Journal of Information Technology Education: Innovations in Practice

An Official Publication of the Informing Science Institute InformingScience.org

JITEiip.org

Volume 19, 2020

RUBRIC FOR MEASURING AND VISUALIZING THE EFFECTS OF LEARNING COMPUTER PROGRAMMING FOR ELEMENTARY SCHOOL STUDENTS

Daisuke Saito*	Waseda University, Tokyo, Japan	d.saito@fuji.waseda.jp
Shota Kaieda	Waseda University, Tokyo, Japan	skaieda@fuji.waseda.jp
Hironori Washizaki	Waseda University, Tokyo, Japan	washizaki@waseda.jp
Yoshiaki Fukazawa	Waseda University, Tokyo, Japan	fukazawa@waseda.jp

* Corresponding author

ABSTRACT

Aim/Purpose	Although many computer science measures have been proposed, visualizing individual students' capabilities is difficult, as those measures often rely on specific tools and methods or are not graded. To solve these problems, we propose a rubric for measuring and visualizing the effects of learning computer programming for elementary school students enrolled in computer science education (CSE), which is independent of the programming language being used.
Background	In this research, we proposed a rubric based on existing CSE standards and criteria having a programming education-learning goal. We then applied this rubric to actual lessons to visualize the educational effects.
Methodology	The proposed new rubric for teaching computer programming, based on existing standards and criteria, was applied to fourth- and sixth-grade students in Japan. We assessed which skills were cultivated through quizzes before and after the teaching.
Contribution	This paper contributes on how to make and utilize a rubric for programming education in computer science. We evaluated and visualized the proposed rubric's learning effects on children and found that our proposed rubrics are independent of any particular method or tool.

Accepting Editor Janice Whatley │ Received: June 24, 2020 │ Revised: August 18, September 15, October 18, October 19, November 11, 2020 │ Accepted: November 12, 2020.

Cite as: Saito, D., Kaieda, S., Washizaki, H., & Fukazawa, Y. (2020). Rubric for measuring and visualizing the effects of learning computer programming for elementary school students. *Journal of Information Technology Education: Innovations in Practice, 19,* 203-227. https://doi.org/10.28945/4666

Findings	The results of this survey are twofold: (1) we proposed a rubric of programming education in computer science, independent of the programming tools used and (2) we succeeded in visualizing students' learning stages by applying the proposed rubric to programming education conducted in a Japanese elementary school.
Recommendations for Practitioners	Evaluating educational effects in CSE is important. In particular, graded assessments of learner abilities can reveal individual characteristics. This research is useful for assessing CSE because it focuses specifically on programming education.
Recommendations for Researchers	The rubric's suggestions and quality improvements in CSE help learners assess their learning progress and will clarify the cultivated computer science skills.
Impact on Society	This research evaluates CSE based on a rubric in the programming education field.
Future Research	Future work is needed to improve the proposed rubric's quality and relevance. Also, this rubric should be applied to many classes to increase the number of evaluations and analyses.

Keywords	computer science education, programming education, rubric

INTRODUCTION

Evaluating computer science education (CSE) learning effects for children is crucial for improving educational outcomes. This has led to the development of many CSE learning design and evaluation indicators such as the Computer Science Teachers Association (CSTA) K-12 Computer Science Standards (CSTA, 2017). Other studies have developed creative computer science rubrics (Catеté et al., 2016; Grover et al., 2018). However, these indicators have evaluation items that are either not divided into stages or depend on specific tools and methods. Additionally, many educators also wish to evaluate the learning effect over time when employing some tools and methods. To cope with these problems, we propose a learning evaluation stage indicator (rubric) for learning computer programming in CSE for children. In the present study, the proposed rubric evaluates items related to programming in CSE for children step by step, regardless of tools or methods. This study's research questions (RQs) are as follows:

- RQ1: Can evaluation criteria be proposed independently of tools and methods?
- RQ2: Can the proposed rubric evaluate and visualize learning effects?
- RQ3: In what ways does the proposed rubric based on Bloom's taxonomy and structure of the observed learning outcome (SOLO) taxonomy, correctly evaluate students' learning of programming?

RQ1 assesses the feasibility of designing evaluation criteria independently of the tools or learning environment used. Thus, tools and methods should provide a fair and consistent evaluation. In contrast, RQ2 assesses whether learning computer programming can be evaluated and visualized independent of tools and methods. Herein, we applied a learning computer programming rubric (PLR) to a programming class planned by an elementary school teacher. Furthermore, we used the proposed PRL to analyze, evaluate, and visualize the learning effects of children learning computer programming. In RQ3, we examined the design of rubrics in learning computer programming to see whether they are useful for assessing learning and setting learning goals at learning stages based on SOLO taxonomy and Bloom's taxonomy. We also examined whether these goals and assessment stages were set correctly in learning computer programming. This will allow us to judge whether the students' learning of computer programming is assessed correctly. We also hope that the solution for RQ3 will improve the usefulness of rubrics based on the educational taxonomy for learning computer programming in computer science.

Our proposed rubric focuses on learning computer programming in computer science, and it does not need to be customized for any particular method or tool; therefore, our proposal is novel. The proposed rubric is intended to guide the assessment of learning computer programming. Furthermore, our evaluation method is generic and does not rely on a specific method or tool.

The remainder of this paper is structured as follows. The next section explains the background of our research. Materials and Methods presents our rubric design. Application of PLR describes the rubric's application experiments. Results and Discussion, respectively, present and discuss the results of applying the rubric. The following section briefly discusses related works to place this work in perspective. Lastly, Conclusion summarizes the study, and the following lists future works and limitations of this paper.

The present work is an extension of a previously published study of the rubric (Saito et al., 2019). In previous studies, the proposed rubric was given only as an overview. In addition, it does not apply to actual school lessons. Therefore, the present work presents a detailed proposal for a rubric and application of the lesson to elementary schools.

BACKGROUND

EVALUATION STANDARDS AND RUBRICS

The most common CSE evaluation standards are the CSTA K-12 Computer Science Standards (CSTA, 2017), which assess programming, algorithms, cyber security, and other computer science learning goals. Other evaluation standards include the International Society for Technology in Education (ISTE) Standards for Students (Permitted Educational Use) (ISTE, 2016) and the Computer Science K-12 Learning Standards (Office of Superintendent of Public Instruction, 2018). Many of the evaluation items in these standards can be applied to learning computer programming However, the issue of these standards includes many elements in one item. For example, item 1B-AP-10 of the CSTA K-12 Computer Science Standards is "Create programs that include sequences, events, loops, and conditionals" (CSTA, 2017). This item contains multiple evaluation points such as sequences, events, loops, and conditionals, obscuring the perspective from which the learner achieves the goal and the evaluator may interfere in the evaluation. Furthermore, their learning goals are not stepwise but rather are evaluated as 0 (satisfied) or 1 (not satisfied). Therefore, the learning effects on individuals cannot be evaluated in detail. Thus, we divide the existing standards' evaluation items into stages, allowing detailed evaluations.

Rubrics are indicators that help to assess student outcomes by describing and defining a description of goals and levels of achievement (Stegeman et al., 2016). Rubrics are commonly used to evaluate learning computer programming in CSE and other similar fields. For example, the rubric study of Mustapha et al. (2016) considered the differences in evaluation caused by variation between evaluators, and this rubric is useful for evaluating programming courses in higher education. Cateté et al. (2016) proposed a rubric to evaluate a curriculum opted by non-CS majors called The Beauty and Joy of Computing, which targets third-year high school students to first-year university students. Their rubric borrows ideas from the brick wall concept, creating a brick wall by programming. Using block-based programming artifacts, Grover et al. (2018) performed a rubric-based analysis to assess what students learned about programming. The rubric in their analysis followed the CSTA K-12 Computer Science Standards and others. Further, Basu (2019) proposed a multidimensional evaluation rubric for block-based programming in open-ended projects. Alves et al. (2020) also developed a rubric based on the CSTA K-12 Computer Science Standards. Using this rubric, they evaluated the algorithms and programming concepts of App Inventor applications. They also allocated automated assessments and successfully achieved consistency in the evaluations. However, these rubrics are tailored to specific tools or methods, whereas educators prefer a common rubric suitable for many learning computer programming assessments. Therefore, instead of proposing a rubric for a specific tool or method, we propose a versatile rubric on which many tools and methods can be evaluated from the same perspectives.

