Universal Journal of Educational Research 7(5A): 103-113, 2019 http://www.hrpub.org
DOI: 10.13189/ujer.2019.071512

The Effect of Network-Based PUMA Teaching-Learning Model on Information Literacy, Computational Thinking, and Communication Skills

Young-Sik Jeong[1,*], Young-Hoon Sung[2]

[1]Department of Computer Education, Jeonju National University of Education, Korea
[2]Korea Department of Computer Education, Chinju National University of Education, Korea

Abstract Network-based Preparing Un-coding Modifying and Adding (PUMA) is a teaching-learning model that focuses on communication among students rather than emphasizing programming language grammar in an online education platform. The Network-based PUMA model consists of four steps: preparing, un-coding, modifying and adding. The results of the PUMA model applied to the National University of Education's student teachers for six weeks show that students were able to maintain their information literacy skills and improve their problem analysis ability and abstraction ability in computational thinking following the implementation of this model. It was also shown to improve communication skills.

Keywords PUMA, Programming Education, Teaching and Learning Model, Computational Thinking, Information Literacy, Communication Skill

1. Introduction

Based on intelligent robots and Big Data, the software is creating new value in the fourth industrial revolution era. In response to this reality, the Korea Ministry of Education revised elementary and secondary school curriculum in 2015. The revisions allowed for the introduction of programming education to secondary schools beginning in 2018 and to elementary schools in 2019 [1]. Due to these factors, programming education curricula designed for elementary schools should develop algorithmic thinking that can solve various problems using computing systems [2]. Additionally, the revised curriculum emphasizes that elementary schools should minimize using procedural thinking [3][29] and grammar-based programming education to teach problem-solving. Instead, students should understand the programming structure and enhance their computational thinking skills by solving problems [1] [4].

Recently, however, programming education in elementary schools has increasingly used grammar-based education to teach students specific programming languages rather than emphasizing computational thinking. This has created several issues. Most specifically, studies have shown that when teacher-followed education techniques are used it is difficult to maintain students' interests and limit their creative problem-solving abilities are limited [5-8].

Because the goal of programming language education is to enable students to create programs necessary to solve various problems using computers [9], it is important that students remain engaged in the curriculum. This is sometimes problematic for students new to programming as they do not fully understand what programming is and are often overwhelmed by the idea of learning how to program using programming languages. However, just as one can sufficiently communicate in natural, everyday languages with only a few words, programming languages can also be communicated contextually using a computer [10].

In the future, core competencies for programming education students must include collaborative thinking, communicational thinking, critical thinking, and creative thinking [11]—all of which can be developed through network-based education. In particular, the Internet can be used to overcome the limitations of time and space by allowing students to exchange opinions with others across long distances. This can lead to an increased interest in learning [12][26]. In addition, when students are able to exchange ideas, they can find more information than when they search for information alone. Thus, allowing for the expansion of their critical thinking skills [13][28].

Therefore, in this study, we examined how to improve

programming education, developed the PUMA model to increase students' communication and computational skills, and applied the model to test subjects to determine how information literacy, computational thinking, and communication skills change when using the PUMA model.

2. Improvement of Programming Education

2.1. Purpose of Programming Education

The purpose of teaching programming in elementary and secondary schools is not to create future programmers. Rather, the goal is to develop students' computational thinking ability [9][30].

Computational thinking employs deep-thinking skills to solve problems using computers. In this style of thinking, students analyze, decompose, find, and abstract the problem. Then the solution is expressed as an algorithm that can be used to efficiently solve the same or similar problem(s) in the future. The algorithm is then implemented as a program and executed on a computer. Using this process helps students further develop their ability for computational thinking [14].

2.2. Definition of Computational Thinking

Computational thinking was derived from the algorithmic thinking proposed by Newell et al. (1960) [15-31]. The concept was embodied and further developed by Wing as a necessary competency for students [16]. Wing defined computational thinking as a thought process designed to help people and computers that process information solve problems more effectively [17].

In the Korea Ministry of Education's 2009 revised curriculum, computational thinking is defined as the ability to understand basic concepts and principles of information science and technology and apply them to observing and solving various problems in real life [18-32]. In the 2015 revised curriculum, computational thinking is defined as the ability to understand diverse disciplines and apply creative solutions to real-life situations using basic concepts, principles, and computing systems [1].

2.3. Components of Computational Thinking

The Korea Ministry of Education has since divided the computational thinking component into six categories: (1) structuring, (2) organizing, (3) abstraction, (4) automation, (5) optimization, and (6) generalization. Based on the results of a Delphi survey conducted by computer education professors at national universities of education around the globe, the Korea Association of Information Education (KAIE) narrowed the components of computational thinking to five categories: (1) problem definition, (2) data analysis, (3) abstraction, (4) automation, and (5) generalization [20] [21]. The definitions for each category are as follows:

First, *problem definition* is the process of understanding, expressing, and decomposing the problem to be solved. Students recognize what the problem is. They then express the current state of the problem and the state of the goal.

Second, *data analysis* is the process of gathering, classifying, and structuring the data necessary for problem-solving. The collected data is expressed through words, text, and pictures. It can also be structured into tables or graphs that will help solve problems.

Third, *abstraction* is the process of analyzing problems and data to find patterns. Problem-solving methods are then expressed through logical reasoning and modeling using algorithms, such as flowcharts and pseudo-codes.

Fourth, *automation* is the process of implementing the problem-solving methods through actual programming.

Finally, *generalization* is the process of assessing whether the chosen problem-solving method is correct, efficient and economical in resource use and whether it is easy for people to use.

