evaluation of relative influence between variables that can improve Software Education by H hypothesis is based on the absolute value of β, which is the standardization coefficient of Model 9, which is the final model. In other words, the independent variable with the greatest absolute value of β, the standardization factor, has the greatest influence on Software Education among the variables having influence on statistical significance level. Comparing the absolute value of the standardization coefficient β in Model 9, Data Collection and Algorithms & Procedures have the greatest influence on Software Education and Problem Decomposition and Automation have the least influence on Software Education.

Therefore, sufficient data collection for the problem is needed for the growth of learners' Software Education. In addition, it is necessary to introduce a teaching - learning process that can experience sequenced algorithms & procedures in solving problems.

The tolerance limits are all 0.1 or more, so it can be judged that there is no problem in multi-collinearity. The Durbin-Watson value is 2.012, which is very close to the reference value of 2 and not close to 0 or 4, so it is judged that there is no correlation between the residuals.

Table 6. Hierarchical Regression Analysis

Independent variable	Model 1			Model 2			Model 3			Model 4			Model 5		
	standard error	β	t-value	standard error	β	t-value	standard error	β	t-value	standard error	β	t-value	standard error	β	t-value
constant	.021		2.256(.025)	.024		3.575(.000)	.025	.	2.910(.004)	.025	.	2.613(.009)	.026		3.067(.002)
Data Collection	.010	.981	96.761(.000)	.011	.997	88.083(.000)	.011	.993	86.932(.000)	.011	.991	86.410(.000)	.015	1.014	66.629(.000)
Data Analysis				.012	-.036	-3.165(.000)	.013	-.043	-3.624(.000)	.013	-.048	-3.923(.000)	.013	-.048	-3.917(.000)
Data Representation							.009	.022	2.322(.034)	.009	.019	1.662(.097)	.009	.019	1.729(.085)
Problem Decomposition										.009	.018	1.978(.135)	.009	.019	1.699(.090)
Abstraction													.014	-.033	-2.877(.013)
Algorithms & Procedures															
Automation															
Simulation															
Parallelization															
statistic	R^2=.972, Modified R^2=.972, F=9270.514, p=.000			R^2=.973, Modified R^2=.972, F=4762.403, p=.000			R^2=.963, Modified R^2=.963, F=3162.810, p=.000			R^2=.953, Modified R^2=.953, F=2382.450, p=.000			R^2=.964, Modified R^2=.964, F=1958.974, p=.000		

Independent variable	Model 6			Model 7			Model 8			Model 9			
	standard error	β	t-value	standard error	β	t-value	standard error	β	t-value	standard error	β	t-value	tolerance limits
constant	.025		2.079(.038)	.029		2.359(.019)	.027		2.765(.006)	.027		2.483(.013)	
Data Collection	.020	.912	44.344(.000)	.020	.913	44.373(.000)	.022	.841	37.706(.000)	.022	.830	36.947(.000)	.154
Data Analysis	.012	-.045	-3.907(.000)	.012	-.045	-3.926(.000)	.012	-.046	-4.208(.000)	.012	-.048	-4.415(.000)	.673
Data Representation	.009	.024	2.230(.026)	.009	.025	2.330(.020)	.008	.024	2.420(.016)	.008	.024	2.419(.016)	.778
Problem Decomposition	.009	.011	1.051(.294)	.009	.012	1.073(.284)	.008	.011	1.114(.266)	.008	.011	1.061(.289)	.763
Abstraction	.013	-.037	-2.743(.006)	.013	-.037	-2.748(.006)	.013	-.036	-2.778(.006)	.013	-.032	-2.466(.014)	.471
Algorithms & Procedures	.017	.123	6.778(.000)	.017	.123	6.841(.000)	.017	.121	4.963(.000)	.017	.120	4.860(.000)	.251
Automation				.008	-.011	-1.122(.263)	.008	-.011	-1.202(.230)	.008	-.011	-1.250(.212)	.987
Simulation							.016	.118	6.980(.000)	.016	.106	5.760(.000)	.232
Parallelization										.011	.032	2.590(.010)	.511
statistic	R^2=.966, Modified R^2=.965, F=1820.935, p=.000			R^2=.968, Modified R^2=.968, F=1562.100, p=.000			R^2=.972, Modified R^2=.971, F=1533.878, p=.000			R^2=.972, Modified R^2=.971, F=1385.726, p=.000 Durbin-Watson= 2.012			

Therefore, this research model is suitable for the regression model. The results of the study model reflecting the results of hierarchical regression analysis result can be shown as **Fig. 3**.