TAXONOMY

Here, we describe two taxonomies of pedagogy: The Structure of the Observed Learning Outcome (SOLO) taxonomy (Biggs & Collis, 2014) and Bloom's taxonomy (Bloom et al., 1984). Both taxonomies are useful for designing our PLR for elementary school students.

SOLO taxonomy

The SOLO taxonomy (Biggs & Collis, 2014) is a classification table that categorizes learning outcomes into different levels of complexity. Previous studies have demonstrated the SOLO taxonomy's effectiveness in learning computer programming (Lister et al., 2006; Whalley et al., 2006). An evaluation framework that evaluates the complexity of programming quizzes using SOLO has also been proposed (Izu et al., 2016). This taxonomy classifies prior knowledge, motivation, and thinking methods on what will eventually be learned into five stages:

- Pre-structural: Learners do not understand the content.
- Uni-structural: Learners understand one aspect of the content.
- Multi-structural: Learners independently understand multiple aspects of the content.
- Relational: Learners understand the relationships and structural combinations between multiple aspects of the content.
- Extended abstract: Learners can generalize the content, understand it from many different perspectives, and create new ideas.

Our study adopts the SOLO taxonomy for rubric grading.

Bloom's taxonomy

Bloom's taxonomy is well known in education. The six cognitive stages of the original Bloom's taxonomy were knowledge (level 1), comprehension (level 2), application (level 3), analysis (level 4), synthesis (level 5), and evaluation (level 6) (Bloom et al., 1984). These have been revised by Krathwohl (2002), for example, to remember (level 1), understand (level 2), apply (level 3), analyze (level 4), evaluate (level 5), and create (level 6). Bloom's taxonomy is also widely used in learning computer programming (Selby, 2015; Thompson et al., 2008). We used Bloom's taxonomy for setting the learning objectives in our study.

MATERIALS AND METHODS

DESIGN OF RUBRIC FOR LEARNING COMPUTER PROGRAMMING

Our PLR evaluates CSE learning achievement stages using student deliverables, quizzes, and questionnaires. Our PLR is based on the SOLO taxonomy, and achievement at the determined learning stage is evaluated from deliverables and quizzes. Additionally, learning goals are referenced to Bloom's taxonomy. Figure 1 provides an overview of the rubric's design. The rating categories and items are based on existing metrics. Each stage and learning item in Figure 1 was assigned using SOLO and Bloom's taxonomies and by referring to previous research (Lister & Leaney, 2003). The designed rubrics are provided in Appendix A, Tables A1-A7.

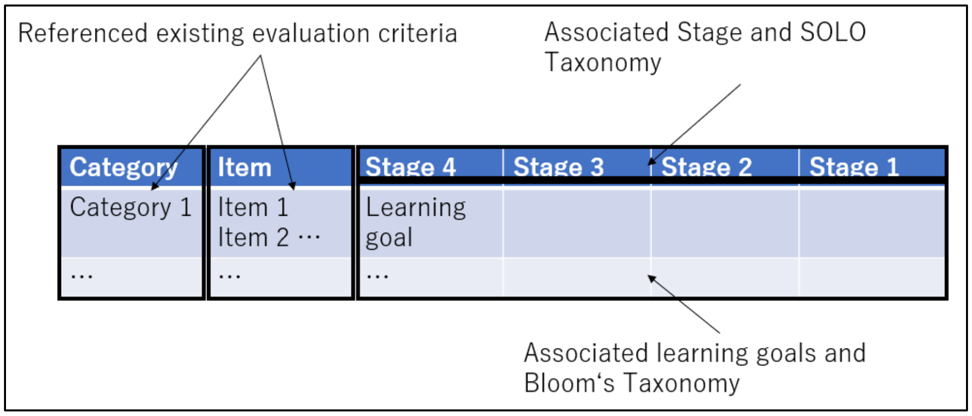

Referenced existing evaluation criteria
Associated Stage and SOLO Taxonomy
Category
Item
Stage 4
Stage 3
Stage 2
Stage 1
Category 1
Item 1
Item 2 …
Learning goal
…
Associated learning goals and Bloom's Taxonomy

Figure 1. Composition of the PLR

Evaluation categories and items

The PLR consists of 30 evaluation items, each of which is assigned to 1 of 8 categories. We determined the evaluation items by referencing existing metrics such as the CSTA K-12 Computer Standards (CSTA, 2017) and the ISTE Standards for Computer Science Educators (ISTE, 2016), as well as other standards (Cateté et al., 2016; Grover et al., 2018; Office of Superintendent of Public Instruction, 2018). Table 1 lists the evaluation items and their categories. This rubric belongs to the CSE's learning computer programming category and assesses a learning computer programming sequence. Specifically, students initially learn basic programming concepts and computers. Next, they think about creating programs and then write programs. Lastly, they confirm the correct execution of their program and share their work with others. The PLR covers all of these stages. Although the "Self-Regulation" and "Cooperation with Others" categories are not directly related to programming, these abilities are covered in programming classes and are thus included in the PLR.

Table 1. Evaluation categories and items in the proposed rubric

Category	Item
Attitude	Positivity, Interest, Toughness
Programming Concept	Sequence, Loop, Conditional
Construction of Computer	Construction of Computer
Designing Programs	Subdivision, Analysis, Extraction, Construction and Functionalization, Generalization, Design Document, Expression, Creativity
Creating Programs	Use of Programming Concepts, Logical Thinking, Use of Software, Programming Language, Data Expression, Use of Formula
Read, Edit, and Evaluate Programs	Read, Edit, and Evaluate
Self-Regulation	Plan, and Safety Considerations
Cooperation with Others	Announce Own Idea, Understand Other's Idea, Cooperate Programming, Contribute to Group Work

Four developmental stages

The PLR specifies each evaluation item's learning goals. Dividing these learning objectives should provide a detailed assessment of learning achievements. To this end, we divided and set the learning goals based on the SOLO and Bloom's taxonomies.

The PLR's evaluation items are divided into four developmental stages, with 4 representing top achievement and 1 representing failure. Stages 1, 2, 3, and 4 are analogous to the SOLO taxonomy's pre-structural, uni-structural, multi-structural, and relational and extended abstraction stages, respectively, and students are expected to satisfy specific requirements at each stage. Furthermore, each stage's learning targets are referenced to the revised Bloom's taxonomy. The six cognitive stages of Bloom's taxonomy (remembering, understanding, applying, analyzing, evaluating, and creating) classify educational goals and are thus useful for setting learning goals. Stages 2, 3, and 4 in our PLR are associated with the bottom two stages (remembering and understanding), the middle two stages (applying and analyzing), and the top two stages (evaluating and creating) of Bloom's technology, respectively. Table 2 presents some of the learning objectives of each item in the PLR's four stages.

Table 2. Developmental stages in the proposed rubric

Category	Item	Stage 4	Stage 3	Stage 2	Stage 1
Programming Concept	Sequence	Learners understand sequential execution, can read the program in order from beginning to end, can write a program of sequential execution, and can construct a program to be sequentially executed without requiring assistance.	Learners understand sequential execution, can read the program in order from beginning to end, and can write a program of sequential execution.	Learners understand sequential execution and can read the program in order from beginning to end.	Learners do not understand sequential execution.
	Loop	Learners understand loops, can read looped programs, and find loops by themselves. They can also incorporate loops into programs.	Learners understand loops, can read looped programs, and can write loop programs.	Learners understand loops and can read looped programs.	Learners do not understand how loops are used in programs.
	Conditional	Learners understand conditional branches, can read programs containing conditional branches, and can incorporate conditional branches into programs.	Learners understand conditional branches, can read programs containing conditional branches, and can write programs containing conditional branches.	Learners understand conditional branches and can read programs containing conditional branches.	Learners do not understand conditional branching.

Correspondence with items in the existing standards

We mapped the proposed PLR's categories to the CSTA K-12 Computer Science Standards and ISTE Standards for Students items in the existing standards, making it possible to conduct classroom evaluations in conjunction with existing standards. Table 3 shows the corresponding items.

