2019 IEEE International Conference on Consumer Electronics
- Taiwan (ICCE-TW)

A New Evaluation Metrics for Block-based Python Code

Zheng Liu
Beijing Key Laboratory of
Intelligent Telecommunications
Software and Multimedia
Beijing University of
Posts and Telecommunications
Beijing 100876, China
Email: woniu@bupt.edu.cn

Hong Luo
Beijing Key Laboratory of
Intelligent Telecommunications
Software and Multimedia
Beijing University of
Posts and Telecommunications
Beijing 100876, China
Email: luoh@bupt.edu.cn

Xiaolin Chai
Beijing Key Laboratory of
Intelligent Telecommunications
Software and Multimedia
Beijing University of
Posts and Telecommunications
Beijing 100876, China
Email:xiaolin_chai@bupt.edu.cn

***Abstract*—The programming education for the students set off a global enthusiasm. The Block-Python programming tools can fill the gap from block-based programming to language coding. However, there is little work focus on the evaluation metrics for block-based python code. The past block-oriented evaluation metrics based on Computational Thinking (CT) were not language-oriented, and program volume was often ignored. In this paper, we propose a new evaluation metrics for block-based python code, which combines the CT dimensions and program volume. The experimental results show that the correlations between the proposed metrics and Halstead, McCabe complexity algorithms are both above 0.7, which is higher than Dr.Scratch.**

I. Introduction

Nowadays, there have been many programming learning tools for K-12 students, such as Scratch from MIT for lower levels, and Block-C, Block-Python tools for advanced students who need to learn a programming language. In terms of evaluation metrics, a suitable evaluation metrics for block-based python code is still demanding. First, the characteristics of the building blocks should be preserved. Second, the characteristics of the python language should also be considered. There are many traditional evaluation metrics for python[1], such as Halstead and McCabe complexity, which evaluate the volume and structural complexity of the program. However, both of them are difficult for beginners. Computational thinking abilities are often taken into consideration when evaluating Scratch-like projects[2]. Jesús Moreno-León et al. proposed Dr.Scratch for Scratch projects [3][4] . The tool has a certain degree of recognition, but not suitable for Block-Python. Compare the examples below:

I: Example 1

```
a=5
b=10
if b>a:
        print ‘too big!‘
else:
        print ‘too small!‘
```

II: Example 2

```
import random
a=5
b=random.randint(0,9)
if b>a:
    print ‘too big!‘
elif b==a:
    print ‘equal!‘
else:
    print ‘too small!‘
if b==0:
    print ‘b is 0!‘
```

The two examples reflect different level of mastery of python language. The use of “import”, multi-branch structure, etc., reflect higher CT capabilities. And McCabe complexity also shows that the circle complexity of E2 is much higher than E1, but the scores produced by Dr.Scratch are the same.

In this paper, we propose a new evaluation metrics for block-based python code, which jointly considers the dimensions of CT and program volume. The experimental results show that the correlation between the proposed metrics and Halstead, McCabe complexity algorithms are both above 0.7, which is higher than Dr.Scratch.

II. Evaluation metrics for block-based python code

New evaluation metrics jointly considers the dimensions of CT and program volume. Block-Python projects are different from Scratch's. First, the outcomes of Scratch projects tend to be open, while the Block-Python is for higher-level learners, who is considered to have a certain programming foundation and are focus more on solving scientific problems. Second, Block-Python projects show more characteristics of python.

For each CT dimension, we refactor and refine the evaluation according to python; for final results, we use a “weight-accumulate” method to reflect the influence of the program volume.

The obtained evaluations weights are given in TABLE I. Each row represents knowledge points for CT dimension, and different knowledge points are given different weights according to difficulty. The final result is weighted-accumulated by the following steps:

1) Convert the source code to be evaluated into an Abstract Syntax Tree(AST).
2) Traverse the AST and calculate the number of each knowledge points according to Tab.1. $N[i][j]$ is the number of the j-th knowledge points in the i-th CT dimension.
3) Calculate the $score[i]$ for each CT dimension, according to Equation 1.

$$score[i] = \min(\sum_{j=1}^{4} N[i][j] \times weight[i][j], 30) \quad (1)$$

978-1-7281-3279-2/19/$31.00 ©2019 IEEE

TABLE I: Weights of Different Knowledge Points for Each CT Dimension

CT Dimensions	1	2	3	4
abstraction and problem decomposition	built in functions	external functions	creation of custom functions	definition and usage of class
user interactivity	standard output	standard input	IO of files, redirection	IO of plot, hardware
flow control	simple loops	parallel simple loops	nested loop construct	complex construct in loop bodies

logical thinking	if	if else
		if elif else
			logic operators	more than 4 exits in a code block

data representation	string and numbers	usage of variables	list, dictionary, tuple	usage of objects

4) Accumulate the scores of the five CT dimensions and get the final sum according to Equation 2.

$$sum = \min(\sum_{i=1}^{5} score[i], 100) \tag{2}$$

III. EVALUATION AND RESULTS ANALYSIS

Conduct the proposed evaluation metrics and Dr.Scratch on 156 samples. The samples are produced by Block-Python tool. The correlations between the two metrics and Halstead-Length, Halstead-Vocabulary and McCabe are calculated, Fig.1-3 show the compared results.