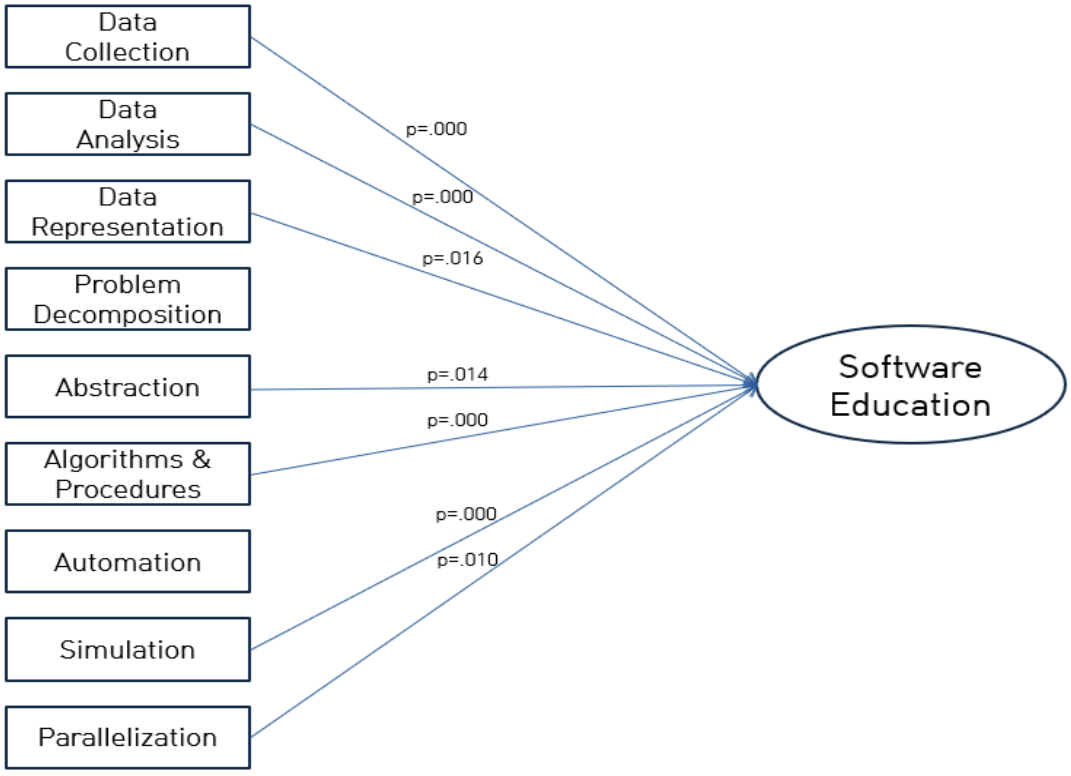

Fig. 3. Hierarchical Regression Result

3.4 Application of Research Result

Based on the results of hierarchical regression analysis, we implemented Software Education applying the concept of Computational Thinking. The study period was 205 students in the 1st and 2nd semesters of 2017, and their composition is shown in **Table 7**.

Table 7. Characteristics of Subjects

Item	Division	Number (persons)	Ratio (%)
Sex	Male	94	45.9
	Female	111	54.1
Major	Department of Humanities & Social	123	60.0
	Department of Social & Science	82	40.0

In order to measure the satisfaction of Software Education, the change of self-efficacy and occupation of the subjects, pre and post t-tests were conducted. The results are shown in **Table 8**.

Table 8. Research Results

	Pre-Post	M	n	SD	t	df	p
Learning Satisfaction	previous	3.14	205	0.34	8.189	204	.000
	post	3.79	205	0.17			
Self-Efficacy	previous	3.04	205	0.51	7.144	204	.000
	post	3.83	205	0.37			
Future Occupation	previous	3.48	205	0.34	6.823	204	.001
	post	3.52	205	0.50			

As a result, Software Education applying the concept of Computational Thinking showed a positive change in learning satisfaction, self-efficacy, and future occupation.

4. Conclusion

As each field of society is changed into software by the fourth industrial revolution, it is aiming to change the talent cultivation, curriculum and occupation. Software Education, which has been limited to specific fields in the past, is now required education for all members of society. In order to educate creative software capabilities, core competencies of future society, Software Education based on Computational Thinking is being implemented recently for non-majors in each field.