Table 3. Correspondence of PLR categories with existing standards

PLR	CSTA	ISTE
Attitude		Innovative Designer(d)
Programming Concept	1A-AP-08, 1A-AP-10, 1B-AP-08, 1B-AP-10	
Construction of Computer	1A-CS-02, 1A-CS-03, 1A-DA-05, 1B-CS-01	
Designing Programs	1A-DA-07, 1A-AP-11, 1A-AP-12, 1B-AP-13	Computational Thinker (a, c, d) Innovative Designer (a, d) Knowledge Constructor (b, d)
Creating Programs	1A-CS-01, 1A-AP-15, 1B-AP-08, 1B-AP-10, 1B-AP-12	Innovative Designer (b) Computational Thinker (b)
Read, Edit, and Evaluate Programs	1A-AP-13, 1A-AP-14, 1B-AP-12, 1B-AP-15	Innovative Designer (c)
Self-Regulation	1A-AP-12, 1B-AP-13	Empowered Learner (a) Digital Citizen (b) Knowledge Constructor (a) Innovative Designer (b)
Cooperation with Others	1A-AP-15, 1A-IC-17, 1B-AP-16, 1B-IC-20	Creative Communicator Global Collaborator

APPLICATION OF PLR

To demonstrate our PLR's usefulness, we applied it to programming classes at a Japanese elementary school in 2017, 2018, and 2019. The targeted students were fourth-grade students aged 9–10 in 2017 and 2018; 43 students participated in 2017 and 39 participated in 2018. In 2019, the PLR was applied to programming instruction for 75 sixth graders (ages 11 and 12). This programming class was planned by an elementary school teacher. The class contents were focused on basic programming concepts, and the material focused on programming a robot. Because the robot used in the lessons differed between 2017, 2018, and 2019, the study was aimed to confirm the PLR evaluation's independence from the tools and methods.

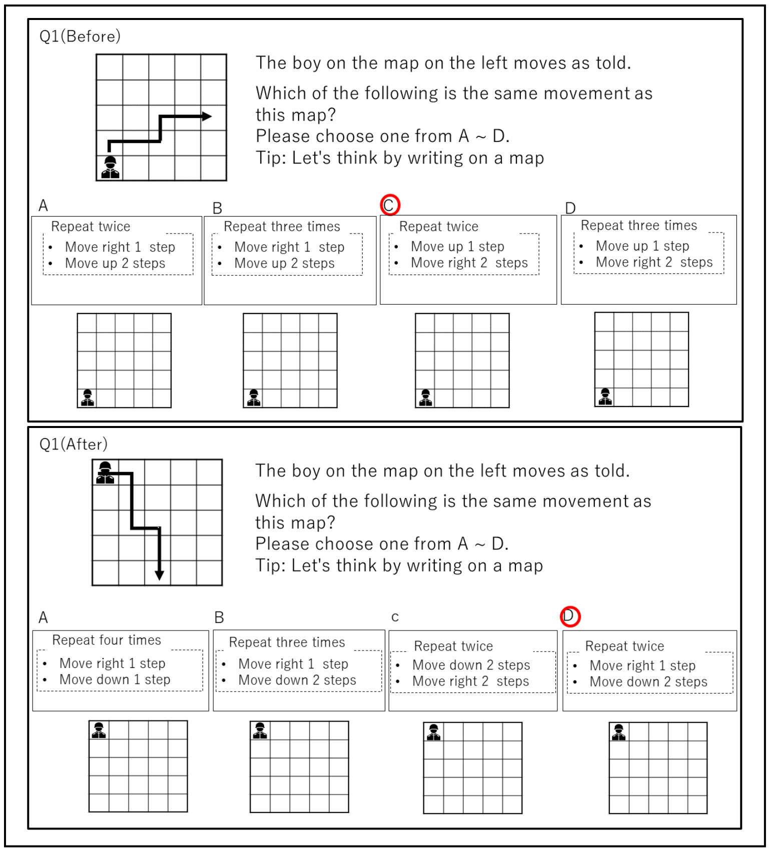
EVALUATION METHOD

Effectiveness of learning computer programming in the participating class was evaluated with quizzes. We evaluated learning stages by mapping programming quiz contents to the PLR. Each year comprised two lessons, each lasting 45 minutes. Quizzes were given before the first and after the second class. Students were allocated 30–45 minutes to complete the quizzes. Each quiz was explained to the teachers and was conducted under their supervision. Table 4 outlines the types of quiz questions, and Figure 2 presents examples of before and after class quiz questions. The before and after quizzes had the same format and tested the same programming topics; however, the questions differed slightly. Figures B1, B2, and B3 in Appendix B provide examples of the quizzes. Table 5 shows the correspondences between the quiz contents and PLR stages. A correct response in the quiz indicated that the student had reached the corresponding PLR stage in Table 5. The analysis for 2019 differed from that for 2017 and 2018, in that the grade levels of the students were different, and the quizzes given were different. The quizzes conducted in in 2017 and 2018 covered up to Q4 in Table 4. In 2019, the following two additional questions were asked:

- Q7: Selecting the type of input device
- Q8: Selecting a value that can be measured by a sensor

Table 4. Details of quizzes

No.	About	Quiz form
Q1	Simple repeat	Four-choice question
Q2	Simple conditional branching processing.	Handwriting question
Q3	Finding a rule from a given sequence	Numerical sequence filling question
Q4	Thinking of algorithms using the law in Q3	Handwriting question
Q5	Drawing a free line through all squares in one stroke	Handwriting question
Q6	"How did you draw that line?" in Q5	Handwriting question


Q1(Before)

The boy on the map on the left moves as told.

Which of the following is the same movement as this map?
Please choose one from A ~ D.
Tip: Let's think by writing on a map

A
Repeat twice
- Move right 1 step
- Move up 2 steps

B
Repeat three times
- Move right 1 step
- Move up 2 steps

[✓ C]
Repeat twice
- Move up 1 step
- Move right 2 steps

D
Repeat three times
- Move up 1 step
- Move right 2 steps

Q1(After)

The boy on the map on the left moves as told.

Which of the following is the same movement as this map?
Please choose one from A ~ D.
Tip: Let's think by writing on a map

A
Repeat four times
- Move right 1 step
- Move down 1 step

B
Repeat three times
- Move right 1 step
- Move down 2 steps

c
Repeat twice
- Move down 2 steps
- Move right 2 steps

[✓ D]
Repeat twice
- Move right 1 step
- Move down 2 steps

Figure 2. Programming quiz example

Table 5. Correspondence between quiz and rubric

	Programming concept			Designing programs					Creating programs
	Se-quence	Loop	Con-di-tional	Sub-divi-sion	Anal-ysis	Ex-trac-tion	Construction and Functionalization	Design Docu-ment	Use of Program-ming Concepts
Q1	2	2							
Q2			2						
Q3					3	3			
Q4							3		3
Q5								2	
Q6				4		4	4		

* The numbers in the table indicate the stage of PLR.

Table 6 shows the correspondence between this quiz and the rubric. If both Q7 and Q8 are answered correctly, the "Construction of Computer" learning achievement level will be evaluated as "3." In addition, Q6 is a free-form question. Hence, if the answer is correct, the document design stage is evaluated as "2" (Table 5). However, as this question has multiple correct answers, it can be evaluated in several ways, as different ways of thinking about programming. However, this survey did not include this in the evaluation.

Table 6. Correspondence between quiz and rubric (questions added for sixth grade)

	Construction of computer	
Q7	2	3
Q8	2	

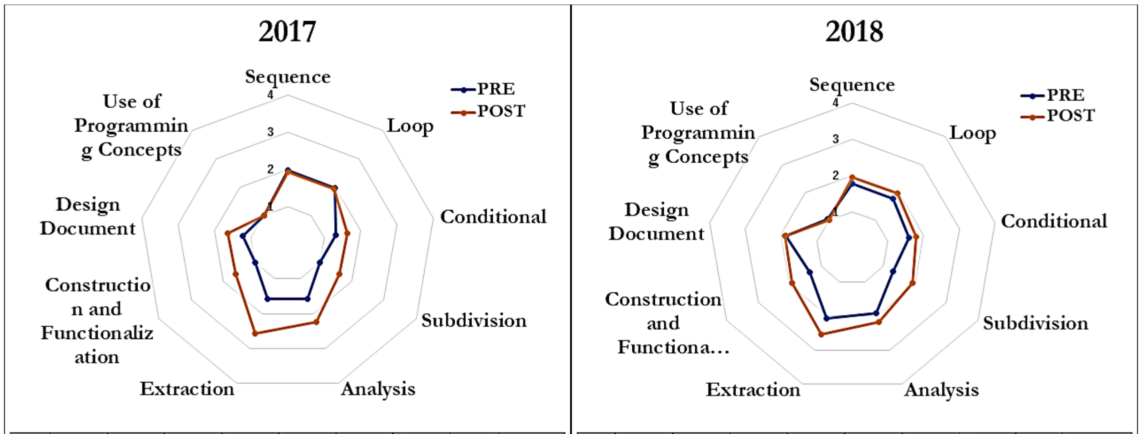
* The numbers in the table indicate the stage of PLR.

