Advances in Social Science, Education and Humanities Research (ASSEHR), volume 201
International Conference on Indonesian Technical Vocational Education and Association (APTEKINDO 2018)

Assessing Computational Thinking using Pseudocode Programming Instrument

Ekohariadi, Yeni Anistyasari, Ricky Eka Putra, Ibnu Febry Kurniawan
Informatics Department
Universitas Negeri Surabaya
Surabaya, Indonesia
ekohariadi@unesa.ac.id

***Abstract*— the purpose of this research is to create an instrument to measure understanding of computational thinking concept and independent programming language that would not be biased by any computer's programming language. Computational thinking can be taught through programming of visual language Scratch as a new way to solve problems. Academic performance can be assessed using pseudocode programming. The basic idea of computational thinking is algorithmic thinking and exploring the concept of mathematics to solve the problems with more convincing results. The concepts of computational thinking which are taught through Scratch visual programming language are conditional, iteration, array and procedure. δ is a parameter location which determines the position of item characteristic curve in the relation with ability scale. 40 items of computer programming tests were evaluated using Rasch model in which one item was deleted. δ parameter was in range of -2.52 to 2.83. The easiest concept to learn is *procedure* while the most elusive one is array. The average of gain scores was 0.20, of which meant computer programming learning was not effective. The correlation between learning gain and pretest scores was -0.174.**

Keywords*— computational thinking, pseudocode, Rasch model, visual programming

I. INTRODUCTION

Programming is a valuable skill. Introductory Programing course is taught using programming language such as C, C++, Java, Scratch, and Python. In addition, programming is important for Informatics students to improve problem solving skills and to interact with computer system. It is important for them to understand not only programming principals, methods, and technique but also computational thinking skills. The need of computational thinking and programming skills enhancement has gained widespread attention for students to be ready for life in the 21st century [1]. Introductory programming therefore is part of curricula in Faculty of Engineering in Universitas Negeri Surabaya.

Students and teachers usually face some problems during learning. The concept of programming and language syntax becomes an obstacle to learning programming. The average score of programming skills from first year students is 22.89 out of 110 points [2]. In addition, the dropout rate is between 30 to 40 percent, which shows how the students' spirit learns programming [3].

Computational thinking (CT) is a new method of problem solving. It uses computer principles such as logic, algorithms, decomposition, patterns, abstraction and evaluation. These techniques can solve the problem [4]. A study conducted by [5] argue that most first year students lack experience in programming and problem solving skills.

To avoid the complexity of programming language, new programmers should learn programming using pseudocode. Giordano & Carlisle [6] suggest that the novice programmer's learning experience and engagement can be improved with a pedagogy introduced into computer programming curricula that is focused on "algorithm visualization". Pseudocode is a description of what a computer does, expressed in natural language instead of programming language. Pseudocode is used as a detailed step in the program development process. Programmers use pseudocode to create models, or "mock-ups" of programs [7]. He simply states that Pseudocode was created to be used by computer programmers as "a tool" to design effective programs and "flowcharts" that then create "models of programs".

II. ITEM RESPONSE THEORY PARADIGM

Item response theory (IRT) model is initially developed to deal with items that are either true-false or dichotomized. In IRT, the mathematical model for item characteristic curve is a cumulative form and a logistical function. There are three models which are one logistic model, two parameters, and three parameters (1-PL, 2-PL, and 3-PL). One parameter model is known as Rasch model which is one of the most widely used model. The probability of test participants with the ability θ to correctly answer a *j* item is formulated in equation (1) [8].

$$p(\theta) = \frac{\exp(\theta - \delta_j)}{1 + \exp(\theta - \delta_j)} \quad (1)$$

The parameter δj is the point on the scale of ability where the probability of answering correctly is 0.5. This

parameter is a location parameter, indicating the position of the item characteristic curve in relation to the scale of ability. The greater δj parameter value, the greater ability that the test participants need to get a 50% chance of answering the item correctly. This is to say more difficult items. Thus, δj parameter is defined as an item difficulty parameter.

When the value of the ability of a group of test participants is transformed so that the average is 0 and the standard deviation is 1, then the value of bj changes is from -2.0 to +2.0. The value of δj approaches -2.0, which means a very easy item, while the value of δj is +2.0, which indicates a very difficult item.

