Multivocal Challenge Toward Measuring Computational Thinking

Bebras Challenge Versus Computer Programming

Yoshiaki Matsuzawa[1(✉)], Kazuyoshi Murata[1], and Seiichi Tani[2]

[1] School of Social Informatics, Aoyama Gakuin University, Tokyo, Japan
matsuzawa@si.aoyama.ac.jp
[2] Department of Information Science, Nihon University, Tokyo, Japan

Abstract. Towards the establishment of an evaluation platform for computational thinking (CT), in this paper, we use the "Bebras Challenge" coined by Dr. Dagienė as a measurement tool of CT skills. This paper presents a "triangle examination" which includes three kinds of testing methods (programming testing, traditional paper testing, and Bebras Challenges). Approximately one hundred and fifty non-computer science (CS) undergraduate students participated in the examination as a part of an introductory programming course. The result indicated a weak but positive correlation (.38–.45) between the three methods. Additional qualitative analysis for each task in Bebras showed that requirements of algorithm creation and interpretation, and explicitness of the description, are two critical factors to determine a high correlation between other testing methods. We conclude our research by showing a clear correlation between the Bebras Challenge and actual programming.

Keywords: Programming · Literacy · Computational Thinking · Bebras Challenge

1 Introduction

With the banner of "Computational Thinking (CT)" [1], a movement of promoting programming education as literacy for all citizens is increasingly growing all over the world. Following European and North American countries, the Japanese government issued a statement which includes compulsory programming education at elementary schools from 2020 [2]. The purpose of the education is not merely to increase computer science engineers in the future, but to increase 'good users' of computers. While advanced technologies have made a fundamental change in science practices over the past 50 years, a renovation of learning environments for both teachers and learners is required to empower computing [3]. Hence, the purpose of education is to develop citizens who can naturally employ new science practices and live in the knowledge society over the next 50 years.

However, the problem is how we can measure (or know) CT has been laid in front of teachers/researchers. Brennan and Resnick stated "there is little agreement about what CT encompasses, and even less agreement about strategies for assessing the

Published by Springer Nature Switzerland AG 2019
D. Passey et al. (Eds.): OCCE 2018, IFIP AICT 524, pp. 56–66, 2019.
https://doi.org/10.1007/978-3-030-23513-0_6

development of computational thinking" on the basis of their long-term experiences in teaching children with Scratch [4]. Weintrop et al. [3] asserted that "much of the difficulty stems from the fact that the practices collected under the umbrella term CT [1, 5]". Recently, several studies on CT have been published under the IFIP umbrella (e.g. [6, 7]); however, in these papers, the discussion around assessment is limited. We think that a further clarification of CT needs a deeper discussion on assessment.

A number of research studies to evaluate student performance in undergraduate programming classes have been conducted using paper examinations. For example, Lister et al. [8] reported on reading and tracing, and Ford [9] tried to assess the achievement of classes incorporating tests used in cognitive studies in programming. However, there have been very few research studies that discuss methods of evaluating CT as formative problem-solving skills, instead of skills to tackle programming language elements.

Our purpose in this paper is the establishment of an evaluation platform for CT based on the "Bebras Challenge", a measurement tool of CT skills. A "triangle examination" is conducted, which includes three types of testing methods (programming testing, traditional paper testing, and the Bebras Challenge), and the correlation between them is analysed.

2 Theoretical Framework

2.1 Computational Thinking

There is some consensus between researchers that the movement of computing education is a revival of 1980s programing education with Logo [10]. The origin of this movement originated from Papert, who coined the term computational thinking [5], but was not primarily intended to develop programming skills but to open a new method of learning mathematics through programming. By working in situated environments, children could construct their ideas by directly operating these ideas in a situated world [11]. Papert criticised technocentrism of programming education, and he expressed the purpose as "to give children a greater sense of empowerment, of being able to do more than they could do before" [12].