RESULTS

OVERALL APPLICATION RESULTS

We devised a rating scheme for the PLR based on quiz results from 2017, 2018, and 2019. Because we surveyed the fourth graders in 2017 and 2018 and the sixth graders in 2019, the grades surveyed were different and are reported separately. The total number of students assessed was 157. On such a small population size, the results of the rubric assessment cannot capture all effects of learning computer programming. Figure 3 shows the average PLR learning stages for all students in 2017 and 2018.

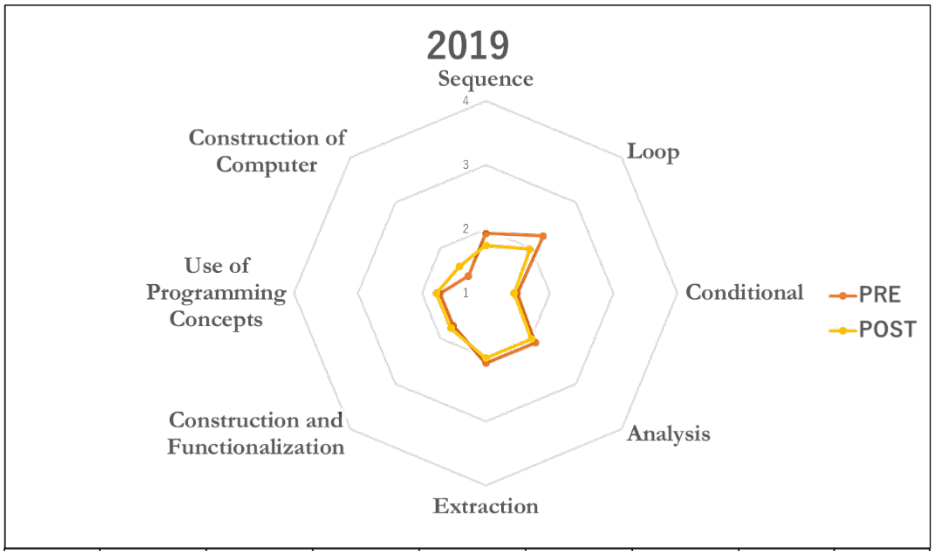
Student learning phases improved between the before and after class quizzes. The students understood the three elements of basic programming concepts (sequential, repeating, and conditional branching) as independent elements before the programming classes commenced, but their understanding approached PLR stage 2 after the class. Subdivision, construction, and functionalization in program design were all at PLR stage 1 before the classes began, indicating that most of the students did not understand these concepts. Students understood the items of analysis, extraction, and document design as single elements before the class. After completing the class, many students reached PLR stage 2 in the program design evaluation item. The PLR stages of students for the "Use of Programming Concept" item in "Creating Programs" remained unchanged after the class, possibly because the quiz evaluated this item at PLR stage 1 or 3. The students could not easily deal with multiple factors in this evaluation item; however, the other results suggested that students were approaching PLR stage 2 for the other items. Figure 4 shows the results for 2019. These results suggest that the learning attainment stage for the assessed items prior to the programming class is very close to stage 2. One reason for this could be that many of the students had previous programming experience. After completing this class, students showed an improvement in their understanding of computer principles, construction and functionalization of behavior, and creation of a program using each element. It seems that these students were able to combine, utilize, and develop their knowledge of programming. However, the fact that the attainment stage had not been raised for basic programming concepts suggests that these concepts serve as a sensory understanding.



	Sequence	Loop	Conditional	Subdivision	Analysis	Extraction	Construction and Functionalization	Design Document	Use of Programming Concepts
PRE	1.769	1.769	1.590	1.308	1.923	2.077	1.359	1.872	1.051
POST	1.949	1.949	1.795	1.923	2.179	2.538	1.923	1.897	1.000

	Sequence	Loop	Conditional	Subdivision	Analysis	Extraction	Construction and Functionalization	Design Document	Use of Programming Concepts
PRE	1.971	1.971	1.324	1.000	1.588	1.588	1.000	1.235	1.000
POST	1.941	1.941	1.647	1.618	2.235	2.559	1.618	1.647	1.000

Figure 3. Rubric evaluation results for 2017 and 2018



	Sequence	Loop	Conditional	Analysis	Extraction	Construction and Functionalization	Use of Programming Concepts	Construction of Computer
PRE	1.93	2.27	1.49	2.09	2.09	1.72	1.72	1.39
POST	1.75	1.97	1.44	2.01	2.01	1.77	1.77	1.59

Figure 4. Rubric evaluation results for 2019

When applied in this manner, the PLR indicates the learning effects of classes. Additionally, although different robot teaching materials were used in 2017 and 2018, it is possible to apply common evaluation items. Therefore, our PLR does not depend on learning tools.

Application to Each Student

Next, we evaluated learning effects at the individual level. Figure 5 shows the results of two students in the 2018 class as an example. The PLR clarifies the different learning effects on students in the same class. The PLR stages of both Student 1 and Student 2 changed after completing the class. Student 1 achieved a high overall PLR stage before the class, which was improved after the class. For example, after the class, Student 1's PLR stage for "Program Concepts" improved from 1 to 3. Student 2 achieved overall PLR stages 1 or 2 before the class. After the class, Student 2's PLR stage for "Extraction and Subdivision of Programs" in the "Designing Programs" unit increased to 3. These results confirm that the PLR distinguishes individual students' learning effects and thus can visualize learning effects in learning computer programming for individual students.

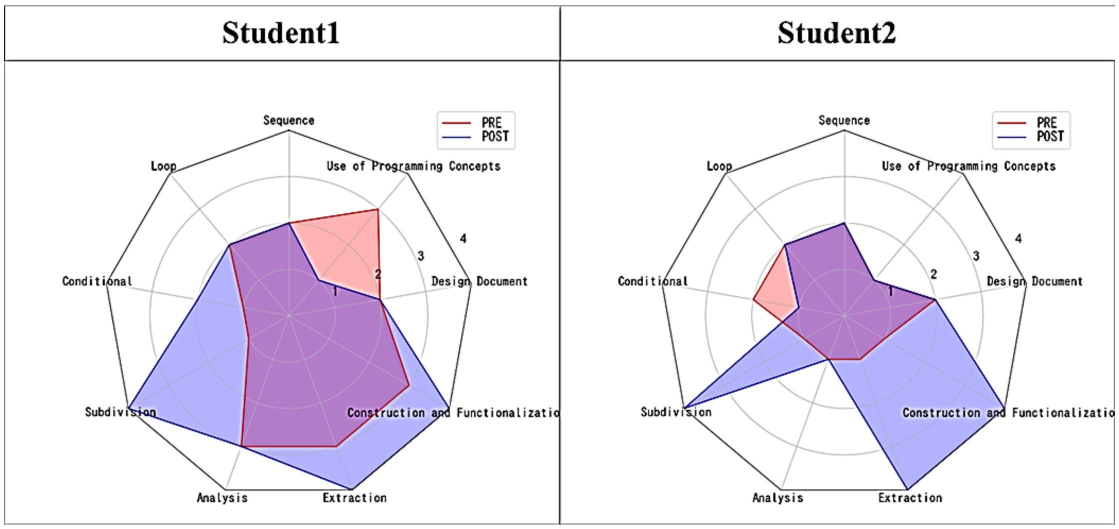


Figure 5. Results from two students in the 2018 class

Discussion

In this section the research questions are answered using the PLR and its application to classes in an elementary school.

