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

Implementation of Computational Thinking Concepts in ICT Learning Using Scratch Programming

Rina Harimurti[1], Anita Qoiriah[1], Ekohariadi[1], Munoto[2]
Informatics Department[1]
Electrical Engineering Department[2]
Universitas Negeri Surabaya
Surabaya, Indonesia
rinaharimurti@unesa.ac.id

***Abstract*— Computational thinking (CT) is a fundamental skill that everyone should have, dealing with problem solving, designing systems, and understanding human behavior by drawing on fundamental concepts to computer science. This study aims to apply the concept of computational thinking in ICT learning in secondary schools using Scratch programming language. Scratch programming language is a programming language developed by MIT (Massachussetts Institute of Technology) Media Lab, aimed at teaching programming to teenagers and novice programmers. This research is a quantitative research with a sample of 40 students. In this research developed CT test items with 5 indicators derived from CT variables. The results of the student responses were then analyzed using Rasch model analysis and Classic Test Theory using the computer program Test Analysis Program to estimate item difficulty parameter (p) and ConQuest used to estimate the difficulty parameter of item (b) based on the item responses.**

Keywords*— computational thinking, ICT Learning, Rasch model, Scratch programming language

I. INTRODUCTION

The development of information technology in all fields demands human resources that have the ability to adapt to changes arising from the development of information technology. Therefore, ICT (information and communication technology) learning has been done in many schools from elementary, middle to upper level to prepare students to face the development of information technology.

The application of technology and information in all fields requires the ability to think computational which is now a trend. This fundamental capability must be owned by all students in the digital age. Three dimensions of computational thinking are computational concepts, computational practices and computational perspectives. [1]. In recent years, the availability of user-friendly and user-friendly programming has improved this is a way for educators to explore how this computational thinking can be easily introduced to learners.

Computer programming involves skills in critical thinking, problem-solving, computational thinking and new system designs. The requirement for computational thinking and improvements in the programming skills of individuals remain to obtain widespread attention to improved arrange future students. Studies expose that many issues in learning programming create from the complexity of concepts such as variables, loops, arrays, functions, and of syntax in programming languages.

Several introductory programming environments were considered to minimize common errors in programming such as syntax errors and logic. A novel programming languages based on blocks are therefore developed. Each block is a component of the programming language which are a control structure, an operator, a variable, or a function etc. These components can be joint with "drag and drop" in an in-built way in line with a certain strategic logic to form a computer program.

Some studies reveal that Scratch programming tool is a stimulator of assisting students in developing skills of computational thinking. Recognizing the importance of computational thinking concept, this study aims to apply the concept of computational thinking to ICT learning in secondary schools using Scratch programming language by analyzing test items from the responses given by the students.

A. Computational Thinking (CT)

CT was first introduced by Jeannete Wing which is a method for solving problems, designing systems, and understanding human behavior by drawing on fundamental concepts to computer science can be done by humans or machines (computer). CT is a fundamental skill for everyone. These skills do not replace creativity, logical thinking and critical thinking but increase the capacity of the concept of thinking in problem solving. According to Hu, CT represents a cognitive process that accommodates logic, algorithmic, analytical, mathematical, technical and creative thinking [3]. CT should be introduced early to the students so that they understand the development of information technology globally.

Computational thinking is thinking using logic, doing things step by step and being able to make decisions when faced with two different possibilities. The viewpoint of CT comprehension is viewed from a wide range, in this study focusing point of view on the field of education. CT is in the realm of educational innovation a part of problem-solving skills that students must acquire to thrive in a digital world full of objects that can be driven by software. [4] From some existing reference then CT can be breakdown into 5 indicators

which will be developed into test item to know student response to applying of CT at learning. The 5 indicators are 1) Understanding the basic concepts of computer science, 2) Understanding the basic concepts of programming, 3) Understanding programming algorithms, 4) Understanding basic programming languages, and 5) Skills to operate computer basics.

B. Scratch Programming

Programming language is known as something hard to learn even many who believe only smart people who can operate it. However, the assumption is no longer appropriate, because now has developed a programming language that is relatively easy to learn is Scratch programming. Scratch is a programming language developed by the MIT (Massachusetts Institute of Technology) Media Lab, aimed at teaching programming to teenagers and novice programmers. [5] Scratch is a new programming language released in May 2007. Scratch supports the development of computer games, interactive stories, graphics and computer animation and other multimedia projects. Scratch consists of programming languages created from different blocks and made in graphical form. The scratch program as shown in Fig. 1, is constructed from a graphical block arrangement. Scratch blocks resemble puzzle pieces arranged together. The Scratch block can only be compiled together in an easy way to prevent combinations of programming errors especially for novice programmers. In this way Scratch creates the correct programming syntax and ensures that novice programmers learn the proper way to compose and formulate programming logic.