One important benefit of the Rasch model is the so-called special objectivity [9]. The term means that the item parameter does not depend on the characteristics of the person performing the test, and that the person's (capability) parameter does not depend on the test item. As a result, the item parameters do not depend on the characteristics of the population. The nature of invariant item parameters and capability parameters is an important feature of IRT. The main advantage of IRT compared to CTT lies in the nature of invariance, which has been identified as a basic characteristic of measurement in psychology and education.

III. Computational Thinking

The definition of computational thinking has evolved from year to year. Wing [4] introduces computational thinking as universal skill that should be learned by every person. Computational thinking is not programming and do not associate to certain software. Wing [4] defines computational thinking as "solving problems, designing systems and understanding human behavior that draws on concepts fundamental to computing". According to the definition, computational thinking is abstraction and automation. Wing et al., [10] improve the definition by stressing the role of information processing: "Computational thinking is the processes involved in formulating problems and their solutions so that the solution are represented in a form that can be effectively carried out by an information-processing agent". Experts from Computer Science Teachers Association (CSTA) elaborate computational thinking as follows:

> *"Computational thinking is an approach to solving problems in a way that can be implemented with a computer. Students become not merely tool users but tool builders. They use a set of concepts, such as abstraction, recursion, and iteration, to process and analyze data, and to create real and virtual artifacts. Computational thinking is a problem solving methodology that can be automated and transferred and applied across subjects"* [11].

Computational thinking as process cannot be taught only through memorization, but it needs a demonstration to come in practice. Someone becomes a proficient reader and writer through practices. The same idea applies to computational thinking, a combination of some skills. When someone solves problems as a computation, firstly the person develops a model of the problem, and then processes it into a problem solving; both aspects rely on the model and principles regarding a computational method. In computational thinking, the thinking process involves two basic skills namely computation and problem solving.

Scratch is a visual programming language created by researchers from Massachusetts Institute of Technology in 2007.The purpose of Scratch is to make animation, games and interactive art. Although Scratch is easy to understand by young, the concept in Scratch cannot be separated from those computational thinking in other programming languages. When students make animation, interactive story, game, music, artwork with Scratch, at the same time they learn ideas of computing and mathematics. When they make Scratch programs, they learn the concept of computing such as iteration and conditional. They also acquire understanding of important thing about mathematical concepts such as coordinate, variable, and random number.

IV. Method

Instrument for assessing computational thinking consisted of 40 multiple-choice items. The instrument assessed the concept of conditional, iteration, array and procedure. Responses from 40 students to 40 tests were then analyzed using ConQuest's computer program [11]. ConQuest was utilized to estimate item difficulty parameter (δ) based on item response theory. The fit analysis was used to check the unidimensionality of programming test items. Unidimensionality was a very important assumption on the item response theory. Test items were unidimensional when the items measured one ability [12]. One indication of whether the test item was unidimensional was that data fit with the Rasch model [13]. To find out whether the Rasch model could predict the response of each respondent, it used *infit mean-square* (IMS) and *outfit mean-square* (OMS) statistics [13]. IMS and OMS statistics were the degree of compatibility measurement between observational data and the values predicted by the model.

The test items used in this study fit the Rasch model. Selected test items were based on the value of IMS and OMS. Linacre [15] constructed tables to interpret the meaning of the value of IMS and OMS. The test items' IMS and OMS values were ranging from 0.5 to 1.5.

The research was the implementation of teaching of visual programming language Scratch on students of Informatics Education study program at Faculty of Engineering of Universitas Negeri Surabaya (State University of Surabaya). The subjects of research were 40 students. The topics taught were the concept of computational thinking in visual programming language Scratch summarized in Table I.

Students were previously required to take pretest to identify their initial knowledge about computational thinking skills. Every topic was taught for two meetings. Every meeting took 100 minutes. Learning model used was direct instruction. With direct instruction model, first the teachers conveyed the purpose of why students learned visual programming language in relation to computational thinking. Next the teachers presented programs that became the model. Then the teacher guided the students on how to make programs using visual programming language Scratch.

After completing the course, students took paper and pencil test consisting of 40 multiple choice items. The items of tests assessed the concepts of computational thinking, namely conditional, iteration, array and procedure. Each concept is measured by 10 multiple choice items. The results of performance assessment and paper and pencil test were analyzed. The result of paper and pencil test was analyzed

using item response theory one parameter or Rasch model. ConQuest program [12] was used to estimate item parameter (δ) and ability parameter (θ) based on item response theory.