Answer to RQs

RQ1: Can the evaluation criteria be proposed independently of tools and methods?

RQ1 asks whether a PLR for elementary school students can be independent of learning tools and methods. The categories and evaluation items in our programming-focused PLR are based on existing computer science and programming evaluation standards and rubrics. Moreover, the complexity of the evaluation can be reduced by refining the elements of the learning objectives in the existing standards. Past research (Grover et al., 2018) suggested that creating evaluation items by referring to existing standards, such as CSTA K-12 Computer Standards (CSTA 2017), can improve the reliability of evaluation. We also utilized educational classifications such as the SOLO taxonomy (Biggs & Collis, 2014) and Bloom's taxonomy (Krathwohl, 2002), which have been effective in previous learning computer programming studies. Furthermore, the stages of the evaluation items can be presented in a form that corresponds to the stages defined in these taxonomies. The PLR was thus divided into four learning stages and applied to programming classes at an elementary school in 2017, 2018, and

2019. These classes were designed by an elementary school teacher and were implemented using different robot-based teaching materials each year. The PLR was applicable to both classes without modification, demonstrating that a stepwise learning evaluation can be performed independently of tools and methods.

RQ2: Can the proposed rubric evaluate and visualize the learning effects?

To answer RQ2, PLR analyzed the learning effects in learning computer programming. We investigated whether the learning effects could be evaluated and visualized from the results, by evaluating and visualizing programming classes at an elementary school. For evaluation, we mapped quizzes examining students' understanding of programming to the proposed PLR. Using the PLR, we could visualize the overall changes in the learners' progress in each learning item, as shown in Figures 3 and 4. In some studies (Danaher et al., 2019; Grover et al., 2018), the overall results are plotted in graphical form. Other visualization systems for rubric-based assessment (Villamañe et al., 2016) confirm the usefulness of rubrics for visualizing learning effects. In addition, as shown in Figure 5, we visualized and evaluated the learning outcomes of various learning items of each student in a common lesson. Therefore, our system helps to characterize the learning effect of each student. The PLR has made it easier to visualize the development of skills of each student from the perspective of learning evaluation. Hence, our PLR can evaluate and visualize learning computer programming with different teaching materials and curricula.

RQ3: In what ways does the proposed rubric based on Bloom and SOLO taxonomies correctly evaluate students' learning in programming?

To address RQ3, we created a PLR based on SOLO taxonomy and Bloom's taxonomy. First, the use of SOLO and Bloom taxonomies was helpful in creating the rubrics. As shown in previous studies, rubrics based on SOLO and Bloom's taxonomy have been created with well-defined assessment stages and learning goals (Lister et al., 2006; Whalley et al., 2006). Therefore, the rubrics were useful for the learning stage and goal setting. Second, we investigated whether the learning stage and learning goals were set correctly. As seen from the answer of RQ2, the learning effect was evaluated and visualized, and the learning stage and learning goal settings were found to be appropriate. In addition, in a previous study, the reliability of the rubric rating was analyzed (Mustapha et al., 2016). The rubric used in this study is based on the pre-revision Bloom's taxonomy level principle. Thus, the rubric is shown to be reliable. Therefore, it suggested that the use of Bloom's taxonomy or SOLO taxonomy is useful for improving the reliability of the learning stage and goals. However, securing validity remains a challenge. Because our PLR is designed based on the revised Bloom and SOLO taxonomies, we need to clarify the relevance of these taxonomies. In summary, our PLR learns stages based on SOLO taxonomy. In addition, there are learning goals based on Bloom's taxonomy. Thus, the learning stage and learning goals can be made clearer, and the evaluation of student outcomes can be evaluated uniformly. Therefore, it is considered that our PLR, which is based on educational taxonomy, can evaluate student achievement learning computer programming more accurately.

Evaluation Difficulties

In this study, we utilized a rubric to evaluate programming education. One of the challenges faced in this study was the difficulty of evaluation. First, verifying whether the rubric can be used for the evaluation target is necessary. Second, the rubric itself must be evaluated as this is key to confirming the rubric's validity. Assessing the rubric's validity is important as it indicates whether the assessment of learning effectiveness is correct. We verified the rubric's usefulness in this study; however, its validity remains an issue. Therefore, we will evaluate the rubric in the future.

In addition, evaluating programming understanding was a difficult task during this research. The answers in many programming problems were not limited to one response. For example, in the free-writing question (Q5 in Table 4) (Appendix B, Figure B3), students provided different correct answers to "draw a line that passes through all squares in the map." Two representative examples are shown in Figure 6. Although both answers are correct, determining which is better depends on what is considered important. Hence, the evaluation should consider program correctness as well as suitability. For this purpose, the evaluators must list all possible answers and compare them from different viewpoints, and this requires thorough understanding of and experience in programming. This situation highlights one difficulty of evaluation.

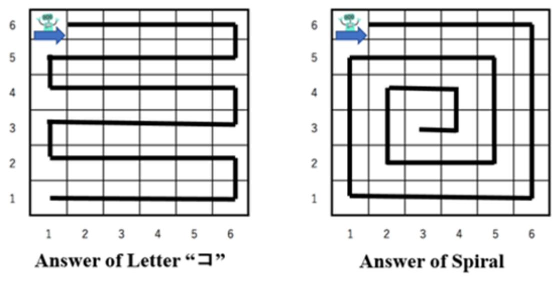


Figure 6. Comparison of the two answers

RELATED WORKS

Several indicators for learning computer programming evaluations are available. Our PLR references typical computer science standards, such as the CSTA computer science standards (CSTA, 2017), which have been systematically created to evaluate learning computer programming. However, the evaluation items in these standards are not stepwise but rather are evaluated as either 0 (satisfied) or 1 (not satisfied). Several PLRs, such as the one proposed by Catetė et al. (2016), are also available. However, these rubrics depend on methods and tools, a problem our PLR is intended to resolve. The evaluation items in our rubric are based on related research and the SOLO taxonomy and Bloom's taxonomy (Biggs & Collis, 2014; Krathwohl, 2002) and are divided into four learning stages.

CONCLUSIONS

We proposed a PLR for assessing learning computer programming in CSE for elementary school students. The PLR's assessment items refer to existing CSE indicators such as CSTA K-12 Computer Science Standards, and the learning stages and goals refer to educational goal classifications, such as those in the SOLO and Bloom's taxonomies. In addition, learning goals can be evaluated at four stages. Moreover, the PLR is designed to be independent of any particular learning computer programming tool or method.

To ascertain the usefulness of our rubric, we applied it to computer programming classes at an elementary school in 2017, 2018, and 2019, which were designed by elementary school teachers and used a variety of robot-based materials. The evaluation was done using a programming quiz. These quizzes were mapped and evaluated at the learning stage of the PLR. Consequently, the PLR was able to carry out assessments in these classes without any modification. Therefore, the evaluation of the PLR does not depend on the educational tools and methods, and the step-by-step learning evaluation can be performed independently. Moreover, the PLR has succeeded in assessing and visualizing student learning effectiveness at the overall and individual levels. Therefore, the proposed PLR renders visualizing the development of each student's skills easy and helps to characterize the learning effect. In addition, PLR learning stages and goals are based on SOLO and Bloom's taxonomies, thereby improving the reliability of the PLR assessment. Therefore, students' grades can be accurately evaluated for learning computer programming. Consequently, the proposed PLR eliminates the reliance on specific computer programming tools and methods. Furthermore, the PLR provides a step-by-step evaluation of learners' computer programming skills, thereby helping to facilitate the visualization and characterization of learning effects.

Limitations and Future Work

This study has certain limitations. First, the PLR learning goals are based on SOLO and Bloom's taxonomies, and the relationship between the two classification tables must be clarified. For example, the "Create" in Bloom's taxonomy is unclear in relation to the Relational and Extended abstraction stages of the SOLO taxonomy. Second, assessing all aspects of learning computer programming in CSE in PLRs remains a challenge. There can be more than one answer in programming so that different students provided different answers and aspects that can be assessed.

