Wireless Pers Commun (2017) 93:187–209
DOI 10.1007/s11277-016-3450-2

Analysis of Factors Affecting Achievement in Maker Programming Education in the Age of Wireless Communication

SeungJin Lee[1] · JaMee Kim[2] · WonGyu Lee[3]

Published online: 5 July 2016

Abstract Since 2010 countries around the world have been emphasizing the importance of informatics education based on computer science rather than ICT use. Korea devised an informatics curriculum that emphasizes SW education as part of its revised curriculum of 2015. The purpose of this study is to examine the factors affecting SW education achievement before implementation of the curriculum so that the revised curriculum could be efficiently established in schools. SW education was provided for 4221 elementary school students, and the factors affecting academic achievement were extracted. The results of the study showed that the achievement level of female students was higher than that of male students, and the level of understanding increased in higher grades. It was found that satisfaction with overall SW education influenced academic achievement. That is, the more satisfied students were with the key factors, such as the infrastructure required for providing SW education and interaction among teachers and students during class time, the higher the level of satisfaction. This study intended to find the key factors necessary for helping SW education take root in schools.

Keywords SW education · Curriculumn · Informatics · Programming

✉ WonGyu Lee
lee@inc.korea.ac.kr

SeungJin Lee
sungjin@keris.or.kr

JaMee Kim
celine@korea.ac.kr

[1] Department of Computer Science Education, Graduate School, Korea University, 145, 5 Ga, Anam Ro, SeongBook Gu, Seoul, Republic of Korea

[2] Majer of Computer Science Education, Graduate School of Education, Korea University, 145, 5 Ga, Anam Ro, SeongBook Gu, Seoul, Republic of Korea

[3] Department of Computer Science and Engineering, College of Information, Korea University, 145, 5 Ga, Anam Ro, SeongBook Gu, Seoul, Republic of Korea

1 Introduction

Korea's informatics education started with the proliferation of ICT (Information Communication Technology). As educational informatization develops, the presence of informatics education was used as an index for measuring the informatization of schools [1]. Using ICT in education contributed to enhancing the effects of education itself [2, 3]. In particular, ICT changed the paradigm of education and had a positive infuence, such as promoting collaboration, sharing, participation and self-directed learning [4–6]. Emphasis of ICT is for more efficient teaching and learning, and recently produced the key phrase of 'Smart Education' [7].

In this way, the development of IT affected education, and now that wireless personal computing is widespread after continuous evolutions, it is leading changes in education. In particular, informatics has gone through five iterations since the first educational guideline was produced, and informatics education is now preparing for a new takeoff. The first ICT education operating guideline was announed in 2000. It was revised in 2005 with an emphasis placed on the concept and principles of computer science rather than ICT [8]. For the revised curriculum of 2007 the subject name was changed to 'Informatics', and the focus of the curriculum was to be on problem solving abilities [9]. The fourth iteration of 2009 revised the curriculum, but 'Informatics' lost ground for education in elementary schools, became an elective in middle schools, and became a subject difficult to acces in high schools as it was changed from a general elective to intensive elective. In the fifth revised curriculum of 2015, the focus began to be to foster "creative manpower of the future" with an emphasis on problem solving abilities. In other words, SW education was emphasized, and in elementary schools, ICT, which has been a 12-h unit in practical arts since 2019, was changed to a 17-h basic SW education program. In middle schools, it will be changed from an elective to a required subject longer than 34 h in 2018, and in high schools it will be changed from an intensive elective to a general elective in 2018 [10].

SW education aims to be maker education that will enable students to make something on their own and express themselves. That is, the revised curriculum of 2015 aims to enhance computational [11]. Computational thinking, emphasized for SW education, is presented in different forms in different countries as they define it. For example, the CSTA (Computer Science Teacher's Association) & ISTE (Information Society for Technology in Education) proposed 10 elements for problem-solving in computational thinking [12]. CAS (Computing at School), in charge of informatics education in the UK, proposed 5 elements: problem decomposition, pattern recognition, abstraction, pattern generalization and algorithm design [13]. As an element of computational thinking, Code.org classified 4 problem-solving elements: abstraction, algorithm, problem decomposition and pattern recognition [14]. Even if different countries define the elements of computational thinking (CT) somewhat differently, it is clear that the purpose of SW education is to develop CT.

Korea is emphasizing SW education in informatics, and in the long run it is operating research schools and advanced schools for SW education in preparation for operation of a regular curriculum in elementary, middle and high schools in 2018 and 2019. Accordingly, this study provided SW education in elementary schools in Korea, and measured the achievement level of SW education. It also aims to extract the factors affecting the achievement level of SW education. The purpose is to show which areas focus must be emphasized to help SW education take root in public education starting in 2018. To achieve the objectives of this study, Chapter 2 analyzed previous CT-related studies, Chapter 3 proposes research methods for conducting this study. Chapter 4 derived research

results based on the research methods, and Chapter 5 described the suggestions and conclusion of this study.

2 Background

2.1 Concept and Element of Computational Thinking

CT, emphasized in informatics education, was first mentioned by Seymour Papert [15, 16]. Computational thinking can solve complex large-scale problems with algorithms. It is also used to improve efficiency [17]. The characteristics defined in computational thinking are problem decomposition, pattern recognition/data representation, generalization/abstraction and algorithms. That is, it can produce results by decomposing problems, identifying variables related to data representation, and generating algorithms. If only these characteristics are taken into consideration, it seems that it can be accomplished through programming courses. CT does not simply represent programming, but is related to the entire process of starting with a given problem and solving it through programming.