Scratch is a new programming language different from programming languages like Visual Basic. Scratch programming is not based on text. Examples of Visual Basic programming applications are as follows.

```
If strCurrentAction = "FillCircle" Then
  Dim objCoordinates As Rectangle
  objCoordinates = _
  New Rectangle(Math.Min(objEnd.X, objStart.X), _
  Math.Min(objEnd.Y, objStart.Y), _
  Math.Abs(objEnd.X - objStart.X), _
  Math.Abs(objEnd.Y - objStart.Y))
  Pick_Color_And_Draw("FillCircle", objCoordinates)
End If
```

In a text-based programming language, the command code is formulated by following a complex set of syntax rules. Error writing that does not match the syntax rules of this programming language will cause the application cannot run. Scratch uses a different approach. The Scratch application project is built by selecting and compiling programming blocks in graphical form, as shown in Fig. 2.

Using a block of code to replace program commands in complex text form, Scratch simplifies application development using the same basic logic and programming concepts with other programming languages. As in Fig. 2, each code block describes a different command. Each block must match each other as in composing a puzzle. Some code blocks can be configured, to specify something like the number of times the job should be executed, the displayed text or the colors used when displaying the screen. Despite using a block of graphical code, Scratch supports the same basic programming and construction techniques as other programming languages.

Fig. 1. The script block used as the base of script writing

Fig. 2. Example of how to outline the logic of programming in a Scratch app

II. METHODS

This research is a quantitative research. The population in this study is high school students who get ICT learning by using Scrath programming, by taking samples of 40 students in 1 class. In this study, the instrument developed in the form of test items consists of 30 multiple choice items that will be used to measure students' computational thinking ability by using Scratch programming. Table 1 shows the development of test items.

TABLE I. TEST ITEMS DEVELOPMENT

Variable	Indicators	Number of Test Item
Computational Thinking	Understanding the basic concepts of computer science	6
	Understanding the basic concepts of programming	6
	Understanding programming algorithms	6
	Understanding basic programming languages	8
	Skills to operate computer basics	6

The instrument was then tested on the sample, which had previously been validated by the relevant expert. The results of the student responses were then analyzed using Rasch model analysis and Classic Test Theory by using the computer program Test Analysis Program to estimate grain difficulty parameter (p) and ConQuest was used to estimate the difficulty parameter of item (b) based on the grain response. Rasch was the first to develop a logistic model of one parameter. According to van der Linden & Hambleton (1997) on the Rasch model, people are given latent ability level

characteristics ζ and grains are characterized by difficulty levels δ [6].

The programming concepts of Scratch which are implemented in this study is adopted from [7] as listed in Table II.

TABLE II. TEST PROGRAMMING CONCEPTS OF SCRATCH

Concept	Indicators
Problem solving	Design an algorithm
	Set up Sprite area and Scripts
	Test program execution
Integrated development environment	Graphic interface, consisting of a code block area, Sprites and a Sprite area
Implementation of program	Place Sprites in code block area. Link Scripts to Sprites by dragging in code blocks.
Syntax	Built-in blocks of code
Variables and data types	Any name can be given.
	Visual representation of the content by monitors
	Shapes of parameter slots indicate data types, namely Boolean, Numeric and String
	No need to distinguish between data types, and no conversion of data types.
	Scope of variables dependent if a variable belongs to a Sprite or a Stage.
Execution of programs	Click on green flag
	Programs will usually execute.
Decision and looping structures	Represented visually with code blocks.
	Variety of structures to choose from
	Concept of conditional and unconditional loops required to write effective programs. Need to know how to define Boolean statements.
Object-oriented programming	Object-based language, but analogies to object-oriented concepts can be made
	Sprites can be seen as objects with properties and behavior.
	Mainly procedural approach within one Script.
Error handling and testing	Logical errors and execution errors can occur when executing programs.
	Monitors display content of variables while program is executing.
	Execution of programs can be done step by step, in slow motion, to resolve logical errors.

III. RESULTS AND DISCUSSION

Response to 30 test items was then analyzed using ConQuest computer program. Table II shows the results of the multiple-choice item parameter estimation. There are 30 item difficulty parameters (β). According to the results as in Fig. 3, the estimate of data processing using ConQuest. The value in the estimate column is showing the value of β, the parameter β is the point on the scale of ability where the probability of answering the bell is 0.5. The parameter is the location parameter, which indicates the position of the grain characteristic curve in relation to the scale of capability. The greater the parameter value of β, the greater the capability required to obtain a 50% chance of answering the item correctly. The β parameter is also called the item difficulty parameter.

