Effective Assessment of Written-Based Computer Programming Codes: Using a Grid-Based Approach

Canayah Cuniah[1]

ABSTRACT

Various studies have noted challenges faced by novice computer programming learners in upper secondary schools. Several researchers suggest different modes of assessments apart from the existing teaching strategies authors to strengthen both feedback to students and improve teaching of computer programming. This paper suggests the use of a pre-formulated, programming grid as an assessment tool, to record and analyse programming errors from students' written-based programming solutions. It reports the empirical findings of a study that used the grid and analysed a total of ninety scripts corresponding to three problem-solving questions answered by a batch of thirty students (17-19 ages) from an upper secondary school in Mauritius. The results not only identify the exact errors made by students but also reveal conceptual difficulties. The grid-based approach helps identify learning and conceptual difficulties and thereby revisit teaching strategies at the individual level and collectively as a class. The conclusion notes the relevance of the grid based approach as an easily accessible assessment in schools to improve computer programming education. The grid is part of a work in progress for a doctoral research.

Keywords: Computer Programming, Programming Grid, Written-Based Programming Codes, Conceptual Difficulties.

Introduction

Computer science education helps students become competent, confident skilled IT users of technology. Various researches carried out globally shows the difficulties of learning computer programming [1], [11]. In Mauritius computer science education has been promoted by the government. Computer science has been newly incorporated into the syllabus by Cambridge International Examinations. It has replaced computer studies and computing subjects. Still, the yearly decline in the performance at the national level is a deep cause for concern. Cambridge examiners reports state that students in Mauritius are weak in problem-solving abilities and unsure about the programming language they use [7]. However, the difficulties of computer science education have not been studied except recently [5]. There is a need to appraise both learning and teaching methodologies. It can begin only with substantial tools that guide and support both teachers and students in the classroom during the learning process.
In Mauritius, current curricular instructions/examination mandates written examination. End-term grades may indicate learning. However, assessments are important to clearly understand students' progress, identify errors, track strength and weakness in computer programming knowledge and continuously mould teaching and learning. This paper proposes the use of a programming grid formulated as a pedagogical framework and tool to identify students' learning difficulties which can be used at any stage of learning. Tools that monitor learners' progress both individually and collectively can assist in improving teaching and can be widely beneficial in computer programming education.

2.0 Literature Review

2.1 Computer Programming Assessments

Learning computer programming is through the process of problem-solving, designing and thinking, which appear as a nuisance to learners [10]. It involves daunting skills of memorising and applying ambiguous programming concepts that demand strenuous cognitive judgment and high level of abstraction [5]. Programming issues like designing a program, breaking down problem into sub modules, and debugging the program for relevancy affects learning process. Authors suggest the need to remedy the difficulties of understanding even the most basic concepts of programming [6]. Concepts like recursion, abstract data types, error handling and language libraries are perceived to be

[1] University of Technology, Mauritius La Tour Koenig, Pointe aux Sables, Mauritius

harder to memorize and apply [2]. Students, therefore, rely on the practical sessions done at school to acquire programming skills, which necessitates regular individual attention.

There are various questions and issues raised in studies related to the teaching and assessing of programming. Learning capacities vary from countries, according to schools, infrastructure, urban-rural locations, student capabilities their socio-economic background and even pedagogical approaches [3]. There is "*no consensus on what the programming process is, much less on how it should be taught*" despite it being around for 50 years [8]. The general consensus in literature on computer programming education is that *'static* teaching materials such as textbooks, lecture notes, blackboards or slide presentations' as insufficient to teach the '*dynamic process'* of developing programs [4]. Some studies are country specific like the teaching of specific topic informatics (particularly programming) as part of computer science education in upper secondary schools in Netherland [10]. Others examine prevailing approaches like the use of online materials, model-driven approaches and the salience of feedback in teaching programming including technology based assessments, exams, project-based feedback and assessment [9]. Several modes are used to assess computer programming learning.