Google for Education classified CT into four different types at the Computational Thinking Workshop at Nebraska University. In other words, abstract thinking is creating and using different levels of abstraction to understand problems by creating relevant models of the real world. Algorithmic thinking is for seeking and finding the most efficient and effective method for solving a problem. The other two types are logical thinking and scalable thinking. Scalable thinking is said to decompose a large problem into smaller problems that are easier to model and compose complex solutions from simpler algorithms and components [5].

Table 1 Elements of computational thinking

CT concept	Keywords	Description
Data collection	Data source	Step to collect information necessary to solve problem
Data analysis	Statistical calculations	Identify data reliability, find pattern and find conclusion
Data presentation	Data structure such as array, linked list, stack, queue, tree and graph	Present and organize data with use of graph, chart, word or image
Problem decomposition	Objects, methods, main, function	Decompose an main idea into small tasks that can be easily processed to solve the original problem
Abstraction	Procedure, encapsulation	Reducing complexity to define main idea
Algorithms and procedures	Algorithm, implementation	A series of ordered steps taken to solve a problem ore achieve some end
Automation		Execute tasks using computer or machine
Simulation	Threading, pipelining, dividing up data	Modeling of presentation of data or procedure and experiment or test based on the model
Parallelization	Algorithm animation, parameter sweeping	Apply the structured model simultaneously and generalize it

CSTA & ISTE [12] divided the core elements into data collection, data analysis, data representation, problem analysis, abstraction, algorithm formation, procedures, automation, simulation and parallelization based on the result of a study by Barr [6] [12].

It is argued that the basic concept of CT is "to develop the models and simulations for the problems to be solved" [18]. That is, to solve a problem, it is necessary to consider the interactions between several elements existing in the problematic situation and develop a model or simulation with a system.

It is part of the effort to disseminate computational thinking among K-12 students. The ISTE (International Society for Technology in Education) working group selected three domains and developed examples related to computational thinking. These three domains are game design, simulations, and models and robotics. This group proposes a frame for developing students' CT, which consists of three stages: the use stage, the modify stage and the create stage [19].

In a bid to create the framework of the curriculum for computer science education at the university level, DePaul University's Perkovic and three others defined the detailed concepts through modification and supplementation based on the 7 principles of computing as defined by Denning which are: computation, communication, coordination, recollection, automation, evaluation and design [20] (see Table 2).

Both Tables 1 and 2 present concepts concerning computational thinking and keywords. CSTA & ISTE approached the concept of CT from the viewpoint of data processing. They have abstractions and algorithms that are necessary for data processing or automation. On the other hand, Table 2 focuses on content that helps lend coherent meaning to the keywords. For example, a typical dictionary definition of the word "communication" means having people think the same or understand each other through verbal or written interaction, but from the viewpoint of computer science, network functions like instant messages, senders/receivers and protocols must be considered. Likewise, the concept of computational thinking is presented but approaches are different. In consideration of the above, for the sake of consistency when CT-related abilities are measured, evaluation questions must be made from the viewpoint of either CSTA & ISTE or Perkovic et al.

In a study using Scratch, Brennan and Resnick [21] created three dimensions to evaluate computational thinking (see Table 3) [21, 22].

Brennan and Resnick [21] chose Scratch as the target language, and used the CT evaluation method to propose a scenario for project analysis (code analysis), interviewing

Table 2 Computer science education for University

CT concept	Keywords
Computation	State and state transition, algorithm, program, recursion and iteration, decision tree, problem complexity
Communication	Messages, sender/receiver, communication protocol, message compression, message encryption, communication channel, encoder/decoder
Coordination	Interacting processes/agents, inter-process protocols, synchronization, concurrency
Recollection	Storage media, data hierarchy, data manipulation, data locality and caching
Automation	Mapping of algorithms to physical computing object
Evaluation	Data analysis, statistics, data mining, recommender system
Design	Abstraction, modeling, modularity, information hiding, class, underlying structure

Table 3 CT dimensions

	Elements	Explanation
Computational concepts	Sequencing, repetition, parallelization, events, conditions, operators, data	Transfer of CT concept commonly included in programming language computations
Computational practices	Steps and repetitions, testing and debugging, reuse and remixing, abstraction and modularizing	Practices structuring CT concepts by way of fulfillment of project
Computational perspectives	Presentation (self-understanding), linking (recognition of others), questions (understanding of technology world)	Explains changes of understanding for oneself and the world from the perspective of computing

about works, and design. In particular, they said that the project analysis method can be used to evaluate the concept of thinking.

Discussion about the elements of CT, which can be the objective of SW education and the evaluation method after education, is ongoing. CT cannot be measured as one of the thinking skills, but it is necessary to clarify its components. Accordingly, this study regarded components of CT as evaluation of achievement, rather than measuring of CT.

2.2 Related Work

It can be said that discussion of CT in informatics education started when algorithmic thinking was first discussed. Then the concept was further defined by many scholars. As mentioned above, research on ICT and informatics education began to be conducted in 2007 [8, 23, 24]. This study will focus on CT and present Brennan and Resnick's research on CT evaluation and SRI PACT.

Brennan and Resnick [21] first developed a tool for measuring computational thinking that can be used as part of the process of improving computational thinking through programming. The tool interacts with the media being used on the premise that design-based learning activities support development of computational thinking [21]. CT evaluation elements can be largely divided into the following three.