2.4. Problems with Programming Education

Building-block programming languages, such as Entry or Scratch, are easy to learn because building programs can be done by stacking already created blocks of code without developing entirely new code [22][34]. However, it is difficult to ensure that students using block-based programming languages have a good understanding of the grammar and basic principles of a particular programming language. In addition, students often experience difficulty in transitioning from using a block-based programming language to a text-based programming language [23]. Additionally, grammar-based programming education, especially for teacher-followers, reduces students' creative thinking skills [5][24].

In order to solve these problems, students should be instructed to learn the program sources contextually by viewing the results of programs they are familiar with.

2.5. Improvement of Programming Education

To be able to communicate freely with their computers, students should not depend on grammar-based instruction for specific programming languages. Unlike natural languages, computer programming languages consist solely of reading and writing [25][33]. Therefore, students should be able to read and understand only the given program sources. Fortunately, programming languages are easier to learn than natural languages because the vocabulary used is extremely small, and most of them can be easily and quickly understood. In addition, the program structures, such as sequential, iteration, selection, event,

and function, are simple. Students can create a program without having to memorize words or phrases in the programming language. They can simply follow the instructions in the language's manuals.

To increase the educational effectiveness of programming education, students should develop grammaticality, predictability, appropriacy, transition and contextuality.

First, *grammaticality* is defined as the ability to judge whether a written program is grammatically correct. Computer programming languages, like natural languages, have a unique grammatical system. Therefore, it is necessary to check whether the program is written in accordance with the grammar of the programming language.

Second, *predictability* is defined as the ability to know the outcome of the program in advance. Students should be able to predict the flowchart or pseudo-code that was created before they wrote the program. In other words, students should be able to predict the results before the program is run.

Third, *appropriacy* is defined as the ability to correct errors found in the program and using the programming language to solve problems. Students should also be able to verify that the program accomplishes the purpose for which it was developed as well as optimize and improve the program's performance.

Fourth, *transition* is defined as the ability to identify and use the programming language necessary to solve real-life problems. Rather than learning and using a single programming language, students must be able to select a language and compare it with other languages to understand the common principles among the various languages.

Finally, *contextuality* is defined as the ability to understand and troubleshoot the program in problem situations. This requires students to develop programs while understanding the overall flow of the program instead of only its specific parts.

3. Development of the PUMA Model

The Preparing Un-coding Modifying Adding (PUMA) model is a communication-based pedagogy that allows students to freely communicate with a computer programming language using the internet. This model aims to nurture students' ability to communicate smoothly with their computers while modifying the sources of the completed program in small increments. Figure 1 describes the four phases of the PUMA model and the processes associated with each.

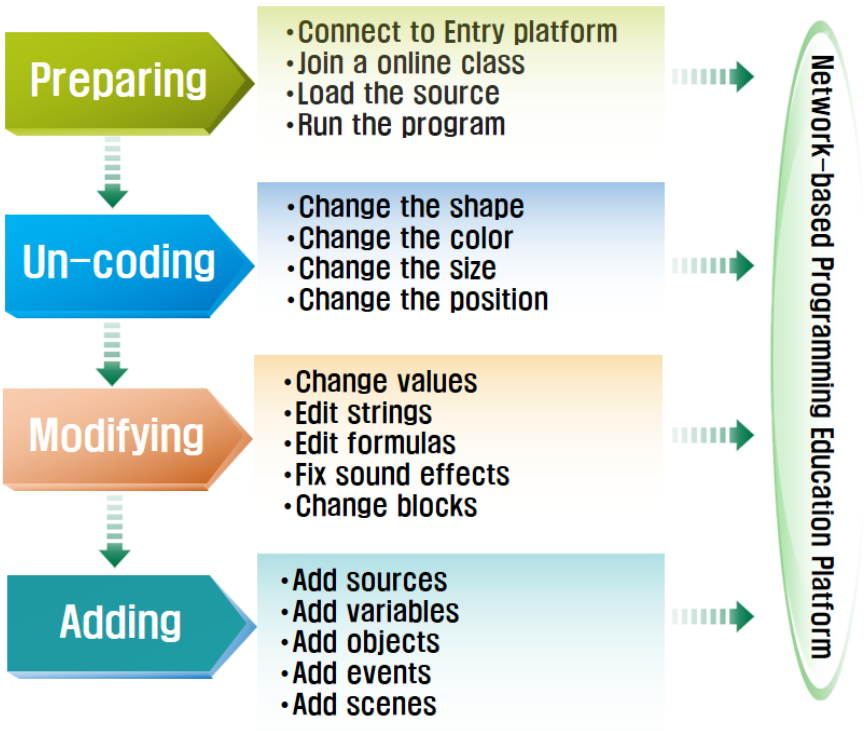

Figure 1. Phases and Processes of the PUMA model

As shown in Figure 2, the PUMA model can also be used to teach programming languages through an online platform for programming education by allowing students to share their programs and receive feedback from other students.

