Application of partial credit models in testing performance assessments for programming course

N M S Mertasari¹ and I M Candiasa²

¹,²Mathematics and Science Faculty, Universitas Pendidikan Ganesha, Singaraja - Bali – Indonesia.

Abstract. The assessment of programming skills must be able to measure holistically the theoretical abilities and programming skills. Performance assessment is seen as an appropriate assessment to measure programming ability because it is able to evaluate authentic ability through the process carried out or the product produced. The problem that often arises is performance assessment testing to measure programming capabilities. On this occasion, the Partial Credit Model (PCM) was tried to test performance assessments to measure programming capabilities. PCM is a development of the Rasch model, which applies one parameter, namely the item difficulty index. If the Rasch model is applied to dichotomous items, then PCM is applied to polotomic items. PCM assumes that all items have the same difference power. The category score on the PCM shows the per-step score for correctly completing the item according to the scoring rubric developed. The probability of each test taker is estimated by calculating the probability of answering each step in completing an item. There were five items that were tested in two parallel classes in relatively different times. The test results showed a difference in probability in the two classes, but the difference was not too far away. So, the performance assessment was precise enough to measure programming capability, and the test results were quite precise when tested with PCM.

1. Introduction

The purpose of learning computer programming is to develop the skills needed by students to become professional software developers [1]. Software or computer programs are generated through a programming process. Programming is seen as giving a series of commands in one unit so that the computer does something [2]. The series of commands given to the computer is called a program. Computer programs are made to bridge communication between computers and their users because computers understand machine language while computer users on the other hand understand their own language. Various components must be considered, so that a computer program can ensure that computer users are able to communicate effectively with the computer. Therefore, many people consider programming as an art other than a science [3].

Programming stages must be followed properly. The programming process goes through four phases, namely problem analysis, design, coding, and validation [4][5]. The problem analysis phase is the stage of identifying the input that must be given to the computer and the expected results from the computer. The design phase is the preparation of a program design, namely the input processing process in order to obtain the desired results using everyday language. In the coding phase, the program design is translated into a program using a programming language. Finally, in the validation phase the program was tested with limited data.
Various abilities are involved in programming learning. The problem analysis phase involves a defined concept in the form of a classification of known components and components to be searched for in the problem. From the design phase, the skills involved are high-level rules and regulations. High-level rules and regulations are structured into a program design, namely the relationship between known components and components that are sought in semantic form in everyday language.

Entering the coding phase, rules in the form of semantics created in the design phase are translated into syntax rules by following the grammar of the selected programming language. In the coding phase, the most widely used capabilities are classification, rules and high-level rules. In order to express semantic rules into syntax rules, the correct command must be chosen along with the correct way of using it. Accuracy in selecting commands and arranging them into syntax rules plays a very important role because syntax rules are very strict, one command only has one meaning. Finally, in the validation phase, skills in the form of rules are also needed. The rules in the phase are traced to see if the results found by the program are correct or incorrect. If the results are still incorrect, then a reverse rule search is carried out to find errors and then corrected them.

Learning programming phase is expected to result in a transfer of learning. In addition to being able to apply programming skills to new problems, students are also expected to be able to use different programming languages or program in different hardware environments. The main goal is to anticipate the rapid development of computers, both hardware and software. Therefore, the theoretical abilities and skills possessed by students must be equally adequate. If students only have theoretical skills without adequate programming skills, then students are only able to theorize but are not able to produce good programs. Conversely, if students are skilled at making programs but are not supported by adequate theoretical abilities, the resulting program will be monotonous, without innovation and without creativity.

Learning programming material requires a typical test form to measure student learning progress. Computer programming is nothing more than problem solving [6]. Performance tests are seen as a suitable form of test for assessing programming skills. The reason is that performance tests can measure the quality of work completed, skills and accuracy in operations, speed and ability to plan work, or component identification [7]. Lane emphasized that performance tests are suitable for assessing higher-order complex thinking skills. Computer programming involves a lot of complex thinking skills [8].

Performance tests are designed to measure specific performance [9], so they are also referred to as authentic tests because they involve performance that is rewarded according to students' own rights [10]. Performance tests include observation of a workable behavior or an evaluation of the product of a behavior [11]. Performance tests emphasize the similarity between the observed performance and the actual criteria situation [12]. Programming is a special skill whose quality is tested by the quality of the program or software produced [13]. Therefore, programming ability is assessed by giving assignments in the form of problems that must be programmed by students. Products produced by students in the form of computer programs are assessed using an assessment rubric arranged based on the limitations of the quality of the computer program.