The computing concepts include the basic concepts of programming that learners must understand for automation. The practice of computing is a supplemental measure taken because the computing concept cannot explain computational thinking sufficiently. The practice of computing makes it possible to check learners' learning process, i.e. what they learned and how while they write code (script) which is a product. Learners gradually develop what they understood in the learning process, their relationship with other factors, and various techniques, which is difficult to evaluate using the computing concept.

To evaluate the development of computational thinking by students conducting design activities using Scratch, there are three approaches: portfolio analysis, product-based interviews, and design scenarios. First, if Scratch is used when a learners' portfolio is analyzed, it is possible to analyze which computing concepts learners used often and which they did not use often. This method has the advantage of enabling a quick evaluation of computational thinking, whereas it is difficult to know the practical aspects regarding how to apply the concepts because of the heavy dependence on learners' products. Second, learners are interviewed about products. In the interview process, questions are asked to examine how learners' computational thinking develops. Questions about the background of activities, such as their thoughts about Scratch and what they do currently with Scratch

and whether they give help to and receive help from others, and questions about the creative project activities, such as how they got their ideas about their projects and how they started them, are included. The interview method can improve the weakness of using blocks to evaluate computational thinking. The weaknesses of the interview method are that there must be enough time for interviews, and that only what students remembered can be evaluated as the interviews are dependent on students' memories. Third, the development of computational thinking is evaluated based on the designed scenario. One of the designed projects is selected to be analyzed and how it can be expanded is then explained. Any functional errors found will be modified. This method makes it possible to systematically understand learners' analysis, expansion, debugging and remixing competency, checks whether they know various computing concepts, and whether they can use the concepts. As we have to wait until learners' debugging or expansion of activities are completed, however, it will take a long time, which is a disadvantage, and if the subject of a given project is a subject that learners are not interested in, there may be deviations.

As the study conducted by Brennan and Resnick [21] proposed a method for evaluating the development of learners' computational thinking in the Scratch programming environment, there must be different standards depending on the programming environment.

Second, the Principled Assessment of Computational Thinking (PACT) is a study on computational thinking evaluation conducted as part of the computer science program supported by the National Science Foundation of the U.S. [25, 26]. This study developed the framework for evaluating the learning of computational thinking based on Exploring Computer Science (ECS), a model curriculum for learning computer science [27].

PACT tried to find out what knowledge and competencies learners must learn according to the evidence-based evaluation design, determine which class activities they must conduct, and then make evaluation indexes based on the evidence supporting learning that appeared in the class activities. Computational thinking is not individual thinking that occurs inside the individual, but aims at the characteristics that appear while learners learn knowledge of computer science, competency and attitude, and practice. The practice of computational thinking, proposed in PACT, is largely divided into the concepts of computer science, research competency and noncognitive areas. The concepts of computer science include algorithms, programming, abstraction, debugging, testing, variables and repetition. Research competencies include evaluation, exploration, analysis, explanation, elaboration and modeling. Also, noncognitive competencies include communication, competency, cooperation, leadership, self-efficacy, concept of self and tenacity. Table 4 shows the components of computational thinking proposed in PACT.

Table 4 Components of PACT's CT

Computer science concept	Research competency	Noncognitive competency
Algorithm	Evaluation	Communication
Programming	Exploration	Cooperation/teamwork
Repetition	Analysis	Leadership
Abstraction	Explanation	Self-efficacy
Debugging and test	Elaboration	Concept of self
Variable	Model	Tenacity, etc

Table 5 Test questions

Number	SW competency CT components Subject of learning	Analysis competency				Design competency		Implementation competency	Inferential competency		Score
		Data collection	Data analysis	Data representation	Problem decomposition	Abstraction	Algorithm	Automation	Simulation	Parallelization	
1	Defining information characteristics to be collected	○									5
2	Establishing the classification criteria for the collected information		○								6
3	Selecting a data representation method appropriate for the information characteristics			○							5
4	Estimating the result of parallelization according to the conditions									○	7
5	Defining the information processing rules					○					7
6	Designing the problem solving process fit for the conditions						○				7

Table 5 continued

Number	SW competency CT components Subject of learning	Analysis competency				Design competency		Implementation competency	Inferential competency		Score
		Data collection	Data analysis	Data representation	Problem decomposition	Abstraction	Algorithm	Automation	Simulation	Parallelization	
7	Estimating the result of repeated execution								○		5
8	Designing the problem solving process fit for the result						○				6
9	Designing the problem solving process fit for the conditions						○				7
10	Dividing and solving the problem				○						7
11	Defining information fit for the result					○					7
12	Designing the problem solving process						○				7
13	Designing the problem solving process fit for the result								○		6
14	Simplifying the problem solving process through the iteration process							○			6

Table 5 continued

Number	SW competency / CT components / Subject of learning	Analysis competency				Design competency		Implementation competency	Inferential competency		Score
		Data collection	Data analysis	Data representation	Problem decomposition	Abstraction	Algorithm	Automation	Simulation	Parallelization	
15	Designing the process of the handling conditions						○				6
16	Estimating the result of the program execution								○		6
Total	16 (100 %)										
CT components		1	1	1	1	2	5	1	3	1	
SW competency		4 (25.0 %)					7 (43.7 %)	11 (63 %)	4 (25.0 %)		100

Scoring criteria: high (7 points), intermediate (6 points), low (5 points)

Computational thinking can be promoted when the elements of each area are in harmony. For example, let's say you are analyzing computing work with several people. The concepts of computer science, which are required at this time, are algorithms and programming, and research competencies, i.e. analysis and explanation competency, and noncognitive competencies such as communication competency and cooperation [27].

