An Education Model for Coding and Software to Improve Computational Thinking

Heeseon Jang, Pyeongtaek University, Korea

The IAFOR International Conference on Education - Hawaii 2019
Official Conference Proceedings

Abstract
The regular coding (programming of software) education in elementary, middle and high school has been begun in Korea since 2018. In this paper, a new coding education model to improve the computational thinking which is critical in software development is proposed. In addition to, 645 questionnaire survey for teachers, programmers, and students is analyzed, and its results are reflected on the proposed model. The model consists of following steps; *i*) problem definition and understanding of mathematical concept, *ii*) problem solving and algorithm design, *iii*) Raptor flowchart development, and *iv*) understanding of source code. Note that the Raptor is the free software tool based on visual flowchart of international standard organization (ISO-97N90), in which the defined problem can be solved with visual tool, and the source code can be easily generated by using Raptor menu. From survey results, it is observed that total of 93% of the respondents is shown as the positive opinion for the usefulness of the Raptor. Under statistical analysis (Chi-square test), we also observe that the experienced respondents for coding show the more positive opinion for the Raptor rather than those of inexperienced. The another results show that the middle school is the proper beginning time to study, and to improve logical thinking capability is the most important factor in coding education.

Keywords: Raptor, coding education, computational thinking

Introduction

In the upcoming era of the fourth industrial revolution, the new products and services associated with artificial intelligence, internet of things, virtual or augmented reality, robot, *etc* will emerge in our everyday life (Jang & Kim, 2017). And, in that age, the main technology to make the products and services is the software engineering. For this reason, the coding educations have been operated in the regular curriculum courses all around the world. Until now, main countries have operated the coding education, so in our country Korea, since 2018, the coding education has been begun in the elementary, middle, and high school. However, in the most cases, a rule-of-thumb like coding education has been managed with relatively easy block-coding tool (Code studio, Entry, Scratch), even without the formal educational model. After all, that way of education will raise useless programmers in the fields of software development. In the future fourth industrial revolution, we need software developers with the capabilities of logical thinking and creativity to solve the defined complex problems. In this paper, to raise the desirable software developers, we propose a coding education model to improve computational thinking based on the processes defined in the regular programming education course (Hadfield *et al*., 2018) of US Air Force Academy (USAFA). The model consists of the following four steps; (1) problem definition and understanding of mathematical concept, (2) problem solving and algorithm design, (3) Raptor flowchart development, and (4) understanding of source code. In particular, at the 3rd step, the Raptor platform is proposed to design procedural algorithm and make the source code. The Raptor platform is the software to make the visualized flowchart developed in USAFA using the international standard ISO-based symbols (ISO-97N90), and it is widely used for the education of algorithmic reasoning, coding, and basic programming (Cheng, 2013; Gaddis, 2015; Venit & Drake, 2015). In the following section, at first, the coding education model is introduced, and an example (giving Letter Grade for numeric score) by using the Raptor platform and block-coding is presented. Then, the survey results for 645 questionnaire are analyzed with the questions for the Raptor's usefulness, proper coding educational tool (or programming language), advisable education timeliness, and main emphasis on the education. Finally, using SPSS statistical package (Noh & Jeong, 2006), the frequency analysis and the test for significance are performed between the experienced and inexperienced respondents. These results indicate the observations such that, when we teach or learn the coding by using the proposed education model along with the previous block-coding and programming language, the efficient and desirable coding education will be realized to improve computational thinking capability.

Coding education model

In general, as shown in Figure 1, the problem solving process (Gaddis, 2015) using computer is as follows; (1) algorithm design, (2) coding, (3) implementation on devices after analyzing the problem or customer's needs, and finally the products or services are developed as the shape of applications, games, ICT (information and communication technology) services, toys, *etc*. For the process, the education to make products or services is called as the maker education (coding and devices implementation), and coding education includes the training for the procedures of algorithm design and coding methods. From Figure 1, we observe that the academic area of science and mathematics can be applied in the algorithm design, and

technology and engineering domain is involved in the coding and device implementation tasks. On the other hand, as defined in the programming regular course in USAFA, to teach the algorithm design and coding procedures in Figure 1, the following four steps are defined; (1) understanding the goal to be accomplished, (2) design your solution, (3) implement your solution, and (4) test your solution (Hadfield *et al*., 2018).