More than 20 years after the first generation of programming education, as discussed above, Wing started to use the term CT [1] independently. Wing's CT has similarities with the term introduced by Papert, since both of them focus on the necessity of developing those general skills which are needed by all citizens in the knowledge society. Wing's CT was initially discussed in the computer science (CS) community, whereas Papert's term was considered from Piaget's constructivism perspective and has been discussed in the cognitive and learning sciences. This difference brought about a difference in focus: Wing's definition sounds like CS, technology-centric concepts; whereas Papert's use of the term is aimed to foster a greater sense of empowerment in solving problems through computing.

Weintrop et al. [3] offer a recent attempt to define CT by comprehending the literature over 30 years including the two generations of CT discussion, as mentioned above. They emphasised the change in science practices while advancing technology,

and how to educate to develop a sensibility in order to literally survive in the world where changed practice is common sense. The higher level problem-solving skills are summarised as: “Data practices”; “Modeling and Simulation practices”; “Computational Problem Solving practices”; and “Systems Thinking practices”. The practice to formulate a problem into computational models is defined as “Designing Computational Models”, which is included in “Modeling and Simulation practices”. “Computer Programming” or “Creating Computational Abstractions” is merely a part of “Computational Problem Solving”. Accordingly, the large problem-solving cycle from the formulation of a problem to an evaluation of solutions was defined as “Computing” and then the competencies to conduct the process were defined as CT. Although the paper claimed it was originally designed for application of CT in science and mathematics, the definition is applicable to other disciplines.

2.2 The Bebras Challenge

The Bebras Challenge is an educational practice where students are challenged to do several small tasks, using CS/CT concepts to complete those tasks [13, 14]. It is also an international informatics contest where a large number of students participate from over 40 countries. Although the activity looks like quizzes that can be used in a classical classroom for grading students, we can see the difference in task design, being described as “wrap[ping] up serious scientific problems of informatics and the basic concepts into playful tasks, inventive questions thus attracting students’ attention” [13]. Accordingly, the thoughtfully-designed tasks are playful and appreciated by students.

Bebras includes a “contest” where students can compete with each other through their scores, but that is not the main goal. The goal is “to motivate pupils to be interested in informatics topics and to promote thinking which is algorithmic, logical, operational, and based on informatics fundamentals” [13]. In other words, it can be explained as promoting the enjoyment of thinking, the learning of activities embedded in the procedure, and consequences performed as a formative evaluation of the learning process.

The Bebras international contest was created in 2004 by Dr. Dagienė, with a first report published in 2006 [14]. From the practice and research conducted over a decade, a review paper was published in 2016 [13], which included approximately 50 papers published during the previous ten-year period.

2.3 The Bebras Challenge as Assessment Tools

Although the Bebras challenge is not designed to assess students’ knowledge or skills, Dr. Dagienė refers to the capability of Bebras as an assessment tool over a long time period [13]. One paper [13] describes how one of the most important and required cognitive operations in CSTA (Computer Science Teachers Association) K-12 Standards - using visual representations of problem states, structures, and data - investigates the “task-based assessment” approach to assess CT. Hubwieser et al. validated the use of item response theory, focusing on whether Bebras tasks could assess CT in CSTA [15, 16]. Dolgopolovas et al. [17] expanded on Hubwieser’s work and tried a validation

of Bebras as assessment tools of CT with first-year software engineering university students. This study included the following two research questions (RQs):

RQ1: How can the CT skills of novice software engineering students be evaluated independently of programming language?
RQ2: What is the relation between novice software engineering students' computational thinking skills and programming course results?

For RQ1, a study was attempted by Bebras with 65 university students. They succeeded in validating the test using item response theory, as described in a previous study [17]. However, for RQ2, they failed to find a correlation between Bebras and examination scores in the programming course. The paper [17] discussed how the failure resulted from the quality of questions in the examination, which asked for practical knowledge in programming (e.g. grammar and knowledge of a particular library) instead of algorithms, formalisation, or abstraction that are required in CT.

Djambong et al. [18] offers another influential study, which attempts a summative assessment of programming education where both grade 6 and grade 9 students engage in a robot programming activity using LEGO Mindstorms. Bebras was examined using a pre- and post-test after five hours (during a five-week period) in a classroom, but significant differences were not identified.

The study detailed in this paper can be seen as a revised version of this work. We conducted an experimental study using the same research question (RQ2), but we asked for actual programming construction in the examination in order to reveal any correlation between computer programming and Bebras activities with non-CS students.