According to PACT, for learners' evaluation, the concepts of computer science, research competencies and noncognitive competencies related to what must be learned must all be extracted, the characteristics of what learners say, do and produce must be found, and it must be possible to predict what learners can produce. Also, it must be possible to provide the evaluation situation for accomplishing the goals that learners must reach. It is also necessary to create an evaluation environment consisting of diversified evaluation levels so that difficulty may be adjusted. SRI's PACT proposed a framework for evaluating computational thinking based on learners' responses that may appear in the classroom environment.

3 Experimental Method

In the experimental method, how the achievement test tool was developed, how the students' perception survey was developed, the subjects, and how the test and class were conducted were all described.

3.1 Instrument

3.1.1 Achievement Test

The achievement test measurement tool was composed of four SW competencies: analysis competency, design competency, implementation competency, and inferential competency. Each SW competency consisted of subordinate elements as follows:

Analysis competency was composed of data collection, data analysis, data representation and problem decomposition.

Design competency was composed of abstraction and algorithm.
Implementation competency was composed of automation.
Inferential competency was composed of simulation and parallelization.

The achievement test for the elementary school students, who were the subjects of this study, were presented with 16 questions as shown in Table 5. The category of Analysis competency had four questions (25 %), Design competency 7 questions (43.7 %), Implementation competency one question (6.3 %), and Inferential competency four questions (25 %).

3.1.2 Investigation of Students' Perceptions

The questions for investigating students' perceptions of SW education were as follows:

First, attitude toward SW education, second, satisfaction with SW education, and third, the effect of SW education. The reliability of the questions was estimated with Cronbach α. As Cronbach α provides the most conservative estimate, this method is suitable for determining reliability.

As a result of estimating the reliability, the reliability coefficient for each element exceeded .70. So it can be said that the reliability coefficient used in this study is reliable (see Table 6).

3.2 Research Process

Here, development of the achievement test tool and survey questions are described.

3.2.1 Procedure for Developing Achievement Test Questions

This study was conducted according to the following procedure (for starters, the procedure for developing the questions for the achievement test is described):

In step 1, for the SW education achievement test, literature review was conducted for CT-related concepts. Based on the concepts derived through the literature review, the 1st expert meeting was held. Participants in the meeting were 15 experts related to contents

Table 6 The questionnaire for students' perceptions of SW education

	Subject		Cronbach α
1	Gender		
2	Grade		
1	Attitude toward SW	Convenience of SW	.762
2		Degree of desiring SW-related jobs	
3		Importance of SW in social development	
4		Importance of SW education	
5		Necessity of SW knowledge from the viewpoint of jobs	
1	Satisfaction with SW education	Expectations for SW class	.817
2		Interest in programming learning activities	
3		Degree of immersion in programming	
4		Interest in SW CLASS	
5		Frequency of voluntary problem solving in SW CLASS	
6		Frequency of making presentations in SW CLASS	
7		Frequency of cooperation activities in SW CLASS	
8		Smart device satisfaction with smart devices used in SW classes	
9		Convenience of programming languages and boards used in SW class	
1	Effects of SW education	Solving real life problems in SW education	.858
2		Confidence in programming after SW CLASS	
3		Frequency of determining the sequence of problem solving in SW CLASS	
4		Degree of desiring to participate in SW-related clubs	
5		Will to study SW	
6		Sense of achievement felt in SW class	

and evaluation, i.e. five informatics teachers who were teaching SW-related contents, five college professors and five evaluation-related experts.

In step 2, based on what was discussed at the 1st meeting, the framework for evaluating the concepts was established. The developed framework was reviewed by the experts, and used as the framework for developing questions.

In step 3, questions were developed according to competencies and contents. Competencies were: analysis, design, implementation and inferential competency. 16 elements were selected as contents. According to the selected elements, one or two questions were developed. In total, 25 questions were developed.

In step 4, the content validity of the developed questions was tested, and a preliminary test conducted. In the preliminary test, as the subjects were elementary school students, we made sure that the contents were not too difficult to answer. Accordingly, the preliminary test was conducted for 20 elementary school students.

In step 5, what was discovered in the preliminary test was modified, and the reliability of each question was calculated. 16 questions were finally selected.

3.2.2 Survey Development Procedure

The survey questions used in this study were developed as follows:

The elements of the survey were extracted from related studies. The extracted elements were reviewed by the 15 experts who verified the evaluation questions. The content validity was reviewed, and like the achievement test, a preliminary test was administered to 20 elementary school students. As a result of the preliminary test, questions that students did not understand or were likely to be distorted were excluded.

The reliability of the preliminary test was verified, and 20 questions were selected for this test.

3.3 Research Subjects and Process

The subjects of this study was students taking classes in SW research schools, which are funded by the Ministry of Education of Korea. A total of 68 schools were selected as SW research schools from among schools that submitted proposals. Among the selected schools, only elementary schools were chosen for this study.

In other words, of the 68 schools, 45 are elementary schools and 23 are middle schools. This study conducted the research only in the 45 elementary schools. All the students in classes participating in the research school project in the 45 elementary schools are the subjects in this study. Accordingly, a total of 4573 students participated, and among them

Table 7 Subjects

	Frequency
Gender	
Male	2150
Female	2071
Grade	
5th	2052
6th	2169
Total	4221

4397 students participated in both the achievement test and the survey. Among the participants, those whose responses were inconsistent or provided confusing responses were excluded, so in the end data from 4221 students were analyzed (see Table 7).

The achievement test and survey regarding SW education were conducted on the web from October 26, 2015 to November 3, 2015.