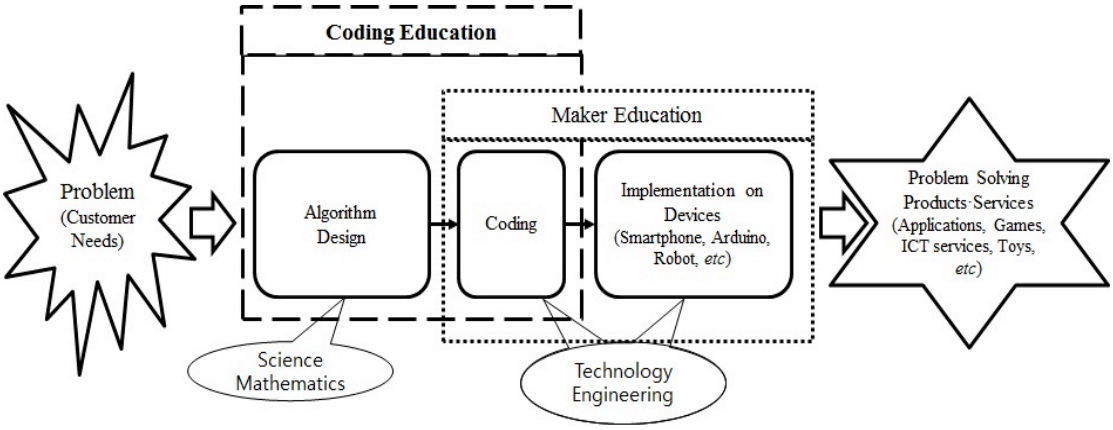


Figure 1. Problem solving process by computer

Based on the problem solving strategy explained in the textbooks of USAFA programming course, we propose the coding education model as shown in Table 1. The model is composed of 4 steps, and in particular, in the 3rd step, the Raptor platform can be efficiently used to develop visualized flowchart based on the procedural mathematical concepts, and generated source code by menu in Raptor is studied after modifying minor syntax errors.

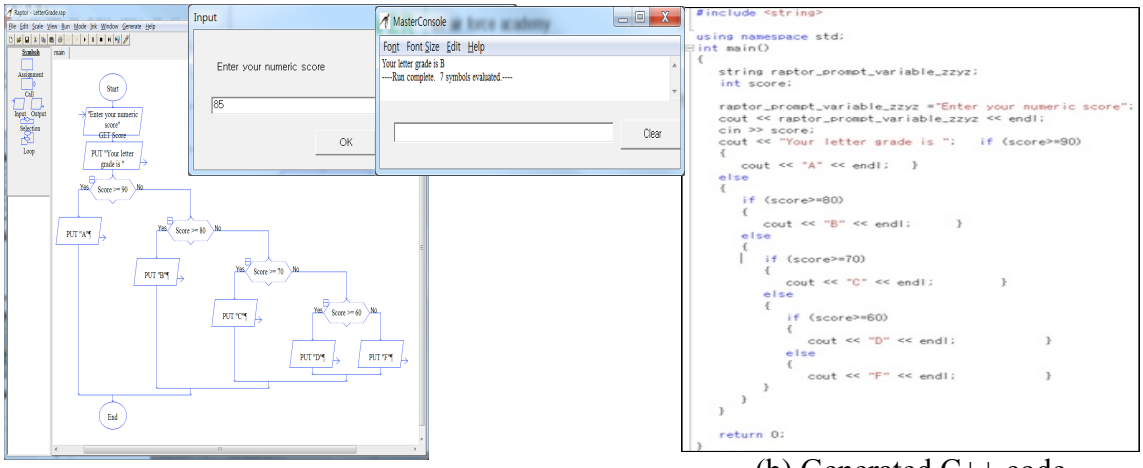
Table 1. Coding education model

Process	Outline
Problem definition and understanding of mathematical concept	Define the descriptions and requirements of the problem. Make an answer for any conflicting questions and requirements. Understand the mathematical concept required to solve the problem. Make a storyboard to describe what the coding must solve.
Problem solving and algorithm design	Make a list of tasks that need to solve the defined problem. If needed, break complex tasks down into smaller sub-tasks. Design each algorithm to solve the sub-task. Make appropriate sequential, conditional, and iterative control logic.
Raptor flowchart development	Raptor flowchart is developed to solve the sub-task. Test the each implemented small part of flowchart. Run the whole flowchart from [Start] to [End] step by step. Using various inputs to make sure the flowchart does what we want.
Understanding of source code	Using the Raptor [Generate] menu, make source code. Correct syntax error, run, and understand the source code.

The Raptor platform is a flowchart-based programming environment, designed specifically to help students visualize their algorithms and avoid syntactic baggage, and it can be freely downloaded at http://raptor.martincarlisle.com (maintained by professor, Martin Carlisle).

Letter grade example

For example, we present the letter grade problem introduced in Hadfield *et al*. (2018), and the Raptor flowchart and block-coding design steps in the proposed process are only explained, excluding the steps of problem definition and algorithm design that are described in the reference in details. Figure 2 shows the letter grade flowchart and C++ code developed by using Raptor menu.



(a) Raptor flowchart (b) Generated C++ code

Figure 2. Raptor flowchart for letter grade

As shown in the Figure, if we are assigning a letter grade (A, B, C, D or F) based on a numeric score (variable, [Score]), we need to select one between five choices. That is, the letter grades are assigned such that, put "A" for numeric [Score] ≥ 90, "B" for 80 ≤ [Score] < 90, "C" for 70 ≤ [Score] < 80, "D" for 60 ≤ [Score] < 70, and put "F" in case [Score] is less than 60 (that is, [Score] < 60). In case of [Score] = 85, we take that the first "No" branch. Because we are on the "No" branch, we know that [Score] < 90. So, we need only test that [Score] ≥ 80 for the next selection decision in the flowchart. Therefore, as shown in the results in Figure 2, for [Score] =85, the second "Yes" branch is taken, letter "B" grade will be put. This makes for an elegant and efficient algorithm to put letter grade given the numeric score.

The scratch block-coding for the letter grade is as shown in Figure 3. Using the blocks of sensing, looks, data, control, and operators, the letter grade is determined. The Figure 4 also shows the entry block-coding and python code generated by entry's menu. The interface, menu, block name, and design process in entry are quite similar as scratch. However, as compared with the scratch, the entry provides the python code while we can't get any code in scratch.