3 Method

3.1 Research Question

Our research question is as follows:

RQ: Could we use Bebras as an assessment tool for computational thinking, and if so, to what extent? (What is the correlation between Bebras, the actual programming test, and the paper test?)

The aim of the RQ is an evaluation of Bebras as an assessment tool of CT by examining the correlation between Bebras, the actual programming test, and the paper test. Although the result of this question in the previous research [17] was negative, we hypothesised that a positive result could be achieved by improving the paper and programming tests by asking about CT instead of practical programming knowledge.

3.2 Experimental Study Environment

An experimental study was conducted in an introductory programming course designed for social informatics (non-CS) major students. The class was designed to develop language independent CT skills through the practical programming experiences with Visual Basic, and HTML (HyperText Markup Language) for all students in the major class (compulsory).

The time schedule of the class and the examination is shown in Fig. 1. The class was scheduled over 15 weeks (three hours per week), including an examination in the 16th week. Students engaged in programming with Visual Basic during the first nine weeks, and then engaged in web page authoring in HTML during the last four weeks.

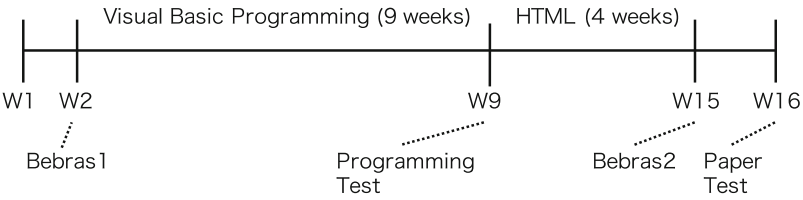

Fig. 1. Time schedule

The first trial of Bebras (Bebras1) was carried out in week 2 as a pre-test, and the second trial was carried out in week 15 as a post-test. The programming test was conducted in week 9 and the paper test was conducted in week 16. As there was a winter break in week 13, there was an approximate 10-week interval between the programming and paper tests. Despite this gap, we assumed the skill levels of the students were essentially the same, as the content of the class during the term was on web page authoring in HTML. As the Bebras trial was addressed as a part of the lectures, students were given as an incentive a maximum score of 10% of their grade.

3.3 Design of the Bebras Task

The Bebras task used in this study was from the senior level Japanese version of Bebras tasks. We considered that the difficulty of the senior tasks fitted well for first-year university students. Ten tasks were selected from Bebras 2015 and 2016 Challenges for the first trial (Bebras1). As a pilot trial had been carried out with a different student cohort one year earlier, the tasks which had a statistically high correlation were selected. Twelve tasks in all were selected from the Bebras 2017 Challenge for the second trial (Bebras2). All of the selected tasks are listed in the results section.

The time limit for the challenge was set to 40 min in both trials. The time was a little longer than that in the international challenge in order to encourage students to think deeply while doing each task. The web-based system used in the international challenge was used for the experiment. Students were given 10 min to practice the use of the system; consequently, there were no students who failed to answer the tasks.

3.4 Design of the Programming Test

The programming test examined during the 9th week of class was an examination to develop a program to meet the requirements presented to students using a specification sheet. All of the students were assigned a single computer, and then were asked to submit a workable code in Visual Basic. The students were allowed to access all course materials during the examination, although access to the Internet was prohibited.

Students were required to create four tasks, given on the specification sheets, within 85 min. The easiest task did not require using any loops, but to use branching. The most difficult task required using a collection (the course teaches Listbox for abstract data collection) and developing an algorithm to process inside the collection.

During the test, the students could use a development environment (Visual Studio). Therefore, the activity measured comprehensive skills of program construction: designing data structure and algorithms; correcting compilation errors; or debugging. Additionally, the time constraint required a certain level of fluency in their programming processes; we observed that some students gained full marks close to the time limit.

3.5 Design of the Paper Test

The paper test administered during the 16th week of the class was the traditional examination carried out using paper and pencil. Students could access all course materials during the examination as well as the programming test, while the use of all electronic devices was prohibited. The test was in six sections, lasting 85 min.