3.4 Designing and Conducting the Class

To conduct this study, research schools were operated, and achievement about SW education was examined.

As research schools conduct the project over a 2-year period, education is provided on an annual basis. Table 8 shows the annual plan for 2015.

Table 8 Annual plans of research schools

Month	Activity	Problem-solving element	Tool
4	Understanding the basics of software	Data collection and analysis	Writing instruments
4	Exploring the code. org site	Data collection, analysis and representation	
5	Light-bot game activity	Data collection, analysis and representation	Light-bot
6	Scratch game and storytelling	Data analysis and representation, problem decomposition, abstraction	Scratch
9–11	Scratch program activity	Data analysis and representation, problem decomposition, abstraction	Scratch
10	Understanding Aduino and controlling LEDs	Automation, simulation, parallelization	Aduino
11	Utilizing illuminance sensors	Automation, simulation, parallelization	Aduino
12	Making smart LEDs	Automation, simulation, parallelization	Aduino

Table 9 Experience of programming (Scratch_game and storytelling)

Topics	Contents to learn	Learning elements	No. of lessons
Making programs	Looking at others' programs to find the algorithm, and then making the programs	Programming experience (algorithm)	1
Making interesting old stories	Making an animation with a story	Programming experience (talking with Sprite)	2
Into the story I make	Making an animation with acoustic effects	Programming experience (sound, acoustics)	1
Drawing	Using the pen function of Scratch to draw abstract paintings	Programming experience (pen function)	1
Moving the stars	Expressing the movement of stars with various graphic effects	Programming experience (graphic effects)	1
Making games that move blocks	Using variables X and Y, and coordinates to make a game in which students move blocks with the mouse to catch balls	Programming experience (game)	1
Making ball catching games	Making a game that counts the number of times a ball is dropped from an airplane and caught	Programming experience (game)	1
Total			8

Table 10 S/W curriculum teaching and learning course plan

Semester	2nd semester	Date	November 26, 2015 (Monday)	Targets	Class 3 of the 6th grade			
Unit	Work faster and more			Class number	2/4	Guidance teacher		
Learning topic	Power			Teaching and learning model		Large group learning		
Learning objective	To understand the concept of power and be able to calculate it To be able to make and code the power algorithm.			S/W education	Area	Understanding algorithms		
					Input stage	Introduction	Development	Closing
Supplies	Textbooks, study materials, pens and computers							
Content element	Algorithm design and basics of programming							

Stage	Learning element	Teaching–learning activities	Time	Remarks
Introduction	Check previous learning	Check the definition of work in science and the method of calculating the amount of work	5 min	
	Set learning objectives	Check learning objectives		
Development	Explain for efficiency of work	[Exploration] Comparing the efficiency of work 1. Have students calculate the amount of work when a person carries 100 bricks weighing 20 N each and a crane carries them to a height of 3 m 2. Calculate the amount of work done by a crane and a person per second, and compare the efficiency of work done by each crane and person	10 min	
		[Explanation of the concept] 1. Power explains that the shorter it takes, the greater the efficiency of the work is when the same work is done, and the greater the amount of work is done, the higher the efficiency of the work is when work is done for the same amount of time Tell them that power can be calculated by dividing the amount of work done by the time consumed Explain that the unit of power is the Watt (W), and that 1 W is the power when 1 J of work is done in a second		
	Use the algorithm to calculate the amount of work	1. Have them make the algorithm for calculating the power and explain it 2. Use the algorithm to explain program coding	20 min	
Closing and evaluation	Wrap up what is learned	Wrap up what is learned in this class	5 min	
	Notice of next class	Tool		

Table 10 continued

Evaluation stage	Evaluation method	Achievement standard	Achievement level
Evaluation	Writing algorithms	Real-life problems can be solved by algorithm design and programming	High
		Real-life problems can be identified and solutions can be expressed with the algorithm	Intermediate
		Real-life problems can be identified and solutions can be expressed in a natural language	Low

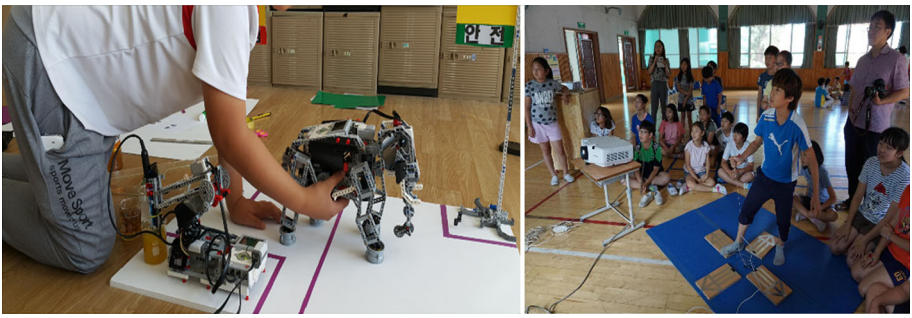


Fig. 1 Robot and Makey Makey activities

According to the annual plan shown in the Table, lectures were first given on contents that would help students understand software, then, second, students learned the contents of code.org that helped them experience programming.

Third, students experienced the education programming language (EPL) in earnest. For the EPL, I had students use Scratch for games and storytelling.

Fourth, in the last 3 weeks of the Scratch activities, Python was used for programming.

Fifth, from October to December, Aduino was utilized as well as Scratch to let students experience physical computing. The focus of the physical computing was placed on understanding input and output.