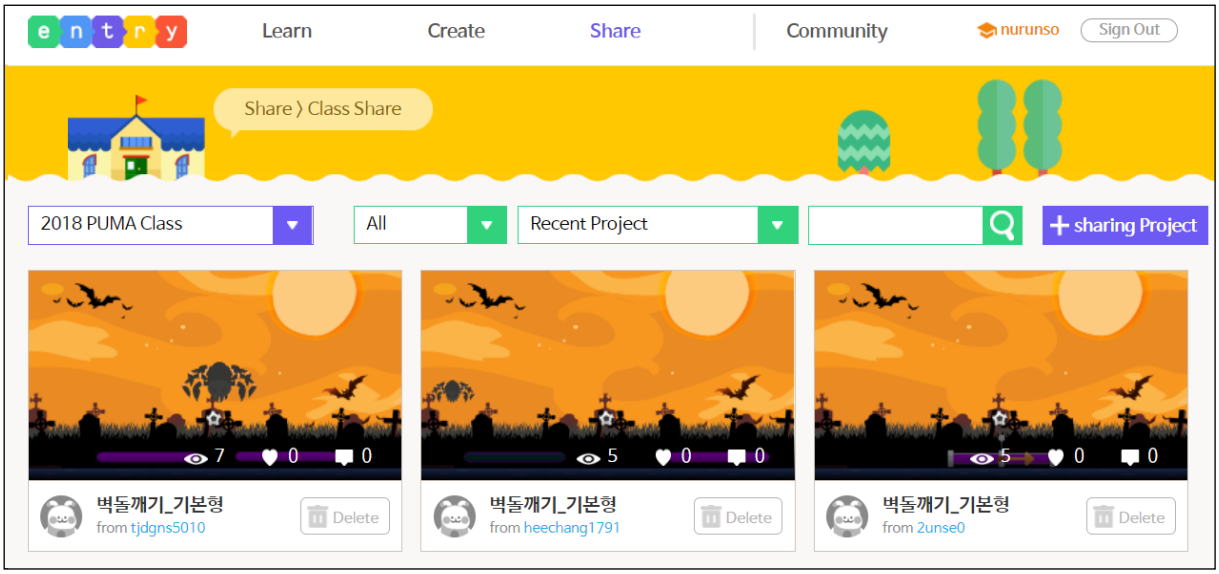

Figure 2. Sharing your work in an Entry platform

3.1. Preparing

In the Preparing phase, the program source connects to the network-based platform used for programming education, invokes the program source prepared by the teacher in advance, executes it, and confirms the results. As shown in Figure 3, we prepared a ‘break-brick’ game so that students can fully predict the program's running results and have fun while doing so.

We created and operated '2018 PUMA Class,' which allowed students and teachers to share their opinions while creating programs. Teachers can place incomplete programs in the online classroom and have students identified the program flow by letting them fill-in or replace specific phrases. Students can also compare completed programs with their classmates to identify problems and determine where improvements can be made.