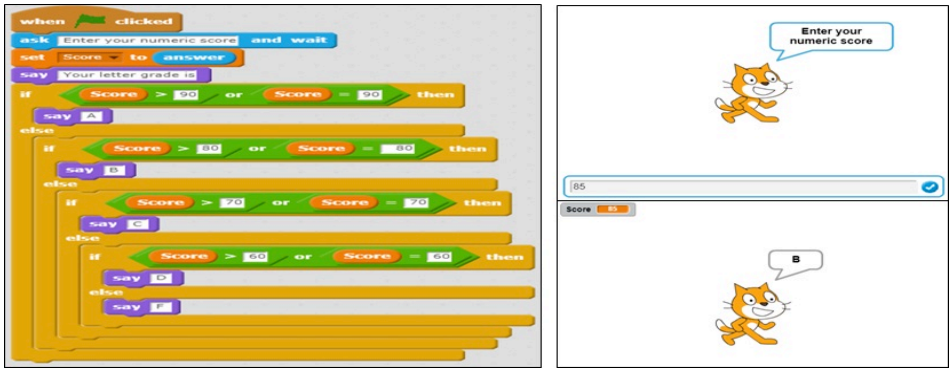


Figure 3. Scratch block-coding for letter grade

Figure 4. Entry block-coding for letter grade

Comparing those tools of Raptor and block-coding, we observe that

(1) Not easy to understand and explain blocks (for example, the functions implemented in each block) to the students or programmers. On the other hand, the algorithm developed by the Raptor is relatively easy to understand, since the Raptor provides the visualized procedural function by using three symbols of logics, sequence, selection, and loop defined in structured programming method.
(2) We couldn't get any programming source code from scratch, while the entry tool mainly used in Korea provides the python code. However, in the Raptor platform, the source code such as C++/C#, Java, VBA (visual-basic applications), and Ada are generated by Raptor's [Generate] menu.
(3) The block-coding has its limitations to use the means as communication tool among learners, teachers, customers, software developers. But, we can use the Raptor flowchart as efficient communication tool since it is developed by using international standard ISO symbols.

From the observations, in the development step before coding tasks in the proposed model, using the Raptor flowchart is more desirable to express its algorithm, and is more successful in creating algorithms than using a traditional language (and block-coding) or writing flowcharts without Raptor. The other examples developed by Raptor can be found in Gaddis (2015), Hadfield *et al*. (2018), Jang (2018), Jang *et al*. (2017), Venit & Drake (2015), Raptor and Cosuda sites. In the next section, the survey results for the usefulness of Raptor including proper coding education tool, timeliness and main emphasis on education are analyzed.

Survey results

Our survey for 645 respondents is performed for students (72%), teachers (professors, 17%), researchers (8%), and programmers (3%), and its ratio for gender is male 68%, and female 32%, respectively. And, the ratio of the respondents who have a coding education or experience to develop software is 58% (experienced), and 42% for inexperienced people. The analytic result for the reliability of the questionnaire indicates that the value of Chronbach's alpha is 0.743, so it then shows the very high reliability (Noh & Jeong, 2006), and we analyze the survey by confidence level of 99% and allowable standard error of ± 5%. At first, the Figure 5 shows the proper coding education tool for each course. From the frequency (ratio) analysis, it is observed that;

(1) In the elementary school, the most desirable tool is evaluated as block-coding method (43.4%) such as scratch, entry, and code.org.
(2) For the middle school, the block-coding is also suitable. However, the ratio of block-coding is decreased to 15.7%, instead of it, C/C++ 15%, physical computing method 11.8% such as Arduino and raspberry are then selected.
(3) In the high school, as different results, C/C++ 26.8%, Java 16.6%, and python 9% are recognized as the appropriate educational language. In addition to, the ratio of visual-basic (9%) are the same as python's ratio.
(4) Finally, in the university or college, as shown in the similar order of priorities in the high school, C/C++ 22%, Java 17.5%, python 11% are perceived as proper language. Note that the adoption ratio (5.3%) of visual-basic is decreased as compared with the high school.

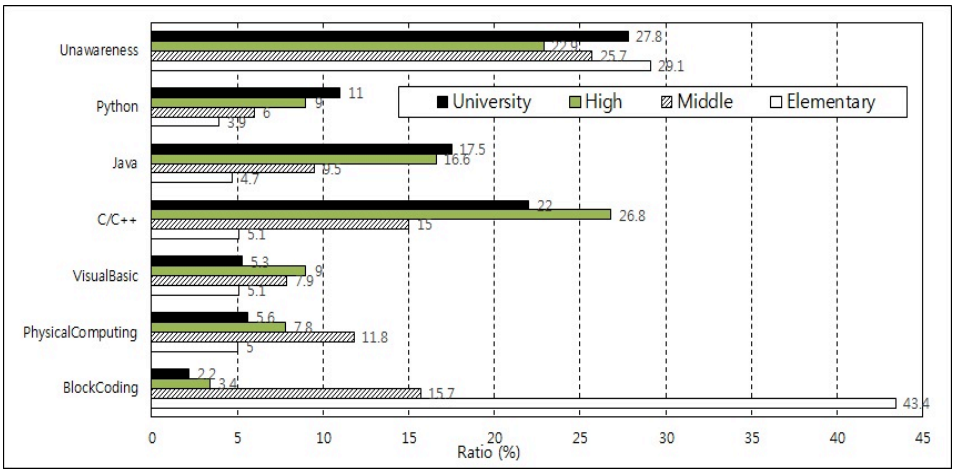


Figure 5. Proper coding education tool (programming language)

To compare the observations with the programming languages used in the industry's fields for software development, we investigate the TIOBE index and Redmonk data which are presented in each company's site. The Figure 6 shows TIOBE index, representing the utilization ratio of language in the market and software development. The results of Java 16.8%, C 14.4%, C++ 8.3%, Python 7.7% are shown in the Figure. Also, as shown in the annual results from 2001 through 2018, the order of priority with overall evaluation in recent trend are C/C++, Java, and then Python.