In addition, several PLR evaluation items were assessed based on the answers to questions that tested students' programming understanding. The quiz contents were mapped to the PLR in the evaluation; however, this correspondence is problematic for two reasons. First, the quiz and rubric stage mapping has not been validated by many assessors. To improve their relevance, more assessors could be used to discuss the correspondence between the PLR and the quizzes, curriculum, teaching materials, and related factors. Second, examining an evaluation item at all evaluation stages may not be possible. For example, the elements of "Use of Programming Concepts" cannot be evaluated at stage 4 of PLR. In future work, the PLR's mapping method must be improved to cover all evaluation stages.

Furthermore, the present study was performed on a small population size. We will continue the evaluation on larger cohorts of students.

Moreover, we plan to revise the rubric to clarify and increase the number of evaluation items. We will also reconsider the learning objectives and conduct a long-term evaluation of our rubric and quizzes. Finally, we plan to analyze the PLR's effectiveness on different forms of learning computer programming (e.g., workshops and lectures), which will enhance the PLR's usefulness.

Acknowledgement

This work was supported by JSPS KAKENHI Grant Number JP19K14328.

References

Alves, N. D. C., von Wangenheim, C. G., Hauck, J. C. R., & Borgatto, A. F. (2020, February). A large-scale evaluation of a rubric for the automatic assessment of algorithms and programming concepts. *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 556-562. https://doi.org/10.1145/3328778.3366840

Basu, S. (2019, February). Using rubrics integrating design and coding to assess middle school students' open-ended block-based programming projects. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 1211-1217. https://doi.org/10.1145/3287324.3287412

Biggs, J. B., & Collis, K. F. (2014). *Evaluating the quality of learning: The SOLO taxonomy (Structure of the Observed Learning Outcome).* Academic Press.

Bloom, B. S., Krathwohl, D. R., & Masia, B. B. (1984). *Bloom taxonomy of educational objectives*. Allyn and Bacon: Pearson Education.

CSTA. (2017). *CSTA K-12 computer science standards* https://www.csteachers.org/page/standards

Cateté, V., Snider, E., & Barnes, T. (2016, July). Developing a rubric for a creative CS Principles lab. *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education,* 290-295. https://doi.org/10.1145/2899415.2899449

Danaher, M., Schoepp, K., & Kranov, A. A. (2019). Effective evaluation of the non-technical skills in the computing discipline. *Journal of Information Technology Education: Research*, *18*, 1-18. https://doi.org/10.28945/4181

Grover, S., Basu, S., & Schank, P. (2018, February). What we can learn about student learning from open-ended programming projects in middle school computer science. *Proceedings of the 49th ACM Technical Symposium on Computer Science Education,* 999-1004. https://doi.org/10.1145/3159450.3159522

ISTE (2016). *ISTE standards for students* (Permitted Educational Use) https://www.iste.org/standards/for-students

Izu, C., Weerasinghe, A., & Pope, C. (2016, August). A study of code design skills in novice programmers using the SOLO taxonomy. *Proceedings of the 2016 ACM Conference on International Computing Education Research*, 251-259. https://doi.org/10.1145/2960310.2960324

Lister, R., & Leaney, J. (2003, January). First year programming: Let all the flowers bloom. *Proceedings of the Fifth Australasian Conference on Computing Education-Volume 20*, 221-230.

Lister, R., Simon, B., Thompson, E., Whalley, J. L., & Prasad, C. (2006). Not seeing the forest for the trees: Novice programmers and the SOLO taxonomy. *ACM SIGCSE Bulletin, 38*(3), 118-122. https://doi.org/10.1145/1140124.1140157

Krathwohl, D. R. (2002). A revision of Bloom's taxonomy: An overview. *Theory into Practice, 41*(4), 212-218. https://doi.org/10.1207/s15430421tip4104_2

Mustapha, A., Samsudin, N. A., Arbaiy, N., Mohammed, R., & Hamid, I. R. (2016). Generic assessment rubrics for computer programming courses. *Turkish Online Journal of Educational Technology-TOJET*, 15(1), 53-68.

Office of Superintendent of Public Instruction. (2018). *Computer science K-12 learning standards*. http://www.k12.wa.us/ComputerScience/LearningStandards.aspx

Saito, D., Washizaki, H., Fukazawa, Y., Tamura, M., & Sakuragi, Y. (2019, February). Rubric to evaluate programming learning of elementary school students. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 1280-1280. https://doi.org/10.1145/3287324.3293807

Selby, C. C. (2015, November). Relationships: Computational thinking, pedagogy of programming, and Bloom's Taxonomy. *Proceedings of the Workshop in Primary and Secondary Computing Education*, 80-87. https://doi.org/10.1145/2818314.2818315

Stegeman, M., Barendsen, E., & Smetsers, S. (2016, November). Designing a rubric for feedback on code quality in programming courses. *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*, 160-164. https://doi.org/10.1145/2999541.2999555

Thompson, E., Luxton-Reilly, A., Whalley, J. L., Hu, M., & Robbins, P. (2008, January). Bloom's taxonomy for CS assessment. *Proceedings of the Tenth Conference on Australasian Computing Education, 78*, 155-161. http://hdl.handle.net/2292/16257

Villamañe, M., Larrañaga, M., Álvarez, A., & Ferrero, B. (2016, November). RubricVis: Enriching rubric-based formative assessment with visual learning analytics. *Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality*, 363-368. https://doi.org/10.1145/3012430.3012541

Whalley, J. L., Lister, R., Thompson, E., Clear, T., Robbins, P., Ajith Kumar, P. K., & Prasad, C. (2006, December). An Australasian study of reading and comprehension skills in novice programmers, using the bloom and SOLO taxonomies. *Conferences in Research and Practice in Information Technology Series.*

APPENDIX A. PROPOSED RUBRIC

Table A1. Proposed rubric 1 (Attitude)

Cate-	Item	Stage 4	Stage 3	Stage 2	Stage 1
Attitude	Positivity	Learners voluntarily learn and challenge themselves without being instructed.	Learners spontaneously learn without being instructed.	Learners tackle learning if instructed but do not learn voluntarily.	Learners do not tackle learning even when instructed.
	Interest	Learners take a great interest in learning computer programming through self-evaluation. In addition, learners enjoy the programming activities.	Learners take a great interest in learning computer programming through self-evaluation.	Learners are interested in learning computer programming through self-evaluation.	Learners are not interested in learning computer programming through self-evaluation.
	Toughness	Learners willingly meet difficulties during the learning process and voluntarily investigate the clues and attempt self-solutions using different methods.	Learners do not concede when encountering a difficulty during learning and will attempt to find solutions.	Learners tend to work on learning but concede when faced with difficulties.	Learners conceded before the learning has properly begun.

Table A2. Proposed rubric 2 (Programming Concept and Construction of Computers)

Category	Item	Stage 4	Stage 3	Stage 2	Stage 1
Programming Concept	Sequence	Learners understand sequential execution, can read the program in order from beginning to end, can write a program of sequential execution, and can construct a program to be sequentially executed without requiring assistance.	Learners understand sequential execution, can read the program in order from beginning to end, and can write a program of sequential execution.	Learners understand sequential execution and can read the program in order from beginning to end.	Learners do not understand sequential execution.
	Loops	Learners understand loops, can read looped programs, and find loops by themselves. They can also incorporate loops into programs.	Learners understand loops, can read looped programs, and can write loop programs.	Learners understand loops and can read looped programs.	Learners do not understand how loops are used in programs.
	Conditional	Learners understand conditional branches, can read programs containing conditional branches, and can incorporate conditional branches into programs.	Learners understand conditional branches, can read programs containing conditional branches, and can write programs containing conditional branches.	Learners understand conditional branches and can read programs containing conditional branches.	Learners do not understand conditional branching.
Construction of Computer	Construction of Computer	Learners understand the principles of computers and external devices, can connect them, and can solve any problems that occur.	Learners understand the basic principles of computers and the connections and roles of external devices (mouse, printer, and network equipment).	Learners understand the basic principles of computers (input–output processor, sensor, and storage).	Learners do not understand the basic principles of computers and devices.