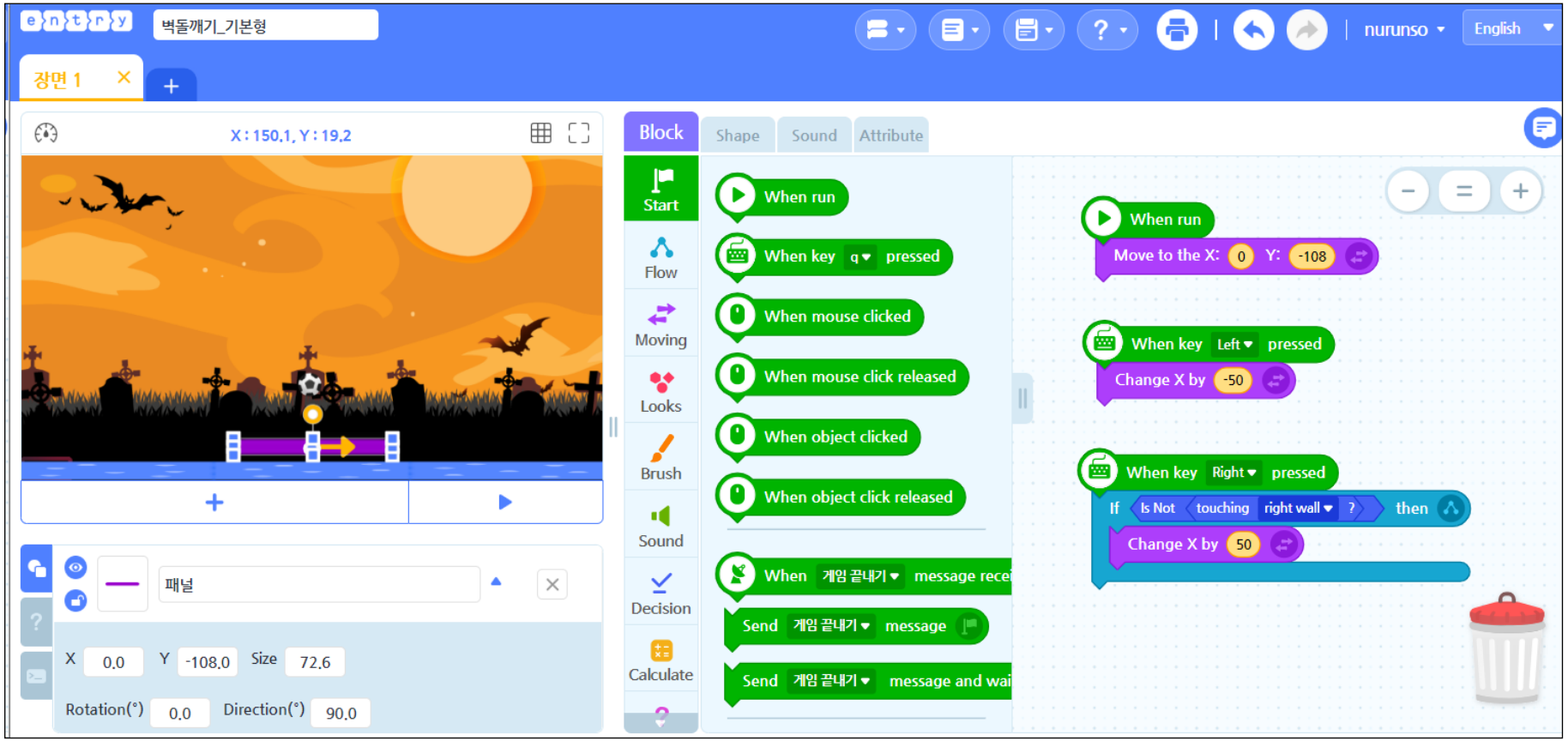

Figure 3. break- Brick game used in PUMA

3.2. Un-coding

In the Un-coding phase, students make modifications without programming. Instead, they can change the shape, background, image, size, color, and location of objects in the program through mouse clicks. Block-based programming tools also allow the size and color of images to be changed simply by using the mouse.

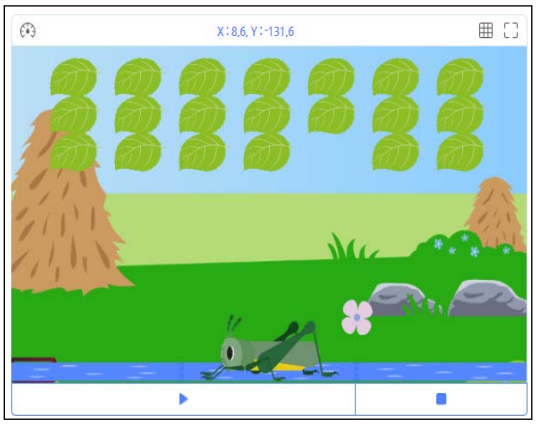

Figure 4. Change the shape with a mouse

As shown in Figure 4, we changed the object shape and size of the game to make a new game. This allows students to learn the features and functions of the programming language while freely using the various menus, icons, right-click menus, pop-ups, and balloons of the programming tool.

3.3. Modifying

The Modifying phase allows students to edit the source of the program as shown in Figure 5. At first, it is possible to grasp the functions of the command blocks in the program by changing numerical values (e.g., the number of repetitions) as well as the size and position of objects without using complex functions. The order of teaching in this phase is as follows:

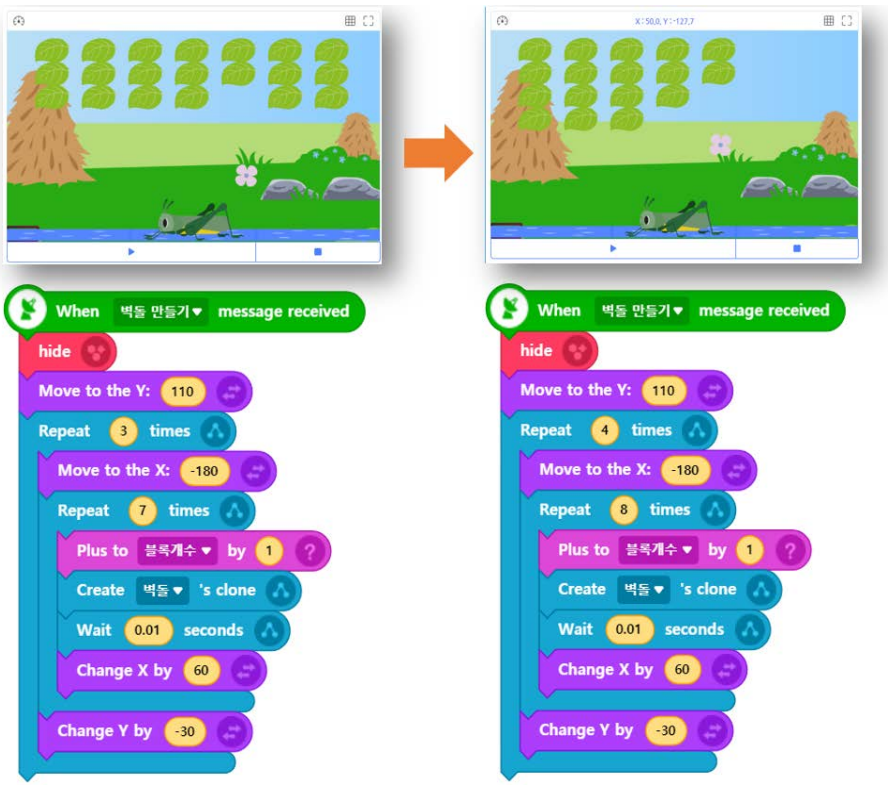

Figure 5. Modify codes in the Program

First, *change the speed of the ball*. The speed can be easily changed by anyone. These activities allow students to learn how to use the command blocks of Entry.

Second, *change the sound of the bouncing ball*. This allows students the opportunity to determine the relationship between objects, command blocks, and resources by changing or adding sound effects to objects.

Finally, *change the number of bricks*. In this step, students gain the ability to understand the structure of overlapping loop statements. At first, we changed the number of rows of bricks. Then we changed the number of columns so that we know the structure of the nested loop statements. By moving the location of the bricks, students learn how to use the programming tool's system.

3.4. Adding

The Adding Phase encourages students to develop new functions and fix inconveniences or errors in a given program. Examples of teaching in this phase are as follows:

First, adding an apple makes the panel longer when the apple touches the ball. In this process, students learn the need for infinite repetition, program events, and object properties.

Second, by creating a two-player game, as shown in Figure 6, it is possible for students to learn how to control the flow of the program using various input devices, such as a keyboard or a mouse.