Nov 2018	Nov 2017	Change	Programming Language	Ratings	Change
1	1		Java	16.746%	+3.51%
2	2		C	14.396%	+5.10%
3	3		C++	8.282%	+2.94%
4	4		Python	7.683%	+3.20%
5	7	^	Visual Basic .NET	6.490%	+3.58%
6	5	v	C#	3.952%	+0.94%
7	6	v	JavaScript	2.655%	-0.32%
8	8		PHP	2.376%	+0.48%
9	-	^^	SQL	1.844%	+1.84%
10	14	^^	Go	1.495%	-0.07%
11	19	^^	Objective-C	1.476%	+0.06%
12	20	^^	Swift	1.455%	+0.07%
13	9	vv	Delphi/Object Pascal	1.423%	-0.32%
14	11	v	R	1.407%	-0.20%
15	10	vv	Assembly language	1.108%	-0.61%
16	13	v	Ruby	1.091%	-0.50%
17	12	vv	MATLAB	1.030%	-0.57%
18	15	v	Perl	1.001%	-0.56%
19	18	v	PL/SQL	1.000%	-0.45%
20	17	v	Visual Basic	0.854%	-0.63%

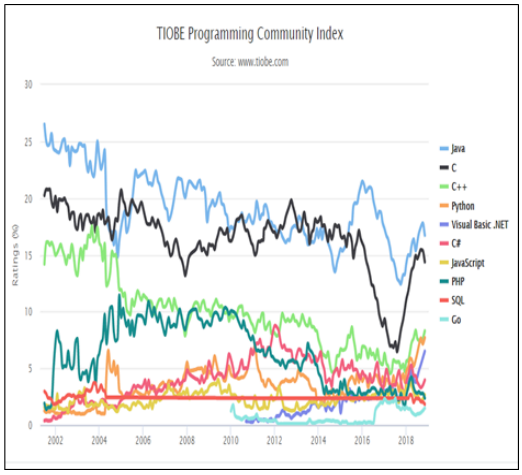


Figure 6. Utilization ratio of programming language (TIOBE index, November 2018)

We also observe that, from the Redmonk rankings in Figure 7, the order is as follows; Javascript, Java, Python, PHP, C#, C++, CSS, Ruby, C and Objective-C. The rankings in Redmonk company are extracted by analyzing the GitHub (number of projects) and stack overflow (number of tags), in which from the company's notes, it is observed that the ranking is not to offer a statistically valid representation of current usage, but rather to correlate language discussion and usage in an effort to extract insights into potential future adoption trends. It is also observed from the Redmonk's data that the programming languages (C#, C++, and Objective-C) for C type are popularly used, and then Javascript, Java, Python follow next. As compared with the results in the TIOBE index, the Javascript is included in the Redmonk rankings because the Javascript is mostly adopted to provide Web and mobile services as the interpreter language. By evaluating the results of TIOBE index and Redmonk rankings, we observe that the priority order used in the software industries is the same as the desirable coding education tool, and it is C/C++, Java, and Python.

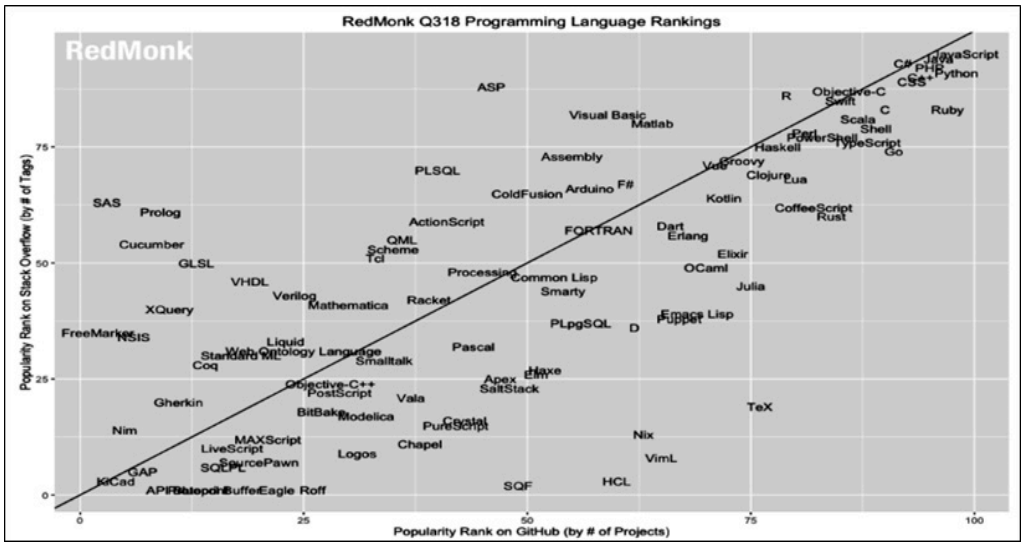


Figure 7. Programming language rankings (Redmonk, June 2018)