The topics and contents of the Scratch activities conducted in June and September based on the annual plan are as shown in the Table 9.

As the students were new to programming, I gave concrete forms step-by-step from using blocks to storytelling.

I set up the schedule for each stage as described above, and taught classes according to the specific teaching–learning plan.

The course was taught for 1 or 2 h a week for a total of 34 h. The sample teaching and learning course plan for the classes is shown in Table 10.

As shown in Table, the teaching and learning course plan includes evaluation of the unit.

As described above, students directly participated in the class, and various unplugged activities were conducted in addition to robot or Makey Makey activities (Fig. 1).

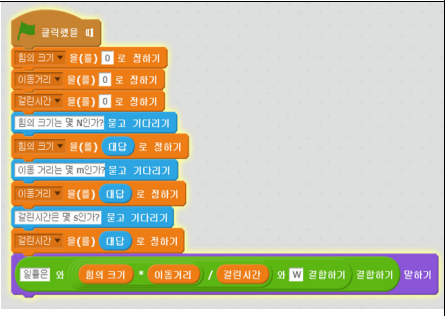
Also, the students will be able to make samples of the algorithm, and they will be able to draw the flow chart at the beginning of the class. The flow chart is shown in Table 11.

4 Results

4.1 SW Education Achievement Test

The result of the SW education achievement test conducted at the elementary schools was as follows: The average of the SW achievement test was 54.238 points (out of a possible 100 points), the average of analysis competency, a subordinate element, was 13.895 points (out of a possible 23 points), the average of design competency was 27.546 points (out of a possible 47 points), the average of implementation competency was 3.204 points (out of a possible 6 points), and the average of inferential competency was 9.629 points (out of a

Table 11 Algorithm and coding sample

Making an algorithm	Coding
Start	
Set variable 'magnitude of force' as 0	When clicked
Set variable 'travel distance' as 0	Set the magnitude of force as 0
Set variable 'time taken as 0	Set the travel distance as 0
'What N is the magnitude of force?'	Set the time taken as 0
Enter magnitude of force	What is the magnitude of force in 'N'? Ask and wait
Set the answer as variable '( )'	Set the magnitude of force as ANSWER
'What is the travel distance?	What is the travel distance? Ask and wait
Enter travel distance	Set the travel distance as ANSWER
Set the answer as variable '( )'	How many seconds of time is taken? Ask and wait
How many seconds of time is taken?'	Set the time taken as ANSWER
Enter time taken	(Power is the magnitude of force times the travel distance/the time taken times W.) Speak
Set the answer as variable '( )'	
Power is ( )/( )W	
End	

possible 24 points). Meanwhile, the average of attitude was 72.853 points (out of a possible 90 points) (Table 12).

As a result of using t-tests to compare the differences in the average scores of SW achievement tests between male and female students in elementary school, female students had higher scores than male students. However, there was no statistically significant difference (see Table 13).

As a result of analyzing the SW achievement test scores of elementary school students by grade, 6th graders had higher scores than 5th graders, and the difference was statistically significant at a significance level of .001. That is, as school ages increased, the achievement level also increased (see Table 14).

Table 12 Result of the SW education achievement test

Classification	Minimum value	Maximum value	Average	SD
Overall achievement	0.0	100.0	54.24	20.76
Analysis competency	0.0	23.0	13.86	6.29
Design competency	0.0	47.0	27.55	12.50
Implementation competency	0.0	6.0	3.20	2.99
Inferential competency	0.0	24.0	9.63	5.95
Attitude	18.0	90.0	72.85	14.44

Table 13 Difference in SW achievement by gender

Gender	M	SD	t	*p*
Male	53.64	21.15	−1.907	.057
Female	54.85	20.35		

Table 14 Difference in SW achievement by grade

Grade	M	SD	t	*p*
5th	51.57	19.95	−8.466	.000
6th	56.93	21.11		

Table 15 Languages used (multiple responses')

Language	(%)
Scratch	59.3
Entry	66.2
Python	.3
App inventor	.8
Java	2.3
C	1.7
BASIC	.4
Java script	.8
C++ etc.	7.1

4.2 Languages Used in SW Education

We also analyzed what languages were used during software education in elementary schools, see Table 15.

The analysis result shows that 59.3 % of all students responded 'Scratch,' and 66.2 % responded 'Entry.' 2.3 % responded 'Java', and 1.7 % responded they learned 'C'.

Table 16 The correlation among the attitude toward SW, satisfaction variables and the achievement test

Classification	SW attitude	Satisfaction with SW education	Effect of SW education
SW achievement test	.189**	.182**	.215**
Analysis competency	.116**	.100**	.133**
Design competency	.174**	.177**	.203**
Implementation competency	.068**	.074**	.087**
Inferential competency	.135**	.120**	.141**
Attitude score	.704**	.735**	.730**

** $p < .001$

Table 17 The influence of perceptions of SW on attitude

	B	S.E	β	t	*p*
(Constant)	−.980	4.076		−.240	.810
Grade	5.470	.636	.134	8.606	.000
Gender	2.466	.654	.059	3.768	.000
Satisfaction with SW	.763	.692	.031	1.104	.270
SW effect	4.971	.844	.205	5.891	.000

Table 18 Influence of students' perceptions of and attitude toward SW education on academic achievement

	B	S.E	β	t	*p*
(Constant)	30.740	1.869		16.445	.000
Grade	1.265	.291	.044	4.343	.000
Gender	1.412	.300	.049	4.713	.000
Attitude toward SW	2.246	.397	.127	5.661	.000
Satisfaction with SW	6.387	.317	.377	20.153	.000
Effect of SW	5.089	.387	.302	13.147	.000

4.3 Correlation Coefficient

As for students' perceptions of SW education, the correlation among their attitude toward SW, satisfaction with SW education, the effect of SW education, the SW achievement test, and the subordinate elements of the SW achievement test (analysis competency, design competency, implementation competency, inferential competency) were all analyzed, the results are shown in Table 16.