TABLE I. CONTENTS OF VISUAL PROGRAMMING

Concept	Contents
Variable	Data type, variable, operator
Conditional	Comparison operations, logical operations, conditional programming logic
Iteration	Iteration block, counter, nested loop.
Array	List that allows variables to hold a series of values.
Procedure	Sending and receiving message, modular programming, and recursion.

The normalized gain, introduced by Hake [15], became the standard measure for reporting scores on research-based concept inventories. Hake [15] defined the average of a normalized gain as a learning gain determined from a pre- versus post experiment could be calculated using a normalized gain. The normalized gain was a rough measure of the effectiveness of a course in promoting conceptual understanding. Hake [15] defined the average normalized gain as follows.

$$< g > = \left(\frac{\%<post>-\%<pre>}{100\%-<\%pre>}\right) \quad (2)$$

where <> indicates average scores. ‘‘High-g’’ courses as those with (<g>)≥0.7, ‘‘Medium-g’’ courses as those with 0.7.>(<g>)≥0.3, and ‘‘Low-g’’ courses as those with (<g>)<0.3.

V. RESULT

A. Instrument of Assessment

The test consists of 40 items of choice response or multiple choice. Responses from 40 students to 40 test points were then analyzed using the ConQuest program [12]. Table II shows the estimation results of selected response type item parameters. There are 40 item difficulty parameters (δ). Table II shows a summary of analysis of item parameter (δ*)* from multiple test. Item parameter is a point on ability scale where probability of answering right equal to 0.5. The parameter is a location parameter, indicating a position of item characteristic curve in conjunction with ability scale. The greater a location parameter, the greater ability required by students to obtain 50% the possibility of right answer. Item 2 has an item parameter of -2.52 which means that it is the easiest item. Furthermore, item 11 has an item parameter of 3.83, this is to say it is the most difficult item.

An important characteristic of a set of test items measuring a construct is that they are unidimensional. In Rasch analysis, if all coherent items form one scale, they are unidimensional. Keeves and Masters [17] propose that item fit is used to check unidimensionality. Fit items were analyzed using the ConQuest program [12]. There are two measures of fit items namely Infit Mean-Square (IMS) and Outfit Mean-Square (OMS). The results of the analysis are shown in Table III.

TABLE II. SUMMARY OF ANALYSIS OF ITEM PARAMETER (δ)

Concept	Parameter (δ)
Mean	0.00
SD	1.09
Minimum	-2.52
Maximum	2.83

TABLE III. ANALYSIS RESULTS OF ITEM FIT

Items No.	IMS	OMS	Items No.	IMS	OMS
1	1.01	1.02	21	1.06	1.06
2	0.93	0.98	22	1.10	1.10
3	0.93	0.94	23	1.09	1.05
4	1.10	1.05	24	0.92	0.92
5	1.04	1.03	25	1.02	1.00
6	0.94	0.95	26	1.03	1.02
7	0.88	0.95	27	1.09	1.06
8	1.41	1.08	28	1.05	1.04
9	1.13	1.05	29	1.15	1.08
10	1.07	1.07	30	0.95	0.96
11	0.54	0.98	31	0.94	0.94
12	0.87	0.89	32	1.01	0.99
13	1.03	1.04	33	0.89	0.90
14	1.22	1.14	34	0.94	0.94
15	1.23	1.14	35	0.87	0.89
16	1.06	1.06	36	1.02	1.01
17	1.17	1.10	37	0.95	0.95
18	0.97	0.98	38	0.83	0.86
19	1.01	1.01	39	0.85	0.89
20	1.07	1.05	40	1.01	1.02

IMS and OMS value approaching 1.0 indicates little distortion to the measurement system. A value of 1.0 is the expected value. If the fit observation data with Rasch model, then the expectation value is 1.0. IMS and OMS value listed in Table IV shows that nearly all the test items useful for measurement. Test items that do not fit is the number 11 which has a value of IMS 0.54.

B. Teaching and Learning

Based on posttest result from 40 students can be known which concept is easy and which is difficult, as listed in Table IV. Based on the score can be sorted level ease of understanding concept of student computational thinking, successively starts of the easiest to difficult are conditional, procedure, array and iteration.