There were three kinds of tasks (code tracing, code complementing, and code ordering) among the sections analysed. Code tracing is a type of task where students read the code written in Visual Basic and then answer the outputs, values of variables at a particular state, or figures expected to be drawn by the code. Code complementing is a type of task where students complete a part of the code to complement the blanked, uncompleted code. Code ordering is a type of task where students initially order code fragments in random order to complete a workable code. In this manner, the tasks of the paper test were designed to assess algorithmic thinking, which is located within basic grammar knowledge.

4 Results

4.1 Descriptive Statistics

The study was conducted in our introductory programming class during the 2017 academic year. Approximately 200 students in the school are required to register for this course, with the students being randomly assigned to one of four separate classes in order to reduce the number of students in each class. Each of the four classes was managed by different teaching staff, but the course materials and examinations were the same.

Three of the four classes were randomly selected for the analysis. The descriptive statistics of the four examinations are shown in Table 1. In Table 1, “n” refers to the number of students, and the other number indicates the average score of the four tests: Programming; Paper; Bebras1; and Bebras2, respectively. A total of 137 students’ data were used in the analysis. No significant differences in scores were found between the classes according to these descriptive statistics.

Table 1. Descriptive statistics

	n	Programming	Paper	Bebras1	Bebras2
Class A	43	49.2	50.5	64.9	55.4
Class B	45	53.3	50.4	71.6	57.4
Class C	49	44.7	44.0	62.3	55.3
	137	48.9	48.2	66.2	56.0

4.2 Correlation Analysis Between the Four Examinations

Figure 2 is a correlation table, showing the correlation and scatter plot between the four examinations. The correlation between Programming and Bebras was approximately 0.4. The correlation between Paper and Bebras was approximately 0.45. Although a strong correlation was not indicated, this is clearly a different result from the previous research [17], as no relationship was reported between the examinations in that study.

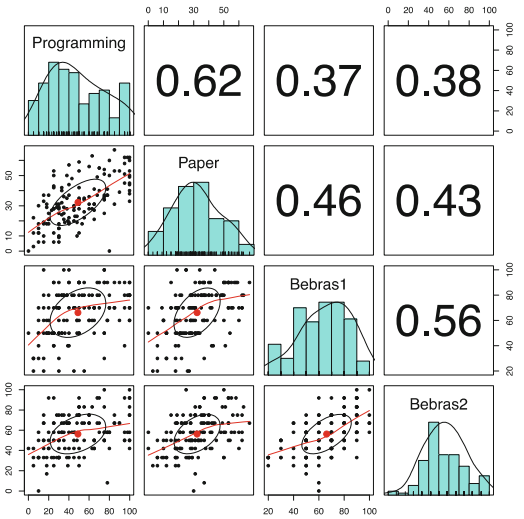

Fig. 2. Correlation table of the four examinations

Additionally, the correlation value can be considered as reaching a significant level when considering it in relation to the other correlations. The paired correlations between Programming and Paper, and Bebras1 and 2, respectively, are approximately .6. We assume the two pairs of tests were quite close in terms of methods, respectively. Therefore, these correlation values were high; however, they could be considered as not being higher than what we had expected. Notably, the correlation between Bebras1 and Bebras2 could be expected to be much stronger (.8–.9) if the validity of the tasks was improved. Accordingly, the correlation value between Bebras and other tests (.4–.45) can be considered sufficient to show a significant relationship between the actual programming and Bebras.

4.3 Detailed Analysis for the Tasks

A detailed analysis of the correlation between the programming test and each task in Bebras was performed. Table 2 shows the percentage of correct answers, and the significant differences between the paper test, programming test, and each task, respectively. The significant differences are indicated by the average scores between the group that answered correctly and the group that failed in answering for each task. The average score of the group that answered correctly was higher than the group that failed their answers in all the tasks.