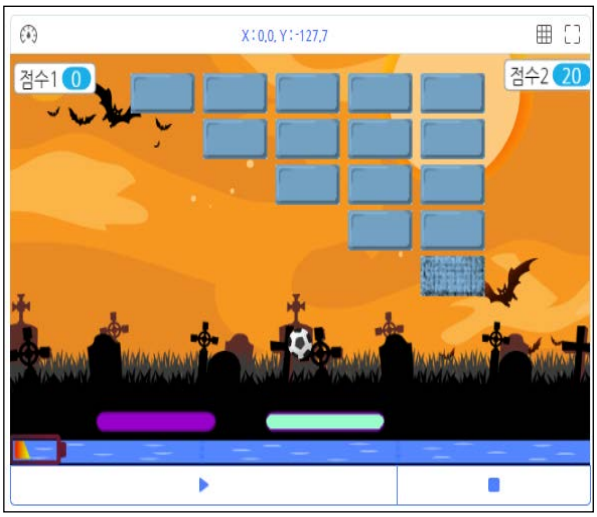

Figure 6. Create a game for 2 players

Third, programming a scoreboard, which creates a 'scoreboard' variable and increases the player's score by one point each time the bricks break. In this way, students learn how to use variables and how to put an initial value on those variables.

Finally, programming changes to the game. In this step, students can include items that change the background image when all the bricks are broken or add a special object to increase the player's energy when the ball touches it. They can also add an item that would allow bricks to break by launching missiles. By continuously changing the game, students can learn the functions of the command blocks and freely develop their programs.

In this way, using the network-based PUMA model allows students to naturally learn the grammar of programming languages and freely implement the desired functions while constantly changing the given program, even if they do not learn the grammar of the programming language separately.

4. Applying the PUMA Model

4.1. Subjects

For this study, two online surveys were distributed to student teachers who took lectures on Information Society and Computers at the National University of Education in March and June 2018.

In order to find the educational effectiveness of the network-based PUMA model, we conducted a six-week programming education course for 47 student teachers at the National University of Education. The 24 experimental groups participating in this study were given programming education using the network-based PUMA model while the 23 pedestrians in the control group used a traditional grammar-oriented model. The network-based programming language used by the experimental group is the Entry platform developed by the Connect Foundation.

4.2. Research Questions

The current study assumes that the PUMA model's applications use block-based programming languages. It is also based on the following research questions:

Q1: Is there a difference in information literacy between groups learning by grammar-oriented teaching methods and those learning by PUMA teaching methods?

Q2: Is there a difference in computational thinking between groups learning by grammar-oriented teaching methods and those learning by PUMA teaching methods?

Q3: Is there a difference in communication skills between groups learning by grammar-oriented teaching methods and those learning by PUMA teaching methods?

4.3. Survey Questionnaire

The questionnaire used in this study analyzed information literacy abilities, computational thinking, and communication skills by using the software education ability measuring tool developed by Seo and Jeong (2018), as shown in Table 2 [8][26].

The information literacy items inquired about information ethics, privacy protection, copyright, and ICT utilization ability. Alternatively, the computational thinking items inquired about problem analysis, data

analysis, abstraction, automation, and generalization. As shown in Table 1, the reliability of the questionnaire items was found to be 0.7 or more, and the reliability of the questionnaire items was judged to be appropriate.

Table 1. Reliability statistics of questionnaire

Area	Section	N	Cronbach's Alpha	
			Pre test	Post test
Information Literacy	Information Ethics	4	.903	.867
	Privacy Protection	4		
	Copyright	4		
	Utilization of ICT	10		
Computational Thinking	Problem Analysis	4	.907	.860
	Data Analysis	4		
	Abstraction	4		
	Automation	4		
	Generalization	4		
Communication Skills		3	.774	.964
Total		45		

Table 2. Respondents' background

Division	total	Groups		Gender	
		Ctrl.	Exp.	Male	Female
N	41	20	21	18	23
(%)	(100.0)	(48.8)	(51.2)	(43.9)	(56.1)
Division	Residence			Programming experience	
	Metro	City	Rural	No	Yes.
N	14	21	6	34	7
(%)	(34.1)	(51.2)	(14.6)	(82.9)	(17.1)
Exp.: experimental group, Ctrl.: control group					

The total number of students participating in the questionnaire was 41 as shown in Table 2. The control group was comprised of 20 students and the experimental group of 21 students. Men accounted for 43.9% (N = 18) and women 56.1% (N = 23) of the respondents. Only 17.1% of the students had programming experience. The remaining 82.9% had no programming experience.

4.4. Change in Information Literacy

As a result of the pre-test, the information literacy ability of the experimental group was 4.06 and the control group was 2.05, as shown in Table 3. The difference between the two groups was not significant. Even when examining the details concerning information literacy, the difference between the two groups remained insignificant.

Table 3. Differences of Information Literacy in Each Group at Pre-test

Division	Exp.		Ctrl.		t-test	F
	M	SD	M	SD		
Information Literacy	4.06	0.582	4.05	4.058	0.040	.030
Information Ethics	4.23	0.656	4.25	0.585	0.122	.057
Privacy Protection	3.95	0.701	3.89	0.750	0.286	.086
Copyright	4.01	0.691	3.89	0.737	0.558	.175
Utilization of ICT	4.04	0.670	4.18	0.571	0.704	1.172
P-value: *p<.05 Exp.: experimental group (n=21), Ctrl.: control group (n=20)						

As shown in Table 4, the information literacy ability of the experimental group was 4.29 and the control group was 3.81, which was higher than the control group ($t = 2.561$, $p < .05$).