As a result of the correlation analysis, satisfaction with SW education and attitude showed the highest level of correlation, i.e. .735, and it was statistically significant. Next, the effect of SW education and attitude had a correlation coefficient of .730. That is, learners who thought SW education was effective can be said to show a high score in attitude as well.

The correlation between the SW achievement test result and the effect of SW education was the highest, .215. That is, it can be concluded that the achievement test score of learners who thought SW education was effective was high.

4.4 The Influence of Students' Attitude Toward SW on Academic Achievement

Among students' perceptions of SW education, how much influence such variables as the attitude toward SW, satisfaction with SW education and the effect of SW education had on the SW achievement test was analyzed, the results are shown in Table 17.

As a result of analyzing the influence of students' variables on their attitude toward SW, the perception that SW education was effective had the most influence (β .205) followed by grade (β .134). The influence was statistically significant. That is, the higher the grade and the more positive their perception of the effect of SW education, the higher their achievement level.

What follows is the result of analyzing the influence of students' perceptions of SW education and attitude on academic achievement (see Table 18).

The analysis result shows that satisfaction with SW had the most influence on academic achievement (β .377). That is, it can be concluded that as satisfaction with SW education increases, academic achievement also increases. Next, the effect of SW education had a statistically significant influence on academic achievement (β .302). Grade, gender and attitude also had a positive influence on academic achievement.

5 Conclusions

The purpose of this study is to make students ready for informatics education that will start in middle schools in 2018 and in elementary schools in 2019. In other words, as informatics has become a mandatory subject that must be taught for more than 34 h in middle schools, this study intended to provide SW education and conduct achievement tests to find the variables affecting academic achievement in preparation for when the subject is offered as a part of the regular curriculum. The class was offered to 4221 elementary school students, and their attitude toward SW education, satisfaction with SW education and its effect were analyzed. The results were as follows:

First, there was no statistically significant difference in achievement by gender, but the achievement level of female students was higher.

Second, the higher the grade, the higher the achievement level.

Third, the result of the correlation analysis showed that learners' attitude was highly correlated with achievement and satisfaction. In particular, satisfaction also had a high level of influence on academic achievement.

Countries around the world have tried to reinforce informatics education since 2010, and many countries like Japan, the U.S., the U.K. and India revised their curriculums to reinforce SW education. Korea also began to make efforts to reinforce SW education through its revised curriculum of 2015. However, what is important in education is to make people perceive the necessity of education. That is, it is necessary to help students perceive the necessity for themselves, and display their competency as makers. In particular, we must not forget that fostering makers is not fostering technicians, but that it is education for improving self-expression.

The difficulties experienced during this study can be summarized as follows:

First, teaching and learning methods need to be changed. A great difficulty that was encountered during the class was that though students have a high interest in SW education there was a difference in level among the students. That is, if there are 25 or more students

in a class, teachers cannot properly conduct the class well if there are too many coding mistakes. In programming classes, teaching assistants are required, but schools cannot provide them. If teachers fail to take the level of individual students into consideration when they teach classes, it will be necessary to find a solution for students who fall behind. For example, teaching and learning methods need to be changed, which might lead to, for example, developing specialized learning modules for different levels.

Second, if the level of satisfaction with SW education was high, the achievement level was also high. Students' satisfaction encompasses various aspects, including the infrastructure, teachers' expertise and cooperative learning. This means that if any one element is lacking, the satisfaction level may be lowered. Accordingly, it is necessary in the future to further clarify what kind of support teachers must provide, and what kind of efforts the office of education is required to make.

Third, programs need to be diversified. The languages that were used most frequently in elementary schools during this study were Scratch® and Entry®. All the other languages were used by less than 10 % of the students. To ensure that SW education will be more effective and students able to display their competencies as makers, not only block-structured languages, but also textual languages must be accessible at the same time. This study targeted elementary school students, but according to the revised curriculum of 2015, middle schools are also using block-structured languages. That is, we must take into consideration the fact that accessibility to textual languages may be decreased. Accordingly, if SW education is provided, it is necessary to clarify what the purpose of it truly is. For example, we must make efforts to select programming languages in consideration of whether SW education is simply to arouse the interest of students or to reinforce their competency as makers.

This study analyzed the influence of attitude, including satisfaction with SW education, on academic achievement in pilot schools with public SW education in the offing. As many countries have become aware of the necessity of SW education, if there are countries providing education similar to that of Korea, they should learn from this study.

Acknowledgments This work was supported by a National Research Foundation of Korea (NRF) Grant funded by the Korean government (MSI No. 2013R1A2A2A03016926).

References

1. JaMee, K., & WonGyu, L. (2011). An analysis of educational informatization level of students, teachers, and parents. *Korea, Computers & Education, 56*(3), 760–768.
2. Shinde, V. V., & Inamdar, S. S. (2013). Problem based learning (PBL) for engineering education in India_need and recommendations. *Wireless Personal Communication, 69*, 1097–1105.
3. Santiprasitkul, S., Sithivong, K., & Polnueangma, O. (2013). The first year nursing students achievement and critical thinking in local wisdom course using problem based learning process. *Wireless Personal Communication, 69*, 1077–1085.
4. Zhang, L., Liu, Y., Zhan, X., Yang, X., Chi, X., & Zhao, S. (2012). An innovative location and contextaware video sharing application on smart phone. *Wireless Personal Communication, 66*, 493–509.
5. Google CS4HS Workshop. (2013). Computational thinking. http://www.cs.unomaha.edu/~hsiy/CS4HS.