Table A3. Proposed rubric 3 (Designing Programs)

Category	Item	Stage 4	Stage 3	Stage 2	Stage 1
Designing Programs	Subdivision	Learners can correctly divide large problems into small problems that cannot be subdivided further.	Learners can correctly divide large problems into multiple small problems.	Learners can divide large problems into two or more smaller problems.	Learners cannot divide a large problem.
	Analysis	Learners can consider the cause and effect relationship of events, derive abstract rules and principles from the specific relationship, and write an upright thread.	Learners recognize the relationship between cause and effect of an event, and can export to a stand thread.	Learners can notice a relationship between the cause and effect of events.	Learners do not recognize the cause and effect relationship of an event.
	Extraction	Learners can select a method that suits the purpose with a minimum number of operations.	Learners can select a method that suits the purpose and take necessary action without guidance.	Learners can select a method that suits the purpose and decide the necessary action from the choices.	Learners cannot decide a method that suits the purpose.
	Construction and Functionalization	Learners realize the objective, consider the optimal combination of a plurality of procedures, and can create procedures with universality and reproducibility.	Learners achieve the objective by a plurality of procedures, including sequential processing, iteration, and conditional branch processes.	Learners notice that a procedure is composed of a plurality of steps, and can rearrange those steps as required.	Learners cannot build up combined operations in a procedure.
	Generalization	Learners seek similarities and relationships between the problem and a plurality of past resolved problems and finds a set of generalizable rules and principles that can be used to resolve new problems.	Learners observe that a problem can be solved using the approaches used to resolve past problems.	Learners recognize events with similarity and relationships to other events.	Learner cannot recognize an association between the current and previous events.

Table A4. Proposed rubric 4 (Designing Programs Continued)

Category	**Item**	**Stage 4**	**Stage 3**	**Stage 2**	**Stage 1**
Designing Programs	**Design Document**	Learners can plan and create a design document with reference to FIG. Learners can convey sentences, ideas, and procedures (story maps, etc.) in an easy-to-understand way.	Learners use ideas and procedures to plan and create a design document from figures and texts (story maps, etc.).	Learners can express ideas and procedures in picture form.	Learners cannot express ideas and procedures.
	Expression	Learners can create new original expressions by fully utilizing various expression methods.	Learners can imitate expressions of existing works and incorporate them into their current work.	Learners can create their own works using basic expression techniques.	Learners cannot make their own works.
	Creativity	Adopting a global perspective and design, learners achieve the purpose by associating the nature of the program, the position of the user, and other cues.	Learners achieve their design goals based on their own understanding by considering the user's position.	Learners achieve their design goals based on their own understanding.	Learners cannot achieve their design goals.

Table A5. Proposed rubric 5 (Creating Programs)

Category	Item	Stage 4	Stage 3	Stage 2	Stage 1
Creating Programs	Use of Programming Concepts	Learners can create a program, solution, or creative expression problems (including sequential execution, events, loops, conditional branching, parallelism, and variables).	Learners can create a program, solution, or creative expression problems (including sequential execution, events, and loops).	Learners can create a program, solution, or creative expression problems (including sequential executions and simple loop).	Learners cannot create a program, solution, or creative expressions.
	Logical Thinking	Learners can use Boolean set logic (e.g., condition setting at the time of a branch).	Learners can understand and use logical structures, such as repetition and conditional branching.	Learner can visually understand the feasibility of operations (e.g., whether operations can be combined in the correct order).	Learner cannot apply programming logic tools.
	Use of Software	Learners can use programming software to create a program that operates as intended.	Learners can use programming software to create some programs.	Learners can use programming software to a limited extent.	Learners cannot use programming software.
	Programming Language	Learners can program in both visual- and text-based languages.	Learners can program in text-based language.	Learners can program only in a visual-based language.	Learners cannot program in either visual- or text-based languages.
	Expressing Data	Learners can accurately represent all data as numbers or other symbols (e.g., the raising and lowering of the Yes/No thumb, number representation of colors, and arrow representation of direction).	Learners can accurately represent some of data by numbers or other symbols.	Learners understand that data can be represented by numbers or other symbols.	Learners cannot understand how all data can be represented by numbers or symbols.
	Use of Formula	Learners can use a range of arithmetic operators and comparison operators.	Learners can use operators to change the value of a variable.	Learners understand the use of operators in programs.	Learners do not understand the use of operators in programs.

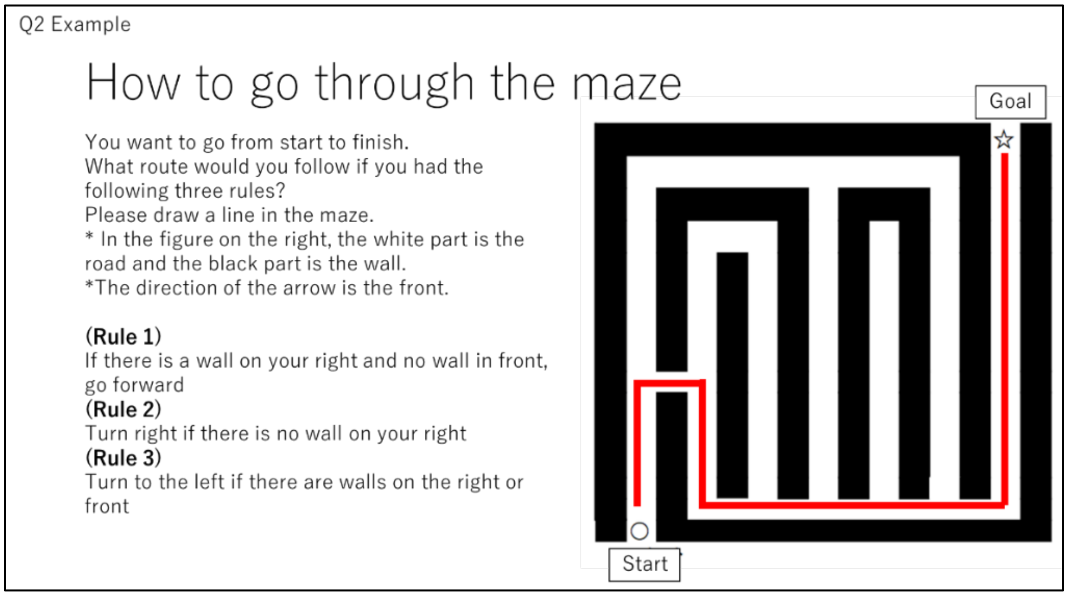
Table A6. Proposed rubric 6 (Read, Edit, Evaluate Programs and Self-regulation)

Read, Edit, Evaluate Programs	**Read**	Learners can read an existing program and explain its contents.	Learners can read an existing program.	Learner can read part of an existing program.	Learners cannot read an existing program.
	Edit	Learners can change an existing program to another program.	Learners can change part of an existing program.	Learners can modify an existing program.	Learners cannot modify a program.
	Evaluate	Learners can ensure that a program works as intended (i.e., can debug a program).	Learners can check that a program works as intended.	Aided by the teacher, learners can verify the operation of the program.	Learners cannot check the correct operation of a program.
Self-regulation	**Plan**	Learners proactively plan the achievement of the purpose or the conditions of running the program.	Learners can proactively plan the achievement of the objective.	Aided by the teacher, learners can plan the achievement of the objective.	Learners cannot plan the achievement of the objective.
	Safety Considerations	Learners make rules that guarantee the safety of their own working environment.	Learners make rules for safe operation in their working environment.	Learners understand the need for a safe working environment.	Learners do not understand safety concerns.

Table A7. Proposed rubric 7 (Cooperation with Others)

Cooperation with Others	**Announce own Ideas**	Learners present their ideas in a way that expresses their thought processes	Learners emphasize their thought processes	Learners can announce their thought processes	Learners cannot formulate thought processes
	Understand Others' Ideas	Learners are willing to hear the announcements of others, and can reference them to improve their own work	Learners are willing to hear and apply the announcements of others	Learners can hear and understand the announcements of others	Learners are unwilling or unable to understand the ideas of others
	Cooperation in Programming	Learners work with others in programming teams and contribute to the team effort without relying on others	Learners work with others in programming teams and contribute to the team effort without relying on others	Learners work in partnership with others as part of the team	Learners cannot partake in team programming exercises
	Contribution to Group Work	When working in a group, learners perform all three of the following activities: 1. Actively use their abilities to achieve their objectives. 2. Think about solutions to problems that were clarified in the middle of their activities. 3. Accept and analyze the opinions of others and compile the ideas of the group to meet the study objective.	When working in a group, learners perform two of the following three activities: 1. Actively use their abilities to achieve their objectives. 2. Think about solutions to problems that were clarified in the middle of their activities. 3. Accept and analyze the opinions of others and compile the ideas of the group to meet the study objective.	When working in a group, learners perform one of the following three activities: 1. Actively use their abilities to achieve their objectives. 2. Think about solutions to problems that were clarified in the middle of their activities. 3. Accept and analyze the opinions of others and compile the ideas of the group to meet the study objective.	Learners cannot contribute to teamwork.