In addition to, to examine the deviation of the opinions for the coding education tool between the experienced and inexperienced respondents, the significance analysis by Chi-square test is performed with the null hypothesis (H_0: Selection trend is the same for experienced and inexperienced respondents). The result of Chi-square test for the

preference of coding educational tool indicates that the significance probability (p = 0.0) is less than the previously determined significance level (α = 0.05), so the H_0 is rejected, and there is proven to be the significant preference deviation between the responses for experienced and inexperienced people. For the preference of the educational coding tool, the results (ratio) for each group are shown as Figures 8 and 9, for elementary/middle and high/university schools, respectively. From the Figure 8, the most preferable tool is the block-coding method in the elementary and middle schools. At first, in the elementary school, the block-coding is mostly adopted for the experienced (greater than 50%) and 33% for inexperienced respondents. The results of elementary school indicate that, as the second coding tool, the selection ratios for physical computing, visual-basic, C/C++, and Java have the similar results (5%). In the middle school, even though the preference deviation for the block-coding are shown as quite small (13%~20%), the experienced people of year 1~16 mostly choose the physical computing and C/C++ as the second preferable tool. Unusually, the adoption ratio for the block-coding in the middle school for the inexperienced respondents is also decreased to 15% from 33% in the elementary school. As the natural results, note that the unawareness ratio for the inexperienced people is the highest among the respondents.

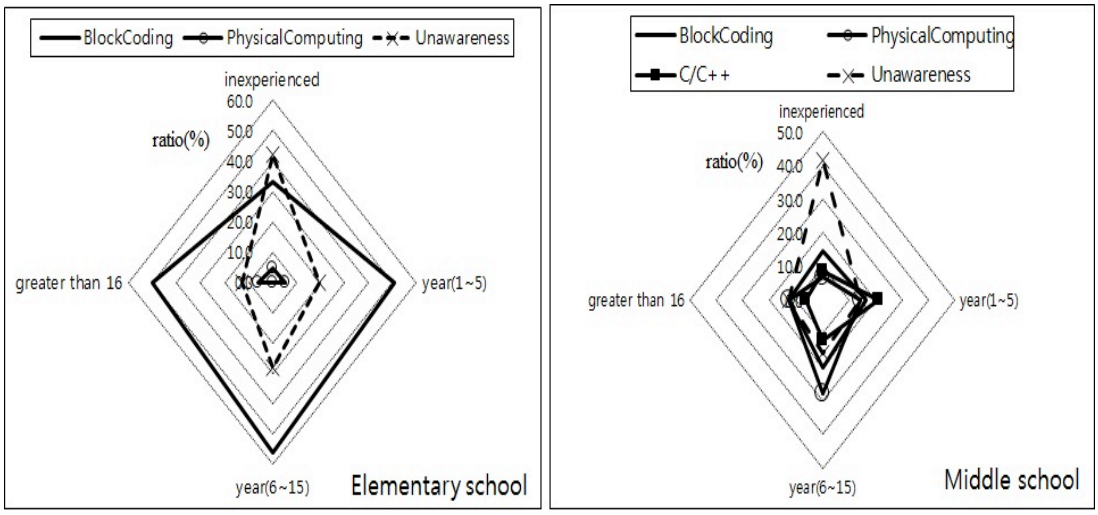


Figure 8. Preferable educational tool (Elementary and middle school)

For high and university (or college), as shown in Figure 9, the experienced respondents with career greater than 1 year mostly prefer the C/C++ language as desirable tool rather than other programming languages. In high school, 16%~44% of experienced people adopts the C/C++, Java (12%~21%), and then selects the visual-basic and python in similar ratio (6%~24%). Under the frequency analysis for all respondents, note that the selection ratio of visual-basic and python is the same (9%). Similarly, in the university or college, the experienced people select the language as same order as high school, C/C++ (25%~63%), Java (12%~22%), and python (8%~14%). However, the python's adoption ratio (11%) is overall greater than the ratio (5.3%) of visual-basic. On the other hand, by evaluating the responses for inexperienced people excluding the unawareness opinions (Note that the unawareness ratio for the inexperienced is the highest), the priority order is the same as C/C++ (14%), Java (13%), python (8%) in university, and C/C++ (19%), Java (12%), visual-basic (8%), python (7%) in high school.