2.2 The Programming Grid

Though it has been found that traditional method which is written-based no longer meets the demands for the expected learning outcomes, [12], this method continues to prevail as a means to test the students' level of understanding in computer programming. In Mauritius where written-based tests are popular, there is a need to develop and use assessment tool that can guide learning and teaching. The complexity of the programming concepts and assessing written-based programming scripts can have an influential impact in the teaching strategies reformulation. A programming grid, therefore, was designed as an endeavour to help educators determining the strength and weakness of students' programming abilities. The Programming grid is an initial attempt in assessing written-based programming scripts. Programming concepts elaborated in the computer science for upper secondary were classified alongside common mistakes and referenced by using customised codes to accommodate diverse concepts of programming. Each programming concept and associated mistakes were categorised and coded into a tabular format. The results of an earlier study conducted by the researcher to note errors also helped formulate the grid.
Programming questions require different level of cognitive abilities. Students are expected to use and apply all the computer programming knowledge based on the lessons that have been taught. However, as the questions demand critical and analytical responses, students fail to use their creative thinking abilities to transform abstract problems to concrete solutions. Therefore, the programming grid as an assessment tool of written-based scripts acts as a conceptual framework to identify in which programming concepts students are recurrently facing setbacks. The grid allows potting the mistakes found in the students' programming scripts during script correction. This can serve to assist the teacher to also modify or improve his pedagogical strategy and content. It can help identify individual difficulties and also collectively identify the problematic areas that need to be further taught. This was tested and the following methodology was used.

3.0 Methodology

Data was collected from the programming scripts of thirty students aged from 16-19 years who are in the final year of the upper secondary in an urban school in Mauritius. Three problem-solving questions were set and each student had to produce written-based programming codes using VB.Net programming language, one of the technical languages recommended by the Cambridge examiners. Each section of the questions was carefully designed based on the curriculum and it was intended to assess the level of understanding and application of the programming concepts. Question 1 is a simple context-based problem supposed to test the students' computer programming basics, to solve mathematical exercise. Question 2 required the use and application of conditional and recursive concepts to write coded solutions for a more complex problem. Finally the third question required the application of arrays which includes memorization and use of all concepts taught.
There were a total of ninety answer scripts which was analysed using the gird by the researcher to generate data. The students are understood to have attended the programming classes and believed to have the similar exposure to knowledge according to the syllabus. Since this research is a work in progress no attempt has been made to survey the educators' or students' perceptions.

3.2 Analysis

The analysis is focused collectively and not individually. Areas of programming difficulties faced by the students are shown in Figure1. This illustration helps in understanding the difficult areas/concepts in which the most number of errors have been recorded form the programming grid. As a pedagogical tool the grid helps in identify the difficult concepts of programming and also where the students have trouble in memorizing and applying the programming concepts while writing the programming statements manually.