TABLE IV. STUDENT'S ANSWERS (%)

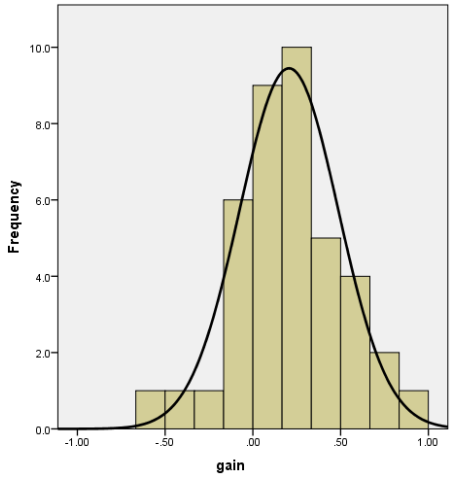
No	Score / Concept	0	1
1.	Conditional	41.20	58.40
2.	Iteration	63.70	32.30
3.	Array	65.20	34.80
4.	Procedure	40.60	59.40

TABLE V. SUMMARY OF PRE-TEST AND POST-TEST SCORE

	N	Min.	Max.	Mean	SD
Pretest	40	3.00	20.00	10.600	3.95358
Posttest	40	5.00	23.00	13.750	4.47643

Pretest and posttest scores for computational thinking are listed in Table V. The average of pretest and posttest scores are 10.60 and 13.75, respectively.

The learning gain distribution is depicted in Figure 1. The average of gain scores is 0.20 which means computer programming learning is not effective. Figure 2 shows correlation between learning gain and pretest scores. Gain average has negative correlation with pretest scores (r = -0.174).



	N	Minimum	Maximum	Mean	Std. Deviation
gain	40	-.58	.88	.2056	.28143
Valid N (listwise)	40				

Fig. 1. Distribution of individual learning gain (N = 40)

VI. DISCUSSION

Pseudocode instrument is used to measure understanding of computational thinking concept. To test whether the computational thinking construct is unidimensional or not, the Rasch fit analysis is used. Evaluation based on infit and outfit statistics states that good test items have mean square values greater than 0.5 and less than 1.5 [15]. A square mean value greater than 2.0 means more errors than information obtained from a test item. The pseudocode instrument test items have infits ranging from 0.54 to 1.45 and outfits with ranges from 0.89 to 1.14. The infit and outfit indexes mean there is consistency between data and unidimnsional Rasch model. The construct of computational thinking is unidimensi.

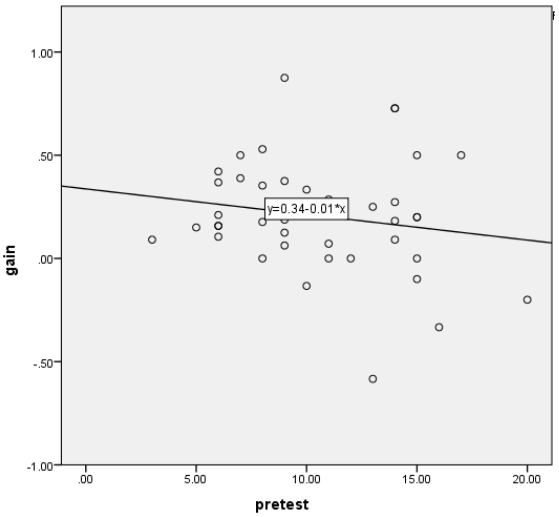


Fig. 2. Correlation of gain with pre-test score (N = 40), $r^2 = 0.030$

Students with low pretest score will have high gain. The reason is that students with low initial score have a higher chance of improvement so they have the potential to gain greater than student with high scores. Therefore, students with high initial scores will have lower gain because it is more difficult to obtain a high end score if it starts from a high initial score.

Normalized learning gains represent an accurate estimate of learning only when students perform better in post-tests than in pre-tests [19], which is not always the case when looking at academic performance in Higher Education [16].

VII. CONCLUSION

The δ parameter is the location parameter, which indicates the position of the item characteristic curve in relation to the ability scale. From 40 computer programming tests, there is 1 item of unfit tests based on the Rasch model. Parameter δ has a range between -2.52 to 2.83. The concepts of computational thinking taught through visual programming language Scratch are conditional, iteration, array and procedure. The easiest concept to learn is procedure while the most elusive one is array. The average of gain scores is 0.20 which means that computer programming learning is not effective. The correlation between learning gain and pretest scores is -0.174.