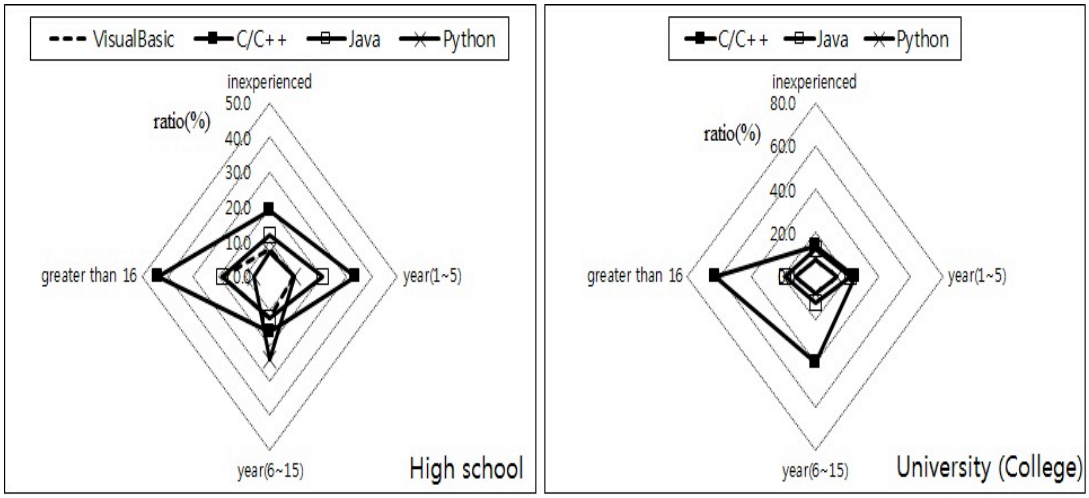


Figure 9. Preferable educational tool (High and university school)

Next, the results to investigate the usefulness of visual tools such as Raptor are as shown in Figure 10. From the results, it is observed that the respondents mostly have the positive agreements for the effectiveness of visual tools. For learning the Raptor flowchart, the responses are as shown as very good 10.4%, good 50.1%, average 32.2%, and overall positive opinions summing up the ratio are evaluated to be 93%. If the students could learn the UML (unified modeling language), IBM RSA (rational software architecture), and LG MDD (model driven development) visual tools in the school, it is very effective because those tools are practically used to develop the software. However, the visual tools are very expensive to learn, and another time and cost are also required to study the complex interface, function, and menus. On the other hand, the Raptor flowchart can be used free to download in Internet, and to easily learn when you only study the ISO-based basic 6 symbols. Note that the 6 symbols defined in the ISO are assignment, selection, loop, call, input and output. In conclusions, by using the proposed coding education model combining with the Raptor platform and other tools (block-coding and programming languages), the desirable coding education will be realized to improve computational and logical thinking capabilities.

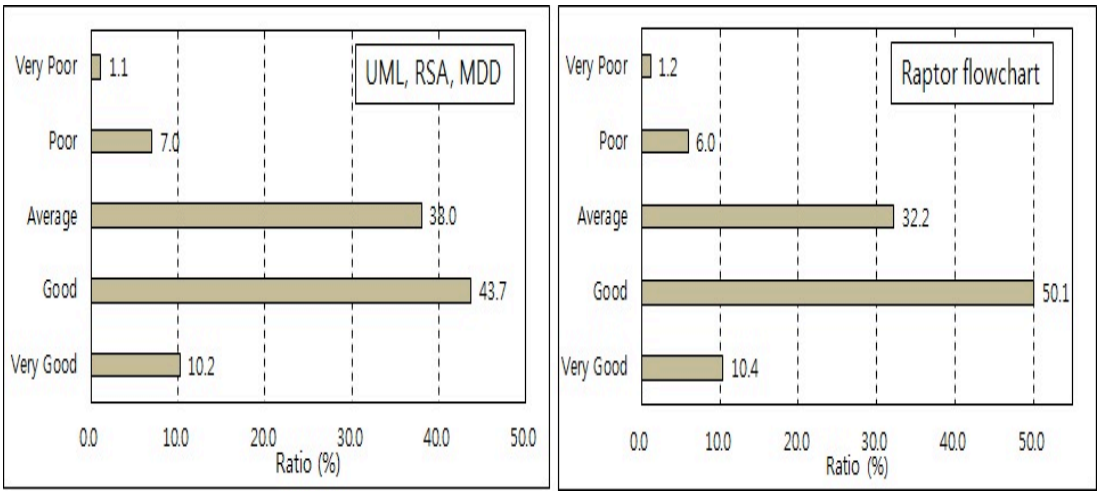


Figure 10. Usefulness of visual tools

The Chi-square test results show that, $p = 0.269 > \alpha = 0.05$, so the response trend is the same for the coding experience in the opinions for the effectiveness of UML, RSA, and MDD visual tools. In the Figure 10, we know that the positive view (92%) for the visual tools is much higher than the negative ratio (8%). However, the test results ($p = 0.0$) for the Raptor platform show the significant deviation in the usefulness. As shown in the Figure 11, the ratio (20%) of the very good is the highest in the career's people of 6~15 years, and the experienced respondents of 1~5 years mainly show the good's opinions (54%). And, for the inexperienced people, the total ratio of very good and good is 54%, and ratio above average is 95%. However, the negative views (31%) for both poor and very poor for the programmers greater than 16 years is much higher than as compared with those of other people. Note that the ratio above average for the 16 years' respondents is 69%. From those survey results, we observe that, even though the person with lots of experience has a rather negative view on the Raptor flowchart coding education, the most respondents express favorable opinions for the Raptor tool.