Table 4. Differences of Information Literacy in Each Group at Post-test

Division	Exp.		Ctrl.		t-test	F
	M	SD	M	SD		
Information Literacy	4.29	0.588	3.81	4.289	2.561*	.112
Information Ethics	4.54	0.566	4.08	0.712	2.299*	.450
Privacy Protection	4.25	0.712	3.81	0.747	1.920	.074
Copyright	4.26	0.727	3.66	0.718	2.656*	.162
Utilization of ICT	4.11	0.644	3.70	0.683	2.000	.077
P-value: *p<.05 Exp.: experimental group(n=21), Ctrl.: control group(n=20)						

Table 5. Differences of Information Literacy in Experimental Group at Pre-test and Post-test

Division	Pre-Test		Post-Test		(2)-(1)	**t-test**
	M(1)	SD	M(2)	SD		
Information Literacy	4.06	0.582	4.29	0.588	0.23	1.326
Information Ethics	4.23	0.656	4.54	0.566	0.31	1.711
Privacy Protection	3.95	0.701	4.25	0.712	0.30	1.383
Copyright	4.01	0.691	4.26	0.727	0.25	1.247
Utilization of ICT	4.04	0.670	4.11	0.644	0.07	0.348
P-value: *p<.05 Exp.: experimental group (n=21), Ctrl.: control group (n=20)						

Table 5 shows results from the paired sample t-test we performed to compare the level of information literacy improvement in the experimental group. The results reveal that information literacy rates improved by 0.23, which is not statistically significant.

TAs shown in Figure 7, the experimental group showed higher post-test results than pre-test results. In contrast, the control group showed higher post-test results than pre-test results. Thus, we found that the PUMA model can help students maintain or improve their digital literacy.

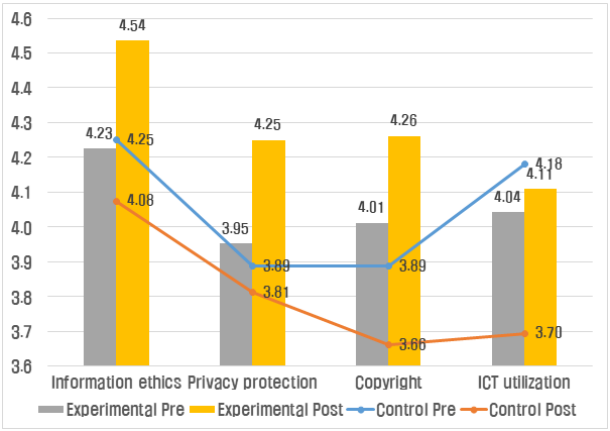

Figure 7. Differences of Information Literacy in Each Group at Pre-test and Post-test

4.5. Changes in Computational Thinking

As a result of the pre-test, the computational thinking power of the experimental group was 3.90 and the control group was 3.87 as demonstrated in Table 6. The difference between the two groups was not significant. Furthermore, in the detailed domain of computational thinking, we found the difference between the two groups to also be insignificant.

Table 6. Differences of Computational Thinking n Each Group at Pre-test

Division	Exp.		Ctrl.		t-test	F
	M	SD	M	SD		
Computational Thinking	3.90	0.522	3.87	3.905	0.207	.591
Problem Analysis	3.94	0.680	4.09	0.608	0.728	.098
Data Analysis	4.10	0.594	4.11	0.615	0.091	.232
Abstraction	3.86	0.589	3.98	0.720	0.575	.377
Automation	3.75	0.729	3.40	0.856	1.412	.167
Generalization	3.88	0.579	3.76	0.656	.614	.231
P-value: *p<.05 Exp.: experimental group (n=21), Ctrl.: control group (n=20)						

As a result of the post-test, as shown in Table 7, the experimental group's computational thinking was 4.15, and the control group was 3.89. Although the results were not statistically significant, this indicates that the experimental group had higher levels of computational thinking than the control group. However, the experimental group (4.36) was significantly higher than the control group (3.84) only in their ability for problem analysis ($t = 2.138$, $p < .05$).

To compare the degree of improvement in computational thinking for the experimental group, a paired sample t-test was performed. This test revealed that computational thinking ability improved by 0.24 as shown in Table 8, but it was not statistically significant. However, the statistical significance of problem analysis ability and abstraction ability among the sub-domains of computational thinking increased.

Table 7. Differences of Computational Thinking in Each Group at Post-test

Division	Exp.		Ctrl.		t-test	F
	M	SD	M	SD		
Computational Thinking	4.15	0.582	3.89	4.148	1.270	.589
Problem Analysis	4.36	0.589	3.84	0.936	2.138*	1.119
Data Analysis	4.33	0.588	4.09	0.722	1.198	.198
Abstraction	4.37	0.640	4.03	0.711	1.629	.082
Automation	3.62	1.027	3.60	0.908	0.063	.059
Generalization	4.06	0.987	3.90	0.868	.549	.019
P-value: *p<.05 Exp.: experimental group(n=21), Ctrl.: control group(n=20)						

Table 8. Differences of Computational Thinking in Experimental Group at Pre-test and Post-test

Division	Pre-Test		Post-Test		(2)-(1)	t-test
	M(1)	SD	M(2)	SD		
Computational Thinking	3.90	0.522	4.15	0.582	0.24	1.901
Problem Analysis	3.94	0.680	4.36	0.589	0.42*	2.500
Data Analysis	4.10	0.594	4.33	0.588	0.24	1.493
Abstraction	3.86	0.589	4.37	0.640	0.51*	2.761
Automation	3.75	0.729	3.62	1.027	0.13	0.500
Generalization	3.88	0.579	4.06	0.987	0.18	0.929
P-value: *p<.05 Exp.: experimental group(n=21), Ctrl.: control group(n=20)						