Meyer divides the quality of the program into two factors, namely external quality factors and internal quality factors [14]. External quality factors related to program use include truthfulness, robustness, extendability, reusability, efficiency, portability, verification, integrity, and ease of use. Internal quality factors include modularity and readability. Cavano and McCall divide program quality into three parts, namely operation, shutdown, and program improvement [15]. Program operation includes components of truth, reliability, efficiency, integrity, and usability. Program rotation includes components of reusability, portability and interoperability. Program improvement includes components of testability, maintainability, and flexibility.

Ghezzi and colleagues classify several components of software quality, including truth, reliability, robustness, performance, friendliness, maintainability, reliability, reusability, interoperability, portability, understandability, timeliness, productivity, and visibility [2]. Based on the three formulations of the quality of the computer program that have been described, in this study the determination of the quality of the program is based on the following indicators: (1) truth, which shows
the accuracy of the program in carrying out its duties, according to specifications; (2) readability, which indicates the level of readability of the program, especially by people other than the programmer; and (3) friendliness, which states the ease of operation of the program and the program's ability to cope with abnormal situations.

2. Method

The performance test to measure programming ability consists of five items. The test was tested on 88 students who were divided into three classes. The test was tested once in three different classes. The trial results were analyzed using the Partial Credit Model (PCM). PCM is a model for a stepwise solution of polimically scored items, where item parameters are interpreted as the difficulty of the steps [16]. PCM aims to describe students' competencies on a continuum from basic competences to complex competency levels [17]. Similar to the Rasch model, PCM assumes that all items have the same difference in power. The item difficulty level parameter is the only item characteristic that is considered to affect student performance. Thus, the chances of students succeeding in working on the item depends on the student's ability and the difficulty level of the item. The sequence of steps for completing the test items is not required in PCM.

The many steps to correctly complete a single item represent a category score. A larger score indicates higher ability, whereas a smaller category score indicates lower ability. If \( i \) is a performance test item that has a difficulty index \( \delta \) with a category score of \( x \) of 0,1,2,…, \( m_i \), then the probability of individual response with the ability level of obtaining a category score on item \( i \) is stated by the following equation [18].

\[
P_{nx} = \frac{\exp(\theta_n - \delta_{ij})}{1 + \exp(\theta_n - \delta_{ij})}, \text{ for } x = 1,2,3,\ldots, m_i
\]

(1)

The difficulty index (\( \delta \)) is calculated by the formula:

\[
\delta = \frac{\sum U + \sum L - (2N_s S_{\text{max}})}{2N(S_{\text{max}} - S_{\text{min}})}
\]

(2)

In the above formula: \( \Sigma U \) = total score of the upper group;
\( \Sigma L \) = total score for the lower group;
\( S_{\text{max}} \) = maximum score of items;
\( S_{\text{min}} \) = Minimum score of items;
\( N \) = many test takers.

3. Result and Discussion

Five items of performance tests to measure programming ability were tested against three different classes at different times. The difficulty index is calculated from the test results. In the assumption of the participant's ability of 0.5, there is a chance to answer correctly (\( P_{nx} (\theta) \)) for the five test items for testing in the three classes as listed in the following table.

| Item Number | Chances of Answering Correctly (\( P_{nx} (\theta) \)) |
|-------------|----------------------------------|
|             | Trials in the First Class | Trials in the Second Class | Trials in the Third Class |
| 1           | 0.433                         | 0.458                       | 0.410                      |
The results of trials in three different parallel classes show that the chances of answering each item correctly are inconsistent. The inconsistencies in the magnitude of the odds vary widely. The inconsistency of the odds on items number 1, number 2, and number 5 does not exceed 0.05. This means that these items are still relevant for use. Meanwhile, for item number 3 the inconsistency of the opportunity value has exceeded 0.05. Therefore, it is necessary to revise these items. Finally for item number 4 the inconsistency of the probability value has exceeded 0.1. This condition indicates that for item number 4 a serious revision needs to be done, even if it is deemed necessary item number 4 must be aborted.

In general, the measurement of programming learning outcomes using performance tests provides accurate results. This happens because the performance test is able to measure the authentic ability of students based on the performance shown. The real condition of students presented in their performance can be measured carefully using a rubric. The assessment rubric that was developed gave fairly consistent results. The criteria for truth, readability, efficiency, reliability, and friendliness used to assess the resulting program are sufficient to capture the ability to program computers. Testing results on certain items (for example item number 4) can occur due to ambiguous statements on items that can confuse students.