6. Barr, D., Harrison, J., & Connery, L. (2011). Learning and learning with technology, computational thinking: A digital age skill for everyone. *Learning & Leading with Technology, ISTE, 38*, 20–23.
7. Jo, J., Park, K., Lee, D., & Lim, H. (2014). An integrated teaching and learning assistance system meeting requirements for smart education. *Wireless Personal Communication, 79*(4), 2453–2467.
8. Seungeun, C., Soojin, J., Daiyong, K., Hansung, K., Seungbum, K., JaMee, K., et al. (2011). Measuring achievement of ICT competency for students in Korea. *Computers & Education, 56*(4), 990–1002.
9. Kim, J. H., Jung, S. Y., & Lee, W. G. (2008). Design of contents for ICT literacy in-service training of teachers in Korea. *Computers & Education, 51*(4), 1683–1706.
10. Ministry of Education of Korea. (2015). Revised Curriculum for informatics. http://www.moe.go.kr/web/100020/ko/board/view.do?bbsId=141&boardSeq=60303.
11. Kim, K. H. et al. (2015). Development of informatics curriculum frame. Korea Institute for curriculum and evaluation. Research Report, 2015–2017.
12. CSTA & ISTE. (2011). Operational definition of computational thinking for K-12 Education. http://csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf.
13. Nacce & CAS. (2014). Computing in the national curriculum: A guide for secondary teachers. http://www.computingatschool.org.uk/data/uploads/cas_secondary.pdf.
14. Code.org. (2014). Computational thinking. http://learn.code.org/unplugged/unplug2.pdf.
15. Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. NewYork: Basic Books Inc.
16. Papert, S. (1996). An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning, 1*(1), 95–123.
17. Guzdial, M. (2008). Education: Paving the way for computational thinking (PDF). *Communications of the ACM, 51*(8), 25.
18. Moursund, D. (2009). Computational thinking. Retrieved from IAE-pedia. http://iae-pedia.org/Computational_Thinking.
19. Lee, I., et al. (2011). Computational thinking for youth in practice. *ACM Inroads Archive, 2*(1), 32–37.
20. PerKovic, L., Settel, A., Hwang, S., & Jones, J. (2010). A framework for computational thinking across the curriculum. In *Proceedings of the international conference on ITiCSE*.
21. Brennan, K., & Resnick, M. (2012). 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.
22. Brennan, K., & Resnick, M. (2013). Imagining, creating, playing, sharing, reflecting: How online community supports young people as designers of interactive media. In C. Mouza & N. Lavigne (Eds.), *Emerging technologies for the classroom*. NewYork: Springer.
23. Aoki, H., Kim, J., Idosaka, Y., Kamada, T., Kanemune, S., & Lee, W. (2012). Development of state-based squeak and an examination of its effect on robot programming education. *KSII Transactions on Internet and Information Systems, 6*(11), 2880–2900.
24. Kwon, D., Yoon, I., & Lee, W. (2011). Design of programming learning process using hybrid programming environment for computing education. *KSII Transactions on Internet and Information Systems, 5*(10), 1799–1813.
25. Bort, H., & Brylow, D. (2013). CS4Impact: Measuring computational thinking concepts present in CS4HS participant lesson plans. *SIGCSE, 13*, 427–428.
26. Roschelle, J. (2015). Innovating pedagogies: SRI Education and UK's Open University Uncover Ten Trends. https://www.sri.com/blog/innovating-pedagogies-sri-education-and-uks-open-university-uncover-ten-trends#sthash.cC3pnRyP.dpuf https://www.sri.com/.
27. Rutstein, D., Snow, E., & Bienkowski, M. (2014). Computational thinking practices: Analyzing and modeling a critical domain in computer science education. Paper to be presented at the 2014 annual meeting of the American Educational Research Association (AERA), Philadelphia, PA.

SeungJin Lee is a Ph.D. student in the Department of Computer Science Education at Korea University. He received his M.S. degree in Science education at Seoul National University. Currently he is a member of Information and Computing (INC) LAB at Korea University and Korea Education Research and Information Service (KERIS). His research interests include ICT literacy, e-Learning and Informatics Education.

JaMee Kim is a assistant Professor in the major of Computer Science Education, Graduate School of Education at Korea University. She received the B.A and M.A degrees in Education Evaluation from Ewha Womans University, Seoul, in 1992 and 1995. She Pursuing a Ph. D degree in Computer Science Education at Korea University, in 2011. Her recent research interests are computer science education, education evaluation, educational informatization evaluation, and robot programming research.

WonGyu Lee is a professor at Korea University, Department of Computer Science and Engineering, College of Information in Seoul, Korea. He received the M.S and Ph.D. degrees in Computer Science from University of Tsukuba. From 1993 to 1996 he was a senior researcher at Korea Arts and Culture Service. He was the president at Korea Association of Computer Education in 2002. He was the chief director of Creative Informatics & Computing Institute at Korea University from 2007 to 2011. He was a dean of the office of Academic affairs at Korea University from 2011 to 2015. He made many progresses in the research of computer science education. His research interests are informatics education, especially educational policy, information representation, information management.