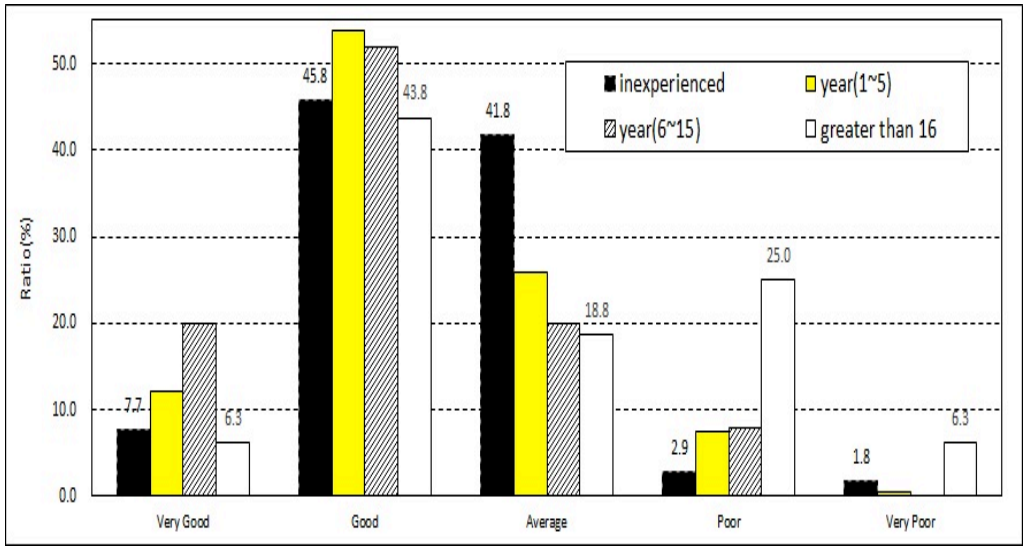


Figure 11. Usefulness of Raptor platform

Then, the Figure 12 shows survey results for the desirable timeliness of coding education. For the question of the proper beginning time to learn coding in the regular curriculum, the respondents select mostly the middle school (45%), and then elementary 23.4%, high school 19.1%, and university 8.5%. The results indicate that the proper educational coding tool in the elementary school is block-coding method, which is widely known easy to learn as compared with the other programming languages. The Chi-square test results show the $p = 0.0 < \alpha = 0.05$, so null hypothesis (H_0) is then rejected, and the selection trend is not the same among the careers of respondents. As shown in the Figure 12, we know that more than 50% of programmers with careers greater than 16 prefer the elementary school for early coding education. However, considering the results such that the inexperienced respondents give an opinion for the middle rather than elementary school, we conclude the middle school as the desirable coding education timeliness. Note that 4% of respondents shows the unnecessary views for the coding education, and they are mostly inexperienced people.

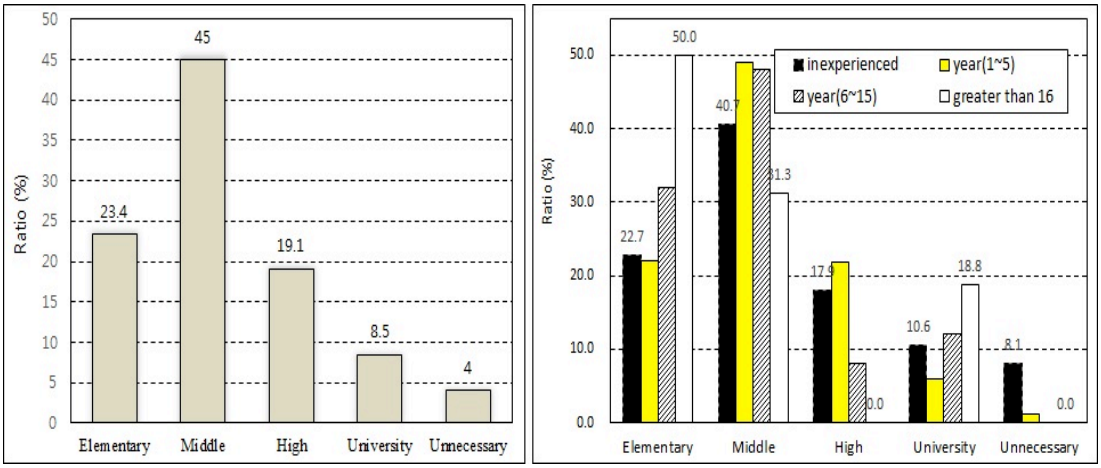


Figure 12. Desirable beginning time of coding education

Finally, through the survey results for the question, "What is the main emphasis on coding education?", as shown in Figure 13, we observe that the logical thinking is mostly selected as the ratio of 28.2%, and then problem solving skill 24.5%, creativity 24.2%, coding syntax 14.9%, and to-follow codes written in text 1.2%. The Chi-square test results show the $p = 0.053$ larger than the significance level ($\alpha = 0.05$), so H_0 is then accepted, concluding that the selection trend is not different between coding experiences. The Figure 13 indicates that, regardless of careers, the logical thinking capability is the most important factor in teaching coding. Therefore, using the Raptor platform can be very useful when we teach and learn coding in the manner of understanding procedural flowchart based on the mathematical algorithmic reasoning.