4. Conclusion

Programming is a skill that is needed in the future. Therefore, programming capabilities must be well developed. Programming learning must be carried out effectively in order to be able to provide adequate programming skills to students. In programming learning, an effective learning outcome measurement tool is needed to monitor student learning progress, so that the quality of the learning process can be improved slowly. The performance test is an effective measuring tool in programming learning because it is able to measure theoretical and practical abilities in an integrated manner in the form of product assessment. The student-generated program is the product of a performance test in learning programming. An effective rubric for assessing the program is able to provide a relatively clear picture of students' ability in programming.

As for other tests, performance tests to measure programming capabilities need to be tested. PCM is an appropriate analytical technique for analyzing trial results of performance tests for programming. PCM considers the students' abilities per category in solving programming problems. The performance test rubric component for programming can be used as an effective category in PCM. Even though the different power is assumed to be the same, with the variable level of difficulty and the effective assessment category, PCM is able to provide analysis results in the form of opportunities to answer the items correctly.

References
[1] Saeli M, Perrenet J, Jochems W M G, and Zwaneveld B 2011 Teaching Programming in Secondary School: A Pedagogical Content Knowledge Perspective Informatics in Education 10

| Item Number | Chances of Answering Correctly (P_nix (θ)) |
|-------------|------------------------------------------|
|             | Trials in the First Class | Trials in the Second Class | Trials in the Third Class |
| 2           | 0.358 | 0.342 | 0.377 |
| 3           | 0.322 | 0.368 | 0.298 |
| 4           | 0.088 | 0.218 | 0.192 |
| 5           | 0.202 | 0.192 | 0.233 |
[2] Ghezzi C, Jazayeri M, and Mandrioli D 1991 *Fundamentals of Software Engineering* (Englewood Cliffs: Prentice-Hall International Inc.)

[3] Horton J 2015 *Android Programming for Beginners* (Birmingham UK: Packt Publishing Ltd)

[4] Clark C B 1995 Cognitive Style and it’s Effect on the Stages of Programming *Journal of Research on Computing in Education* 27

[5] Jalote P 1991 *An Integrated Approach to Software Engineering* (New York: Springer Verlag)

[6] Wallace W 1991 *Beginning Programming All-In-One Desk Reference For Dummies* (Indianapolis Indiana: Wiley Publishing Inc.)

[7] Denova C C 1979 *Test Construction for Training Evaluation* (New York: Van Nostrand Reinhold Company)

[8] Lane S 2010 *Performance assessment: The state of the art (SCOPE Student Performance Assessment Series)* Stanford CA: Stanford University Stanford Center for Opportunity Policy in Education

[9] Brown J D 2004 Performance assessment: Existing literature and directions for research *Second Language Studies* 22 (2)

[10] Linn R L, Baker E L, and Dunbar S B C 1991 Performance-based assessment: Expectations and Validation Criteria, *Educational Researcher* 20 15-21

[11] Callahan J F and Clark L H 1977 *Teaching in the Middle and Secondary Schools* (New York: Macmillan Publishing Co. Inc.)

[12] Palm T 2008 Performance Assessment and Authentic Assessment: A Conceptual Analysis of the Literature, *Practical Assessment, Research, and Evaluation* 13 Article 4

[13] Abass Q A 2006 *Introduction to Computer and BASIC Programming Concepts* Ijebu-Ode (Ogun State: Over to God Press)

[14] Meyer B 1988 *Object-Oriented Software Construction* (New York: Prentice Hall)

[15] Cavano J P and McCall J A 1991 A Framework for The Measurement of Software Quality cited by Pankaj Jalote *An Integrated Approach to Software Engineering* (New York: Springer Verlag)

[16] Verhelst N D and Verstralen H H F M 2008 Some Considerations on the Partial Credit Model, *Psicologica* 29 229-254

[17] Eggert S and Geholz S B 2009 Students’ Use of Decision-Making Strategies With Regard to Socioscientific Issues: An Application of the Rasch Partial Credit Model *Wiley InterScience* www.interscience.wiley.com DOI 10.1002/sce.20358

[18] Master G N 1988 The Analysis of Partial Credit Model *Applied Measurement in Education* 1 279-297