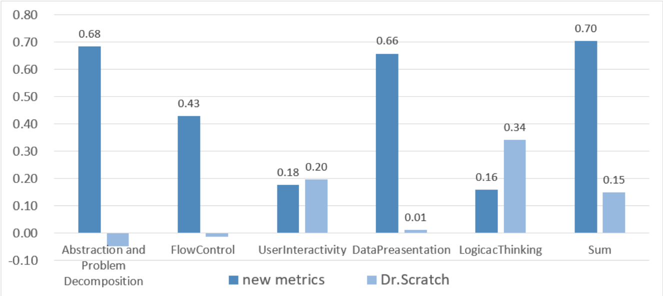

Fig. 1: Correlations between Halstead-Length and Proposed Metrics.

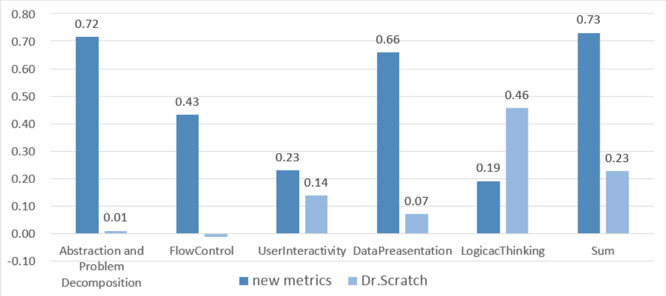

Fig. 2: Correlations between Halstead-Vocabulary and Proposed Metrics.

It can be seen from the above three figures that the correlations between the sum and the three indicators of the two complexity are all above 0.70, while the sum of Dr.Scratch and the two complexity metrics show a weak correlation. The reasons are as follows: First, the proposed metrics is language-oriented, ensuring that the python characteristics are not ignored. Second, when evaluating CT abilities, we also take the volume of program into consideration, while Dr.Scratch just focuses on whether the CT skills are used.

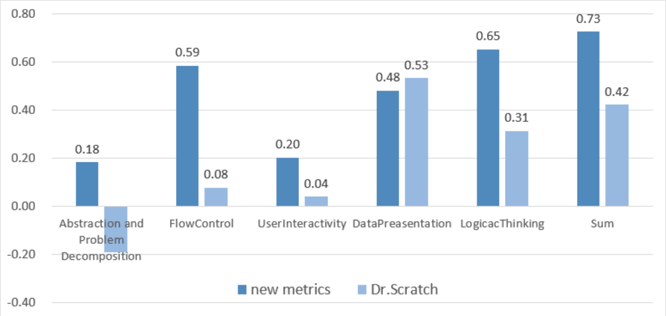

Fig. 3: Correlations between McCabe and Proposed Metrics.

For some dimensions, they have a strong correlation with particular complexity metrics, Figure 3 shows the example: the correlation between Logical Thinking and McCabe complexity is 0.65, explaining that Logical Thinking reflects the structural complexity of a program.

IV. Conclusion

In this paper, we propose the new evaluation metrics for block-based python code. The metrics is based on CT, and combined with specific language knowledge points and program volume. For the final result, the sum produced by new evaluation metrics is in the range of 0-100, which has a higher degree of discrimination than 0-21 in Dr.Scratch. It demonstrateds that compared with Dr.Scracth, new metrics is more accurate, and is more suitable for block-based python code.

ACKNOWLEDGMENT

This work is partly supported by the National Natural Science Foundation of China under Grant 61877005, 61772085, 61672109.

References

[1] D. I. D. Silva, N. Kodagoda, and H. Perera, “Applicability of three complexity metrics,” in *International Conference on Advances in ICT for Emerging Regions*, 2012, pp. 82–88.

[2] P. Techapalokul and E. Tilevich, “Understanding recurring software quality problems of novice programmers,” 2017.

[3] J. Moreno-León, G. Robles, and M. Román-González, “Comparing computational thinking development assessment scores with software complexity metrics,” in *Global Engineering Education Conference*, 2016.

[4] J. Moreno-León and G. Robles, “Dr. scratch: a web tool to automatically evaluate scratch projects,” in *Workshop in Primary and Secondary Computing Education*, 2015, pp. 132–133.