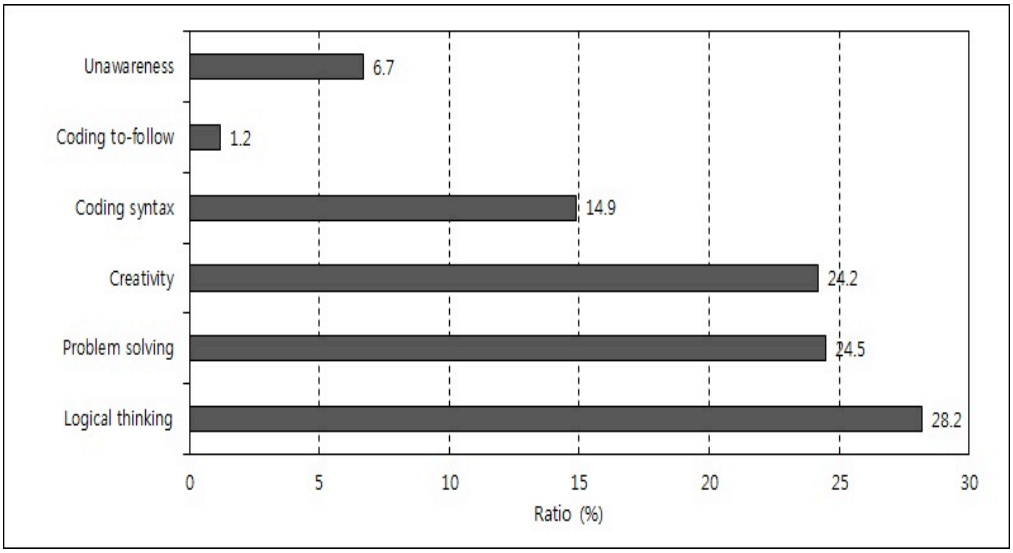


Figure 13. Main emphasis on coding education

Conclusions

Globally, the early coding education has been becoming the important main factor to develop the new products and services in the upcoming fourth industrial revolution era. By complying with this world's trend, our country Korea also has initiated the coding education in the regular curriculum courses of elementary, middle, and high school. However, in most cases, the teachers teach the coding by using relatively easy

block-coding or physical computing tools, without the education model, in the end, to raise useless manpower in the software fields. We need the software developers with logical thinking capability and creativity to solve the emerging complex problems in the fourth industrial revolution age. In this paper, a coding educational model was proposed to improve computational thinking. The model was developed based on the processes presented in the programming education course of the US Air Force Academy (USAFA). The steps in the model were defined as follows; (1) problem definition and understanding of mathematical concept, (2) problem solving and algorithm design, (3) Raptor flowchart development, and (4) understanding of source code. In particular, at the 3rd step, to improve the computational thinking, we propose the Raptor platform, that is a flowchart-based programming environment, using international standard ISO-based symbols, designed specifically to help students visualize their algorithms avoid syntactic baggage. Raptor programs are created visually and executed visually by tracing the execution through the flowchart, where the required syntax is kept to a minimum. Students prefer using flowcharts to express their algorithms, and are more successful in creating algorithms using Raptor than using traditional language or writing flowcharts without Raptor. To verify the usefulness of the Raptor, and investigate the proper coding tool in each educational course, timeliness of coding education, and main emphasis on education, questionnaire survey for 645 people were analyzed by using SPSS package. From the results, it was observed that the respondents mainly have the positive opinions (93%) for the effectiveness of Raptor education, in particular, the experienced peoples of 1~5 years mostly show the higher good's views as compared with inexperienced or lots of experiences greater than 16 years. Through the survey results for proper education timeliness, the respondents select mainly the middle school (45%), on the other hand, under the Chi-square significance test, it was observed that the people with more careers give the opinion on the earlier education in elementary school rather than inexperienced respondents. Finally, for the question of the main emphasis on coding education, regardless of experiences, the most important factor is the logical thinking. The results indicate that the Raptor platform can be used to improve computational logical thinking by implementing the visualized procedural flowchart based on the mathematical algorithmic reasoning. In conclusions, using the coding educational model proposed in this paper, when we teach and learn the Raptor flowchart along with the previous block-coding tool and programming languages, the efficient coding education can be realized to improve computational thinking.

Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (Grant Number: NRF-2017R1E1A1A03070134). The author would like to thank the professor, Martin Carlisle at Carnegie Mellon University whose efforts to develop the Raptor platform. Also thank to the professor, J.H. Baek at Chonbuk National University to give research motivations.

References

Cheng, X.Q. (2013). *Visualized Computation* (可視化計算). Tsinghua University Press.

Code studio: https://code.org.

Cosuda contents: http://sites.google.com/view/cosuda.

Entry block-coding: https://playentry.org.

Gaddis, T. (2015). *Programming logic and design* (4th ed.). PEARSON.

Hadfield, S., Weingart, T., & Brown, W. (2018). *An introduction to programming and algorithmic reasoning using Raptor*. (1st ed.). CreatSpace (an Amazon.com Company).

Jang, H.S. (2018). Mobility modeling and analysis in mobile communication networks, *The 10th international conference on ubiquitous and future networks*, 641-643, Czech Republic.

Jang, H.S., Seo, J.Y., & Baek, J.H. (2017). Analysis of location area residence time in mobile cellular networks, *Far East Journal of Electronics and Communications*, 17(4), 761-774.

Jang, H.S., & Kim, D.C. (2017). Research and development of mathematical algorithm curriculum for coding education in industry 4.0 era, *Proceedings of 2017 international conference of joint societies for mathematics education: KSME, KSESM, Singapore NIE (KSME policy committee for mathematics education)*, 549-553, Korea.

Noh, H.J., & Jeong H.Y. (2006). *SPSS from basic to applications*. Hyeongseul Press.

Raptor platform by Martin Carlisle: http://raptor.martincarlisle.com.

Redmonk data: https://redmonk.com.

Scratch block-coding: https://scratch.mit.edu.

TIOBE index: https://www.tiobe.com.

Venit, S., & Drake, E. (2015). *Prelude to programming concepts and design* (6th ed.). PEARSON.