VARIABLES			UNWEIGHTED FIT			WEIGHTED FIT		
item	ESTIMATE	ERROR^	MNSQ	CI	T	MNSQ	CI	T
1	0.953	0.425	0.97	(0.56, 1.44)	-0.1	1.00	(0.00, 2.00)	0.2
2	0.385	0.374	1.05	(0.56, 1.44)	0.3	1.02	(0.30, 1.70)	0.2
3	-2.544	0.306	1.01	(0.56, 1.44)	0.1	1.00	(0.69, 1.31)	0.1
4	-0.718	0.300	1.00	(0.56, 1.44)	0.1	1.00	(0.73, 1.27)	0.0
5	-1.772	0.282	1.00	(0.56, 1.44)	0.1	1.00	(0.92, 1.08)	0.1
6	-0.595	0.306	0.99	(0.56, 1.44)	0.0	0.99	(0.69, 1.31)	-0.0
7	-0.465	0.313	0.99	(0.56, 1.44)	0.0	0.99	(0.64, 1.36)	0.0
8	-0.326	0.321	1.00	(0.56, 1.44)	0.1	1.00	(0.59, 1.41)	0.1
9	-0.718	0.300	1.02	(0.56, 1.44)	0.2	1.02	(0.73, 1.27)	0.2
10	-0.595	0.306	1.01	(0.56, 1.44)	0.1	1.01	(0.69, 1.31)	0.1
11	-1.978	0.285	0.99	(0.56, 1.44)	0.0	0.99	(0.87, 1.13)	-0.1
12	-1.368	0.282	0.99	(0.56, 1.44)	0.0	0.99	(0.92, 1.08)	-0.3
13	2.107	0.517	0.97	(0.56, 1.44)	-0.1	1.00	(0.00, 2.89)	0.3
14	-1.469	0.281	0.99	(0.56, 1.44)	0.0	0.99	(0.94, 1.06)	-0.2
15	0.637	0.397	1.03	(0.56, 1.44)	0.2	1.01	(0.17, 1.83)	0.2
16	0.953	0.425	1.05	(0.56, 1.44)	0.3	1.01	(0.00, 2.00)	0.2
17	2.107	0.517	1.10	(0.56, 1.44)	0.5	1.01	(0.00, 2.89)	0.3
18	1.386	0.463	1.01	(0.56, 1.44)	0.1	1.00	(0.00, 2.28)	0.2
19	0.385	0.374	1.04	(0.56, 1.44)	0.2	1.01	(0.30, 1.70)	0.2
20	-0.176	0.331	1.01	(0.56, 1.44)	0.1	1.00	(0.53, 1.47)	0.1
21	0.385	0.374	1.00	(0.56, 1.44)	0.1	1.01	(0.30, 1.70)	0.1
22	1.386	0.463	1.06	(0.56, 1.44)	0.3	1.01	(0.00, 2.28)	0.2
23	-1.056	0.288	0.99	(0.56, 1.44)	0.0	0.99	(0.83, 1.17)	-0.1
24	0.385	0.374	1.03	(0.56, 1.44)	0.2	1.01	(0.30, 1.70)	0.1
25	-0.326	0.321	0.97	(0.56, 1.44)	-0.1	0.98	(0.59, 1.41)	-0.0
26	1.386	0.463	0.95	(0.56, 1.44)	-0.2	1.00	(0.00, 2.28)	0.2
27	0.385	0.374	0.96	(0.56, 1.44)	-0.1	0.99	(0.30, 1.70)	0.1
28	0.953	0.425	1.00	(0.56, 1.44)	0.1	1.00	(0.00, 2.00)	0.2
29	-0.326	0.321	0.98	(0.56, 1.44)	-0.0	0.99	(0.59, 1.41)	0.0
30	2.107	0.517	0.92	(0.56, 1.44)	-0.3	1.00	(0.00, 2.89)	0.3

Fig. 3 The data analysis process uses ConQuest

The next step is data processing using Test Analysis Program (TAP10K) software, the results of data analysis are shown in the table below. TAP is used to estimate grain difficulty parameters based on classical test theory

Fig. 4 presents data processing table. In the column "Number Correct" is the number of students who answered correctly on each item. In the "Diff item" column is the P value expressing the probability of answering correctly from the respondent having the ability to answer an item having β difficulty.