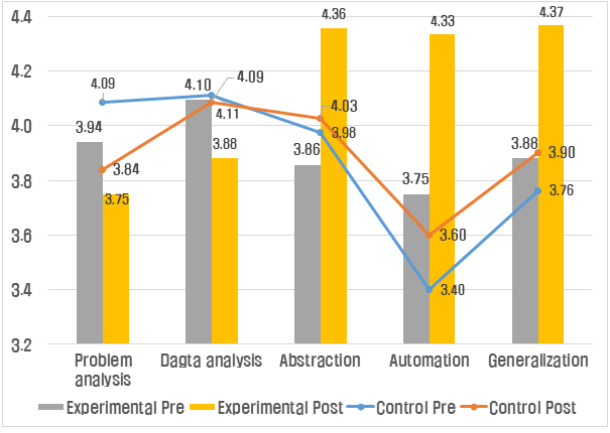

Figure 8. Differences of Information Literacy in Each Group at Pre-test and Post-test

The results captured in Figure 8 show that the experimental group's post-test results were higher than the pre-test results whereas the control group has higher pre-test results than post-test results. Thus, we found that the PUMA model can help students maintain or improve computational thinking.

4.6. Change in Communication Ability

As shown in Table 9, the experimental group saw significant improvement ($t = 2.256$, $p < .05$), but the control group's improvement was not statistically significant.

Table 9. Differences of Communication Skills in Experimental Group at Pre-test and Post-test

Division	Pre-Test		Post-Test		(2)-(1)	t-test
	M(2)	SD	M(1)	SD		
Experimental group	4.14	.619	4.48	.602	.333	2.256*
Control group	4.31	.643	4.04	1.020	.275	1.608
P-value: *p<.05						

5. Discussion

We developed a network-based PUMA model that allows students to understand, modify, and create new programs by using communication-oriented programming education instead of grammar-oriented programming education. By using the network-based PUMA model, students can create programs on online platforms, such as Entry, share them with classmates or teachers, and solve problems using their programs. The PUMA model consists of four steps: preparing, un-coding, modifying, and adding.

In order to analyze the educational effects of the network-based PUMA model, we conducted a six-week demonstration class for student teachers at the National University of Education. The results are summarized as follows.

First, using the network-based PUMA model shows that the experimental group's information literacy ability was higher than the control group in the post-test. However, it was not improved compared to the results of the pre-test. In other words, the PUMA model does not improve information literacy; rather, its current status is maintained.

Second, the network-based PUMA model improves the ability to analyze problems and the capacity for abstraction among sub-areas of computational thinking. It can be assumed the PUMA model improves these areas because they are necessary for grasping not only the whole flow of the program but also predicting what function a specific command block will perform during program development.

Third, the network-based PUMA model improved communication ability. Before beginning the class, participants in the control group had less ability for communication than members of the experimental group; however, when using the PUMA model, students' communication ability improved. This is because the PUMA model allows students to exchange ideas among themselves and with their teacher(s) using an online platform. In sharing their work and hearing the opinions of others, the students' communication skills improved.

In conclusion, for programming education to be communication-oriented rather than grammar-oriented, it is necessary to continuously improve the network-based PUMA model developed in this study. It is also worth noting that because the PUMA model allows students to connect through the internet, any further studies must take into consideration the demand for data privacy in both industry and academia. Therefore, we must also strive to protect and promote information privacy [27].

REFERENCES

[1] Korea Ministry of Education, Curriculum revision of the practical arts subject, 2015.

[2] Ministry of Education, Practical arts (technology / home economics) / Information science curriculum, Ministry of Education Notice No. 2015-74 [Separate 10], 2015.

[3] E. J. Oh, and Y. S. Jeong, "A case study of T-PUMA pedagogy using traditional fairy tales," The Korean Association of Information Education Research Journal, vol. 9, no. 2, pp. 163-168, Apr. 2018.

[4] Y. S. Jeong, "Problems and improvements of SW education policy in elementary schools," The Korean Association of Information Education Research Journal, vol. 9, no. 1, pp. 91-97, Jan. 2018.

[5] J. M. Kim, W. G, Lee, I. G. Yoon, and J. H. Seo, "Development of competency diagnostic tool for elementary and secondary school students," Korean Education and Research Information Service, 2017.

[6] Y. S. Jeong, "The direction of SW education in the era of the 4th industrial revolution," NIA intelligence research series, National Information Society Agency, 20" 17.

[7] Y. S. Jeong, "The Problems and Improvement of the SW Education Policy in Elementary School," The Korean Association of Information Education Research Journal, vol. 9, no. 1, pp. 91-98, Jan. 2018.

[8] H. S. Seo, and Y. S. Jeong, "Educational Effects of Collaborative Story Creation Activities Using the Entry Programming Language," Journal of The Korean Association of Information Education, vol. 22, no. 6, pp. 651-660, Dec. 2018.

[9] Y. S. Jeong, J. S. Yu, J. S. Lim, and Y. K. Son, Theory of Software Education. Seoul, Korea: Cmass, 2015.

[10] S. H. Kim, and Y. S. Jeong. "Exploring a Method of Physical Programming Education using Story-based PUMA Model," The Korean Association of Information Education Research Journal, vol. 9, no. 2, pp. 117-123, Aug. 2018.