Table 2. Percentage correct answers and average score significant differences by task

Q ID	Task ID	Name	Correct	Program	Paper
AQ1	2015-LT-04	Pencils alignment	0.78	**	**
AQ2	2015-JP-05	Ice Cream Shop	0.74	ns($p < .1$)	*
AQ3	2015-CA-02	Fireworks	0.64	ns	ns
AQ4	2015-DE-05	Mobiles	0.39	ns	*
AQ5	2016-CZ-06	Finding the final state	0.76	ns($p < .1$)	**
AQ6	2016-MY-02	Scanner code	0.85	*	**
AQ7	2016-NL-04	KIX Code	0.58	**	**
AQ8	2016-AT-06	Recursive painting	0.49	ns	*
AQ9	2016-JP-02	Paint it black	0.70	ns	ns($p < .1$)
AQ10	2016-IT-02b	Red and blue marbles	0.69	ns($p < .1$)	ns($p < .1$)
BQ1	2017-RU-04	Grandmother's jam	0.42	ns	ns($p < .1$)
BQ2	2017-JP-04	Colorful Building	0.64	ns	ns($p < .1$)
BQ3	2017-AT-03	Files	0.67	ns	ns
BQ4	2017-MY-05	Moving Die	0.60	**	ns
BQ5	2017-IR-06	Bebragram	0.80	**	**
BQ6	2017-CH-01a	Exit the maze	0.73	ns($p < .1$)	**
BQ7	2017-SK-12b	Robot	0.54	**	**
BQ8	2017-KR-07	Icon Image Compression	0.48	ns($p < .1$)	**
BQ9	2017-KR-02	A Stray Baby Beaver	0.69	*	*
BQ10	2017-DE-09	BikeFun	0.72	ns	**
BQ11	2017-RU-02	Digit recognition	0.09	ns	ns
BQ12	2017-IT-10	Library	0.37	ns($p < .1$)	ns

*$p < .05$, **$p < .01$

The result of a qualitative analysis for each task was conducted by the first author, although he was not a specialist in Bebras tasks, or in the interpretation of the results from the viewpoint of the correlation between Bebras, Programming, and CT.

Overall, the results of the programming and paper test indicate a similar tendency in the correlations. For example, significant differences were consistently found in AQ1, AQ6, AQ7 and BQ5, BQ7, BQ9, whereas no significant differences were consistently observed in AQ3, AQ9, AQ10, and BQ1, BQ2, BQ3, BQ11, or BQ12.

The tasks showing significant difference involved algorithm comprehension, creation, and abstraction. A typical example is AQ1 "Pencils alignment" where the task asks about the result of sorting algorithms written in natural language. There was a direct relationship to the topic of sorting algorithms that was taught in the lecture. Tasks which included an algorithm in a geometric field such as AQ5 "Finding the final state" or BQ6 "Exit the maze", were expected to show strong correlation. For the two tasks, only the paper test indicated significant differences, so further consideration is needed for this in terms of interpretation.

The tasks indicating no significant differences included relatively little algorithm comprehension, but also included other CT concepts, for example, data structure expressions, such as AQ4 "Mobiles." Although the lecture included basic data structure expressions with linear collection, further data structure expression, used to model the actual world, seemed to be difficult in introductory programming for non-CS students. For other examples, AQ2 "Ice Cream Shop" requires the concept of database structure, and BQ1 "Grandmother's jam" requires the concept of task scheduling.

Another possible deterministic factor to be considered was whether an expression of a task was explicit or not. For example, in BQ7 "Robot", the rule to be applied is simple and also clearly given by illustrations, whereas in BQ3 "Files" students have to construct a procedural algorithm by declarative rules. Another example, BQ12 "Library" was seen to create difficulty in description, although this may have been caused by local translation.

5 Discussion

The RQ of the study was "Could we use Bebras as an assessment tool for computational thinking? And if so, to what degree?" As shown in Sect. 4.2, the results indicate a positive correlation (.38–.45) between the three methods. As we discussed, this was an opposite result from previous research [17]. One notable difference of the two separate research studies is the design of the test; the test in this research required algorithm creation with actual programming, whereas the previous research required declarative information for a programming language. The results of the qualitative analysis for each task in Bebras support this consideration. The analysis revealed two critical factors: a requirement of algorithm creation and interpretation; and the explicitness of the description. Hence, we conclude here that the research showed clearly the difference between the Bebras Challenge and actual programming.