Fig. 5 shows the characteristic curve of item mode with one parameter, where the horizontal axis is the value of β that is the ability of students to answer the test item, while the vertical axis is the probability of correctly answer each test item. The relationship between parameter β and the difficulty of the grains is shown in the figure 5.

TITLE: Computational Thinking
COMMENT:

Quick Item Analysis Results:

Item	Key	Number Correct	Item Diff.	Disc. Index	# Correct in High Grp	# Correct in Low Grp	Point Biser.	Adj. Pt Bis
Item 01 (2)#		8	0.24	0.29	6 (0.40)	1 (0.11)	0.39	0.16
Item 02 (4)#		5	0.15	0.27	4 (0.27)	0 (0.00)	0.47	0.29
Item 03 (3)#		1	0.03	0.00	0 (0.00)	0 (0.00)	-0.04	-0.13
Item 04 (3)#		3	0.09	0.02	2 (0.13)	1 (0.11)	-0.01	-0.17
Item 05 (4)#		3	0.09	0.20	3 (0.20)	0 (0.00)	0.16	0.01
Item 06 (1)#		2	0.06	0.07	1 (0.07)	0 (0.00)	0.30	0.17
Item 07 (2)#		3	0.09	-0.04	1 (0.07)	1 (0.11)	-0.01	-0.17
Item 08 (2)#		14	0.42	0.24	7 (0.47)	2 (0.22)	0.19	-0.09
Item 09 (1)#		1	0.03	0.07	1 (0.07)	0 (0.00)	0.06	-0.04
Item 10 (2)#		21	0.64	-0.11	10 (0.67)	7 (0.78)	-0.08	-0.33
Item 11 (1)#		3	0.09	0.20	3 (0.20)	0 (0.00)	0.16	0.01
Item 12 (2)#		9	0.27	0.18	6 (0.40)	2 (0.22)	0.21	-0.04
Item 13 (3)#		2	0.06	0.07	1 (0.07)	0 (0.00)	0.16	0.02
Item 14 (1)#		6	0.18	0.22	5 (0.33)	1 (0.11)	0.29	0.08
Item 15 (2)#		8	0.24	0.29	6 (0.40)	1 (0.11)	0.39	0.16
Item 16 (4)#		3	0.09	0.20	3 (0.20)	0 (0.00)	0.46	0.32
Item 17 (1)#		19	0.58	0.11	10 (0.67)	5 (0.56)	0.05	-0.22
Item 18 (5)#		3	0.09	0.07	1 (0.07)	0 (0.00)	-0.01	-0.17
Item 19 (1)#		16	0.48	-0.16	6 (0.40)	5 (0.56)	-0.11	-0.36
Item 20 (2)#		4	0.12	0.20	3 (0.20)	0 (0.00)	0.28	0.10
Item 21 (1)#		1	0.03	0.07	1 (0.07)	0 (0.00)	0.25	0.16
Item 22 (1)#		12	0.36	0.47	7 (0.47)	0 (0.00)	0.39	0.14
Item 23 (4)#		0	0.00	0.00	0 (0.00)	0 (0.00)	0.00	0.00
Item 24 (1)#		6	0.18	0.13	2 (0.13)	0 (0.00)	0.16	-0.06
Item 25 (2)#		7	0.21	0.22	5 (0.33)	1 (0.11)	0.38	0.16
Item 26 (1)#		2	0.06	0.07	1 (0.07)	0 (0.00)	0.01	-0.12
Item 27 (3)#		2	0.06	0.13	2 (0.13)	0 (0.00)	0.16	0.02
Item 28 (1)#		5	0.15	0.02	2 (0.13)	1 (0.11)	0.14	-0.06
Item 29 (2)#		6	0.18	0.02	2 (0.13)	1 (0.11)	0.07	-0.14
Item 30 (5)#		3	0.09	0.20	3 (0.20)	0 (0.00)	0.34	0.19

Fig. 4 Data processing using TAP10K software

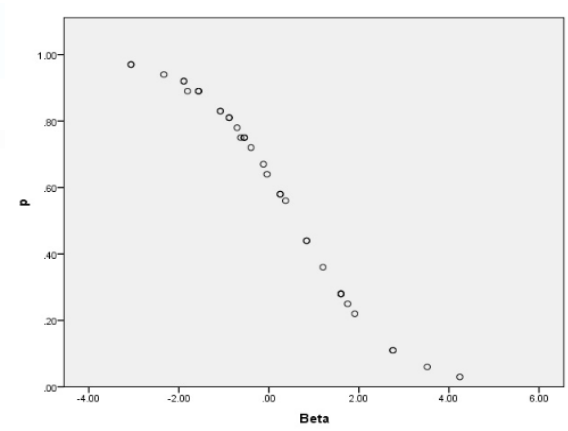

Fig. 5 Graph of the relationship betweenβ and p

In Rasch analysis, if all grains form a scale then the grain is unidimensional. To check unidimensi used grain *fit*. *Fit* items are analyzed using ConQuest. There are two sizes of *fit* grains IMS (Infit Mean-Square) and OMS (Outfit Mean-Square). The results of the analysis are shown in Figure 6.