APPENDIX B. QUIZ EXAMPLE

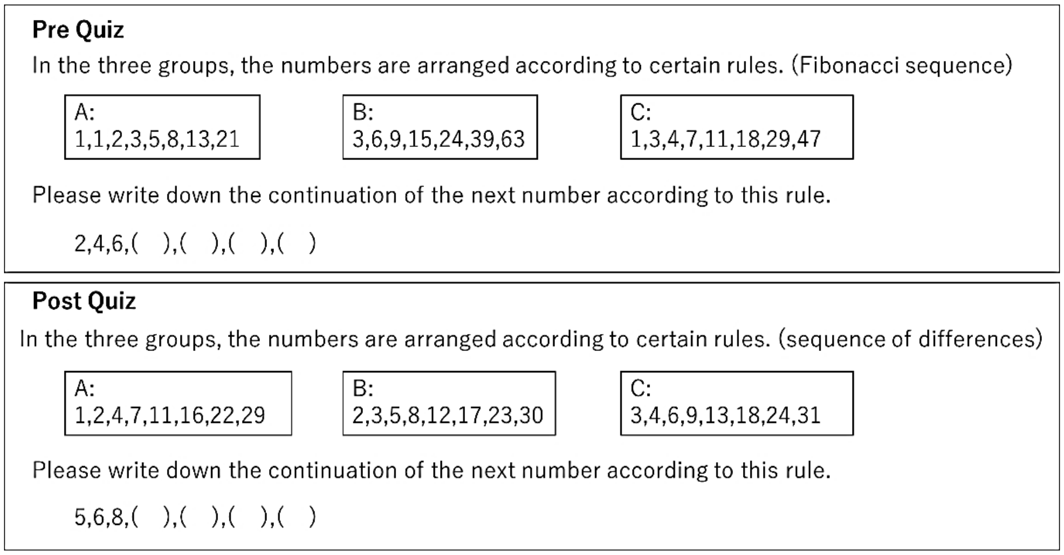

Q2 Example

How to go through the maze

You want to go from start to finish.
What route would you follow if you had the following three rules?
Please draw a line in the maze.
* In the figure on the right, the white part is the road and the black part is the wall.
*The direction of the arrow is the front.

(Rule 1)
If there is a wall on your right and no wall in front, go forward
(Rule 2)
Turn right if there is no wall on your right
(Rule 3)
Turn to the left if there are walls on the right or front

Figure B1. Quiz Example 1 (Q2 in Table 4)


Pre Quiz

In the three groups, the numbers are arranged according to certain rules. (Fibonacci sequence)

A:
1,1,2,3,5,8,13,21

B:
3,6,9,15,24,39,63

C:
1,3,4,7,11,18,29,47

Please write down the continuation of the next number according to this rule.

2,4,6,( ),( ),( ),( )

Post Quiz

In the three groups, the numbers are arranged according to certain rules. (sequence of differences)

A:
1,2,4,7,11,16,22,29

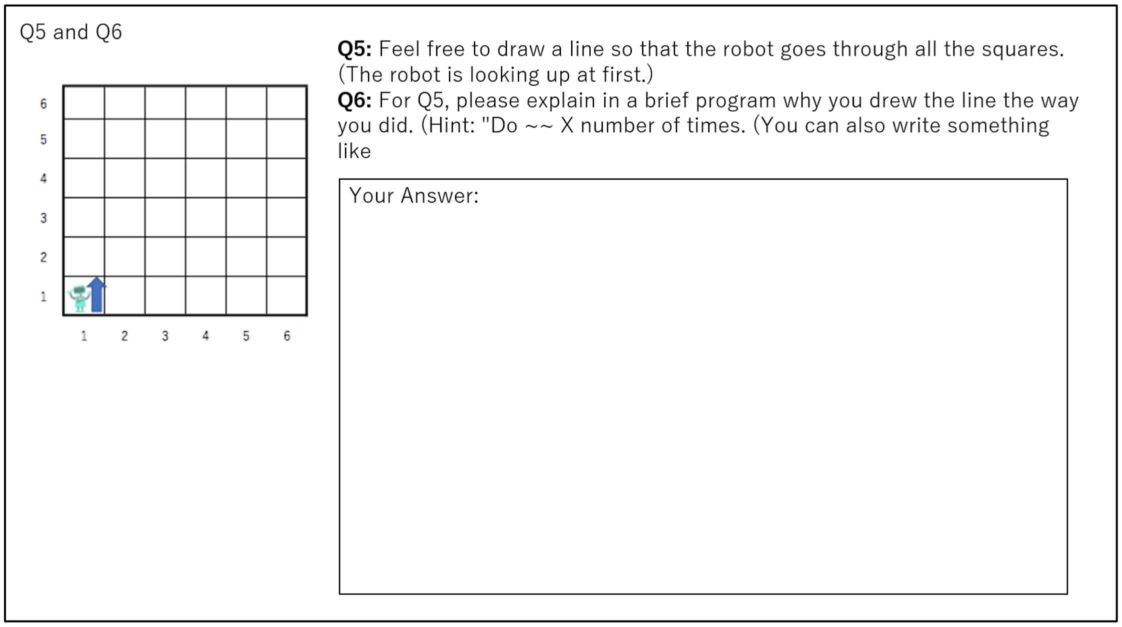
B:
2,3,5,8,12,17,23,30

C:
3,4,6,9,13,18,24,31

Please write down the continuation of the next number according to this rule.

5,6,8,( ),( ),( ),( )

Figure B2. Quiz Example 2 (Q3 in Table 4)


Q5 and Q6

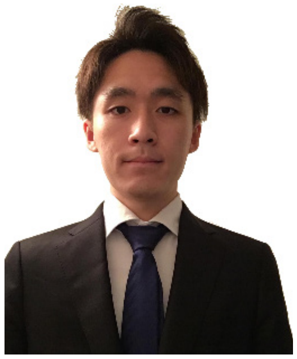
Q5: Feel free to draw a line so that the robot goes through all the squares. (The robot is looking up at first.)
Q6: For Q5, please explain in a brief program why you drew the line the way you did. (Hint: "Do ~~ X number of times. (You can also write something like

Your Answer:

Figure B3. Freewriting quiz question (Q5 in Table 4)

BIOGRAPHIES

Daisuke Saito is an assistant professor at the School of Fundamental Science and Engineering, Waseda University, Japan. He acquired his Doctor of Engineering degree from Waseda University, Japan. His research interests include programming education and digital game-based learning.



Shota Kaieda entered the Waseda University's English Program and received his Bachelor of Engineering degree in September 2020. During his freshman year, he started working as a programming teacher for elementary school students with Scratch and Minecraft. This experience peaked his interest and he decided to proceed with Programming Education as his research topic for his Bachelor Thesis. The research conducted for the thesis was a proposal and analysis on merging the mentor method with mob programming for programming education.

Hironori Washizaki is a Professor and Associate Dean at the Research Promotion Division at Waseda University, Tokyo and a Visiting Professor at the National Institute of Informatics. He also works in industry as Outside Directors of SYSTEM INFORMATION and eXmotion. Since 2017, he has been the lead on a large-scale grant at MEXT called enPiT-Pro SmartSE, which encompasses IoT, AI, software engineering, and business.

Yoshiaki Fukazawa received the B.E., M.E. and D.E. degrees in Electrical Engineering from Waseda University, Tokyo, Japan in 1976, 1978, and 1986, respectively. Currently, he is a professor in the Department of Information and Computer Science, Waseda University and the Director of Institute of Open Source Software, Waseda University. His research interests include software engineering, particularly reuse of object oriented software and agent-based software.