As was expressed earlier, a significant assumption in this research was that the actual programming requires a certain level of assessment ability in CT in higher-level problem-solving skills [3]. One significant criticism is that Bebras can be more accurate in measuring language-independent CT than actual programming. However, even if Bebras tasks enable users to operate CT concepts in a language-independent way, the concepts should be finally applied in practical situations: computer programming. Bebras Challenge trials lead to basic understanding of CT concepts, while subsequent programming experiences make the understandings deeper, and consequently they appear as a Bebras score. This cycle is expected in the design of Bebras. Accordingly, we believe our results will encourage all programming researchers/practitioners who

are engaged in supporting language-independent and creative programming practices instead of conveying only syntax knowledge.

Acknowledgments. This work was supported by JSPS KAKENHI Grant Numbers 16K00488, and 17H06107.

References

1. Wing, J.M.: Computational thinking. Commun. ACM **49**(3), 33–35 (2006)
2. Prime Minister of Japan and His Cabinet: 26th Council for Industrial Competitiveness (2016). http://www.kantei.go.jp/jp/singi/keizaisaisei/skkkaigi/kaisai.html
3. Weintrop, D., et al.: Defining computational thinking for mathematics and science classrooms. J. Sci. Educ. Technol. **25**(1), 127–147 (2016)
4. Brennen, K., Resnick, M.: New Frameworks for Studying and Assessing the Development of Computational Thinking, Presented at the Annual Meeting of the American Educational Research Association (2012)
5. Papert, S.: Mindstorms: Children, Computers, and Powerful Ideas. Basic Books, New York (1980)
6. Angeli, C., et al.: A K-6 computational thinking curriculum framework: implications for teacher knowledge. J. Educ. Technol. Soc. **19**(3), 47–57 (2016)
7. Fluck, A., et al.: Arguing for computer science in the school curriculum. J. Educ. Technol. Soc. **19**(3), 38–46 (2016)
8. Lister, R., Adams, E., et al.: A multi-national study of reading and tracing skills in novice programmers. SIGCSE **2004**, 119–150 (2004)
9. Ford, M., Venema, S.: Assessing the success of an introductory programming course. J. Inf. Technol. Educ. **9**, 133–145 (2010)
10. Tedre, M., Denning, P.: The long quest for computational thinking. In: Proceedings of the 16th Koli Calling Conference on Computing Education Research, pp. 120–129 (2016)
11. Papert, S.: Perestroika and epistemological politics. In: Harel, I., Papert, S. (eds.) Construtionism, pp. 13–28. Ablex, Norwood (1991)
12. Papert, S.: A Critique of technocentrism in thinking about the school of the future. talk presented at children in an information age: opportunities for creativity, innovation, and new activities, Sofia, Bulgaria (1987)
13. Dagiene, V., Stupuriene, G.: Bebras a sustainable community building model for the concept based learning of informatics and computational thinking. Inf. Educ. **15**(1), 25–44 (2016)
14. Dagiene, V.: Information technology contests – introduction to computer science in an attractive way. Inf. Educ. **5**(1), 37–46 (2006)
15. Hubwieser, P., Mühling, A.: Playing PISA with Bebras. In: Proceedings of the 9th Workshop in Primary and Secondary Computing Education, WiPSCE 2014, pp. 128–129 (2014)
16. Hubwieser, P., Mühling, A.: Investigating the psychometric structure of bebras contest: towards measuring computational thinking skills. In: International Conference on Learning and Teaching in Computing and Engineering, pp. 62–69 (2015)
17. Dolgopolovas, V., Jevsikikova, T., Dagiene, V., Savulioniene, L.: Exploration of computational thinking of software engineering novice students based on solving computer science tasks. Int. J. Eng. Educ. **32**(3A), 1107–1116 (2016)

18. Djambong, T., Freiman, V.: Task-based assessment of students' computational thinking skills developed through visual programming or tangible coding environments. In: Sampson, D.G., Spector, J.M., Ifenthaler, D., Isaias, P. (eds.) Proceedings of the International Conference on Cognition and Exploratory Learning in the Digital Age (CELDA), pp. 41–51 (2016)