TABLE III. VALUE RANGE IMS AND OMS

Mean-square value	Implication for Measurement
>2.0	Distorts or degrades the measurement system. May be caused by only one or two observations
1.5 – 2.0	Unproductive for construction of measurement, but not degrading
0.5 – 1.5	Productive for measurement
<0.5	Less productive for measurement, but not degrading. May produce misleadingly high reliability and separation coefficients

Using the benchmark in Table III, the results of the IMS and OMS analysis concluded that the value of IMSs and OMSs in Figure 6 show that all items are useful for measurement. This is because the value of IMSs and OMSs is approximately 1.0 productive for measurement

item	IMS	OMS
1	0.97	1.00
2	1.05	1.02
3	1.01	1.00
4	1.00	1.00
5	1.00	1.00
6	0.99	0.99
7	0.99	0.99
8	1.00	1.00
9	1.02	1.02
10	1.01	1.01
11	0.99	0.99
12	0.99	0.99
13	0.97	1.00
14	0.99	0.99
15	1.03	1.01
16	1.05	1.01
17	1.10	1.01
18	1.01	1.00
19	1.04	1.01
20	1.01	1.00
21	1.00	1.01
22	1.06	1.01
23	0.99	0.99
24	1.03	1.01
25	0.97	0.98
26	0.95	1.00
27	0.96	0.99
28	1.00	1.00
29	0.98	0.99
30	0.92	1.00

Fig. 6 Fit item analysis results

IV. CONCLUSION

In this research can be concluded that the important indicator in CT there are 5, which are 1) Understanding the basic concepts of computer science, 2) Understanding the basic concepts of programming, 3) Understanding programming algorithms, 4) Understanding basic programming languages, and 5) Computer literate operating skills. Furthermore, from the five indicators are made item test items to measure students' ability to implement CT in ICT learning. From the results of the test analysis results obtained β values close to -2.0 means the grains are very easy, otherwise the value of β approaching +2.0 means very difficult grains. Rasch's formula calculates logistic probabilities with the same slope. The curves differ only in place on the scale of ability.

REFERENCES

[1] Sze Yee Lye , Joyce Hwee Ling Koh, Review on teaching and learning of computational thinking through programming: What is next for K-12?, Computer in Human Behaviour 41, 2014, pp 51-61.

[2] Jeannette M. Wing, Computational Thinking, Communications Of The ACM, March 2006, vol 49, No 3, pp 33-35.

[3] Ana Liz Souto O. de Araujo, Wilkerson L. Andrade, Dalton D. Serey Guerrero, A Systematic Mapping Study on Assessing Computational Thinking Abilities, IEEE, 2016.

[4] Marcos Roman-Gonzalez, Juan-Carlos Perez-Gonzalez, Carmen Jimenez-Fernandez, Which cognitive abilities underlie computational

thinking? Criterion validity of the Computational Thinking Test, Computer in Human Behaviour xxx, 2016, pp 1-14.

[5] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, Yasmin Kafai Communications of the ACM, November 2009, Vol. 52 No. 11, Pages 60-67 doi: 10.1145/1592761.1592779

[6] Wim J. van der Linden, Ronald K. Hambleton, 1997, Modern Item Response Theory, Springer-Verlag New York.

[7] Sukie van Zyl, Elsa Mentz and Marietjie Havenga, 2016, Lessons Learned from Teaching Scratch as an Introduction to Object-oriented Programming in Delphi, African Journal of Research in Mathematics, Science and Technology Education

[8] James J. Lu, George H. L. Fletcher, *Thinking About Computational Thinking, SIGCSE'09,* March 3–7, 2009, Chattanooga, Tennessee, USA, Copyright 2009 ACM 978-1-60558-183-5/09/03

[9] Osman Yasar, *The Essence of Computational Thinking*, State University of New York, Brockport

[10] Hui-Chi, Chuang; Chiu-Fan, Hu; Cheng-Chih, Wu; Yu-Tzu, Lin, *Computational Thinking Curriculum for K-12 Education-A Delphi Survey,* 2015 International Conference on Learning and Teaching in Computing and Engineering

[2