[11] Partnership for 21st Century Learning (2018) Framework for 21st century learning. [Online]. Available: http://www.p21.org/storage/documents/docs

[12] S. K. Yeom, "A case study of elementary school students' English education through e-learning project". Honam University, MA, 2009.

[13] S. H. Kim, and Y. S. Jeong, "A Study on Improvement of

Interactive e-Learning Model for Elementary Students in Small Classes," The Korean Association of Information Education Research Journal, vol. 10, no. 1, pp. 125-130, Jan. 2019.

[14] Korea Ministry of Science, ICT and Future Planning, Practical arts (Technology & Home Economics)/ Information Curriculum, 2011.

[15] Jeannette M. Wing (2014) Computational Thinking Benefits Society Social Issues in Computing. New York: Academic Press. [Online]. Available: http://socialissues.cs.toronto.edu/index.html%3Fp=279.html

[16] Jeannette M. Wing, Computational Thinkin, Our CS Workshop Carnegie Mellon University, 2011.

[17] Y. S. Jeong, S. B. Shin, and Y. H. Sung, "A Study of the Connection between Achievement Criteria and Computational Thinking in the Areas of Algorithms, Programming and Robotics, and Computing," Journal of The Korean Association of Information Education, vol. 21, no. 1, pp. 105-114, Feb. 2017.

[18] Lupu, C. The Relationship between Types of Intelligence and Study Profile Options. American Journal of Education and Learning, vol. 2, no.1, pp. 23-33, 2017.

[19] Masciantonio, T. A., & Berger, P. D. Is Alumni Salary an Appropriate Metric for University Marketers?. Journal of Social Economics Research, 5(1), 1-9.

[20] Ahmed, U., Zin, M. L. M., & Majid, A. H. A. Impact of Intention and Technology Awareness on Transport Industry's E-service: Evidence from an Emerging Economy. Journal of Industrial Distribution & Business Vol, 7(3), 13-18, 2016.

[21] Mertoglu, M. (2018). Happiness Level of Teachers and Analyzing Its Relation with Some Variables. Asian Journal of Education and Training, vol. 4, no.4. pp, 396-402, 2018.

[22] Miyaji, I. Comparison of Useful Activities of Improving Awareness in Blended Classes in Java Script and PHP Programming. International Journal of Educational Technology and Learning, vol.3, no .2. pp, 78-92, 2018.

[23] Mokhtar, S. B. Teaching-Learning Model of Islamic Education at Madrasah Based on Mosque in Singapore. International Journal of Asian Social Science, vol.7 no. 3 .pp, 218-225, 2017.

[24] Mujtaba, M., & Jamal, S. Enhancing Work Climate to Improve the Perceived Performance Leading to Talent Retention-A Study of Pakistani Service Sector. International Journal of Social Sciences Perspectives, vol. 3 no. 1. pp, 21-33, 2018.

[25] Okendo, O. E. Strategies Used to Enhance Parent Involvement on Performance of Early Childhood Education in Kisii, Nyamira and Homabay Counties, Kenya. International Journal of Educational Technology and Learning, vol. 4no. 1.pp, 1-7, 2018.

[26] Korea Ministry of Education, 2015 Management Guideline of Software Education, 2015.

[27] Haseeb M, Abidin IS, Hye QM, Hartani NH. The impact of renewable energy on economic well-being of Malaysia: Fresh evidence from auto regressive distributed lag bound testing approach. International Journal of Energy Economics and Policy. Vol. 7, no. 9. 269-75, 2018.

[28] Y, G, Kim, K. S. Kim, J. H. Kim, H. S. Kim, J. Y. Yang, S. J. Lee, J. Y. Jeong, H. J. Choi, and K. H. Chae, Developmnet of the Software education guideline, Korean Education and Research Information Service, 2015.

[29] P. J. Denning, "The Great Principles of Computing", The Scientific Research Society. Sep-Oct, p. 371, 2010.

[30] M. J. Kim, "The Impact of Information Education on Elementary Students Using Scratch Programming", Mokpo University, MA, 2017.

[31] Entry training materials (2018) Block images and help. [Online]. Available: https://playentry.org/tt#!/basic/materials

[32] Barefoot Computing (2016) Computational Thinking, [Online]. Available: http://www.elearning.edu.my/ASKKSSM/Bahan/What%20is%20computational%20thinking.pdf

[33] E. J. Oh, and Y. S. Jeong, "The Effects of Programming Education using G-PUMA Teaching and Learning Model," The Korean Association of Information Education Research Journal, vol. 10, no. 1, pp. 111-116, Jan. 2019.

[34] Y. S. Jeong, "A study on the educational competency of information and communication technology at national universities of education," The Korean Association of Information Education Research Journal, vol. 5, no. 2, pp. 115-120, 2014.

[35] Jermsittiparsert K, Atsadamongkhon A, Sriyakul T. Politics in the Process of Establishing Local Development Plan: Case Study of the Lan Tak Fah Subdistrict Administrative Organization, Nakornchaisri, Nakornpathom, Thailand. Review of European Studies. Vol. 1, no. 7:75-89. 2015.

[36] J. H. Youn , Y. H. Seo , and M. S. Oh , "A Study on UI Design of Social Networking Service Messenger by Using Case Analysis Model," Journal of Information and Communication Convergence Engineering, vol. 15, no. 2, pp. 104-111, 2017.

[37] M.A. Abbas, and J.P. Hong, "Survey on Physical Layer Security in Downlink Networks", Journal of Information and Communication Convergence Engineering, vol. 15, no. 1, pp. 14-20, 2017.