This paper has proved that Data Collection and Algorithms & Procedures are the most influential factors when applying Computational Thinking to Software Education for non-majors. Software Education based on Computational Thinking stimulates intrinsic motivation through learner-learner interaction and active participation by giving individual responsibility to learners. In addition, the learner's self-efficacy and self-efficacy are improved by solving problems by finding various ways of learning based on problem solving ability. Self-efficacy is defined by self-belief in the ability to organize and sustain the necessary activities for achieving the goal for some work. The higher the self-efficacy, the higher the level of performance that individuals can achieve by improving the performance level, goal level, effort level, and degree of immersion level. Computational Thinking has proven to have an impact.

Therefore, in this study, Software Education was conducted focusing on Data Collection, Algorithms & Procedures, and Simulation when Software Education for non-majors. As a result, we could see the positive change of satisfaction, self-efficacy, and future job occupation of learners.

In the future, we will be advance this study and suggest Software Education systematic education model for non-majors who got a difficult to Software Education.

References

[1] Kyunghun, Kim etc., "Creative problem solving information-based education policy direction navigation key competencies for the future promotion of Koreans," *Korea Institute of Curriculum and Evaluation Research Report RRC* 2012-7, 2012. Article (CrossRef Link)

[2] J. M. Wing, "Computational Thinking," *Communications of the ACM*, 49(3): 33-35, 2006. Article (CrossRef Link)

[3] J. M. Wing, "Computational Thinking and Thinking about Computing," *Philosophical Transactions of the Royal Society*, 366: 3717-3725, 2008. Article (CrossRef Link)

[4] J.M. Wing, "Computational Thinking's Influence on Research and Education for All," *Italian Journal on Educational Technology* (formerly *TD Tecnologie Didattiche*), 25(2), 7-14, 2017. Article (CrossRef Link)

[5] "Computer Science Teachers Association & International Society for Technology in Education," *Computational Thinking Teacher Resources*, 2011. Article (CrossRef Link)

[6] D. Bar, J. Harrison and L. Conery, "Computational Thinking: A Digital Age Skill for Everyone," *Learning & Learning with Technology*, Vol. 38, NO. 6, pp. 20-23, 2011. Article (CrossRef Link)

[7] Barrows, H. S., "*How to design a problem-based curriculum for the preclinical years*," New York: Springer, 1985.

[8] Jungin Kwon & Jaehyoun Kim., "A Study on Design and Development of SW Course based on Computational Thinking," in Proc. *of Asia Pacific International Conference on Information Science and Technology(APIC-IST)*, 2017.

[9] Bell, T., Witten I., & Fellows, M., "Computer Science Unplugged," Retrieved from 2015. Article (CrossRef Link)

[10] Hmelo-Silver, Cindy E., "Problem-Based Learning: What and How Do Students Learn?," *Educational Psychology Review*, 16 (3): 235, 2004.Article (CrossRef Link)

[11] Guttag, J. V., "*Introduction to computation and programming using Python with application to understanding data*," *Second Edition. Cambridge, MA: MIT Press*, 2016. Article (CrossRef Link)

[12] L. Torp and S. M. Sage., "Problems as possibilities: Problem-based learning for K-12 education," *Alexandria, VA: Association for Supervision and Curriculum Development*, 2002.

[13] Ornelas Marques, F., Marques, M.T., " No Problem? No Research, little Learning ... Big Problem!," Systemic, *Cybernetics and Informatics*, Vol.10, No. 3, pp. 60-62, 2012. Article (CrossRef Link)

[14] B. B. Levin, "Energizing teacher education and professional development with problem-based learning," *Alexandria, VA: Association for Supervision and Curriculum Development*, 2001.

[15] Hey, T., & Papay, G., "The computing universe," *Cambridge, UK: Cambridge University Press*, 2014. Article (CrossRef Link)

Jungin Kwon received her Ph.D. in Computer Education at Sungkyunkwan University. She is currently Assistant professor at Sangmyung University College of General Education. Her main research area is Computational Thinking, Problem Solving, SW Education for non-majors, SW Education, ethics, Statistics. Email : jikwon@smu.ac.kr

Jaehyoun Kim received his B.S. degree in mathematics from Sungkyunkwan University, Seoul, Korea, M.S. degree in computer science from Western Illinois University and Ph.D. degrees in computer science from Illinois Institute of Technology in U.S.A. He was a Chief Technology Officer at Kookmin Bank in Korea before he joined the Department of Computer Education at Sungkyunkwan University in March 2002. Currently he is a professor at Sungkyunkwan University. His research interests include software engineering & architecture, e-Learning, SNS & communication, internet business related policy and computer based learning.