REFERENCES

[1] L. A. Vaca-Cárdenas, F. Bertacchini, L. Gabriele, A. Valenti, D. E. Olmedo and E. Bilotta, "Coding with Scratch: The design of an educational setting for Elementary pre-service teachers," in *Interactive Collaborative Learning (ICL)*, Florence - Italy, 2015.

[2] L. Ma, J. Ferguson, M. Roper and M. Wood, "Investigating and improving the models of programming concepts held by the novice programmers," *Journal Computer Science Education,* vol. 21, no. 1, pp. 57-80, 2011.

[3] T. Beaubouef and J. Mason, "Why the high attrition rate for computer science students: some thoughts and observations," *ACM SIGCSE Bulletin,* vol. 37, no. 2, pp. 103-106, 2005.

[4] J. M. Wing, "Computational thinking," *Communications of the ACM,*

vol. 49, no. 3, pp. 33-35, 2006.

[5] E. H. Cohen, "Information and beyond: issues in informing science and information technology," *Journal of Issues in Information Science and Information Technology,* vol. 4, no. 1, pp. 271-289, 2007.

[6] J. C. Giordano and M. C. Carlisle, "Toward a More Effective Visualization Tool to Teach Novice Programmers," in *Proceedings of the 7th Conference on Information Technology Education, SIGITE 2006*, Minneapolis, Minnesota, USA, 2006.

[7] T. Gaddis, Starting Out with Programming Logic and Design (4th ed), Boston: Pearson, 2016.

[8] S. E. Embretson and S. P. Reise , Item Response Theory for Psychologists (Multivariate Applications Series), Mahwah: Lawrence Erlbaum Associates., 2000.

[9] W. Fisher, "Reliability statistics.," *Rasch Measurement Transactions,* vol. 6, no. 3, p. 238, 1992.

[10] J. M. Wing, "Research notebook: Computational thinking—What and why?," *Magazine: The Carnegie Mellon University,* 2011.

[11] . T. Ball and B. Zorn, "Teach Foundational Language Principles," *Communications of the ACM,* vol. 58, no. 5, pp. 30-31, 2015.

[12] M. Wu, R. Adams and M. R. Wilson, ConQuest: Generalised item response modeling software, Camberwell: Australian Council for Educational Research, 1998.

[13] C. L. Hulin, F. Drasgow and C. K. Parso, Item Response Theory: Application to Psychological Measurement, Homewood: Dow Jones - Irwin, 1983.

[14] E. Smith Jr , "Evidence for the reliability of measures and the validity of measure interpretation: A Rasch measurement perspective," *Journal of Applied Measurement,* vol. 2, no. 3, pp. 281-311, 2001.

[15] J. M. Linacre, "What do Infit and Outfit, Mean-square and Standardized mean?," *Rasch Measurement Transaction,* vol. 16, no. 2, p. 878, 2002.

[16] R. R. Hake, "Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses," *American Journal of Physics,* vol. 66, no. 1, pp. 64-74, 1998.

[17] G. N. Masters and J. P. Keeves, Advances in measurement in educational research and assessment / edited by Geofferey N. Masters and John P. Keeves., Amsterdam: Pergamon, 1999.

[18] . J. D. Marx and K. Cummings, "Normalized change," *American Journal of Physics,* vol. 75, no. 1, pp. 87-91, 2007.

[19] J. L. Jensen, T. A. Kummer and . P. D. d. M. Godoy, "Improvements from a Flipped Classroom May Simply Be the Fruits of Active Learning," *CBE Life Science Education,* vol. 14, no. 1, pp. 71-76, 2015.

[20] V. Barr and C. Stephenson, "Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community?," *ACM Inroads,* vol. 2, no. 1, pp. 48-54, 2011.

[21] J. Martín-Gutiérrez, W. Benesova, M. D. Meneses and C. E. Mora, "Augmented Reality to Promote Collaborative and Autonomous Learning in Higher Education," *Computers in Human Behavior,* vol. 51, pp. 752-761, 2015.

[22] Y. Yalaki, "Simple Formative Assessment, High Learning Gains in College General Chemistry," *Eurasian Journal of Educational Research,* vol. 10, no. 40, pp. 223-241, 2010.