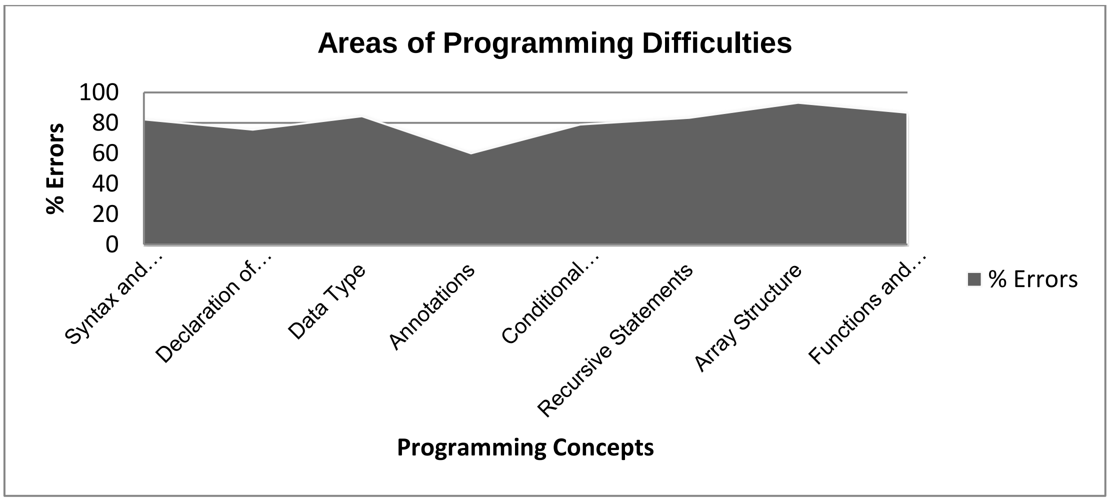

Figure 1: Percentage Errors

A classification of errors based on the parameters in the grid was further analysed. While most educators mark errors in written-based, there is a need to identify what are the errors and where does programming difficulties of the students lie. The record not only benefits the educators to sharpen their pedagogical efforts but also convey and correct the students as and when mistakes occur.
Figure 2 illustrates the numbers of errors found in each programming concepts for all the three questions alongside the number of correct answers. It is noted that of all the 90 scripts there were 82.2% errors in syntax and semantics, 75.6% no declaration of variables, wrong use of data type 84.4%, wrong placement of the if and else keywords 78.9%. The loop constructs for the recursive statements were incorrectly implemented, the irrelevant use of parameters for functions and procedures and finally the array structure is by far very complex for the students to master as it can be seen that many had difficulties in applying the concepts while writing the programming codes.
Compared to the others students fare slightly better in annotations. On the other hand there are scripts which have very less number of errors but they are very few. The grid hence provides an effective tool that indicates problematic areas and helps to assess and comprehend the students' level of understanding in computer programming concepts.

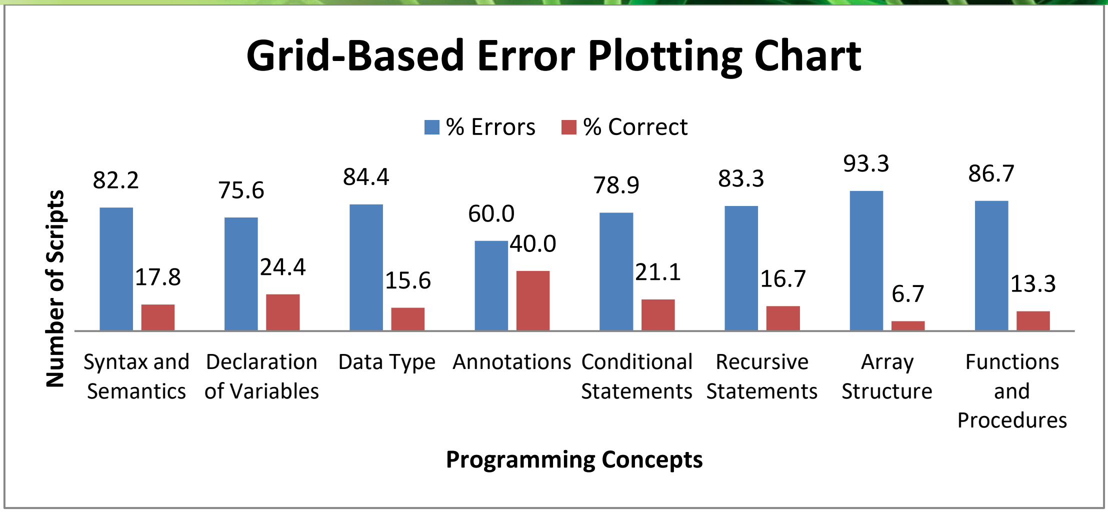

Figure 2: Grid-based Errors Plotting Chart

4.0 Conclusion

The programming grid is thus designed to support and improve the pedagogy of computer programming. From the grid it is found what programming concepts students have difficulties and the number of logical and syntactic mistakes. As a pedagogical tool it is capable of visualising the students' common programming mistakes both individually or combined. The difficult areas are systematically identified and therefore pedagogical means and strategies can be further amended or developed by the educators as an endeavour to alleviate programming learning difficulties.
Therefore, in view of improving the pedagogical content framework for computer programming, the use of the programming grid helps in learning progress of the learners and assess where the students stand post teaching sessions. Its conceptual framework proposes solutions to the teaching and learning of computer programming. Knowledge of conceptual errors and problem-solving can help learners and teachers to better grasp difficulties. A research with a larger population is being conducted to understand the programming grid better. Future studies intent to further test the grid with teachers and students to make it reliable and valid.

References

[1] Altadmri, A. & Brown, N. C. C., 2015. 37 Million Compilations: Investigating Novice Programming Mistakes in Large-Scale Student Data. kansas City, MO, USA, IGCSE'15.

[2] Altun, A. & Mazman, S. G., 2015. Identifying latent patterns in undergraduate Students' programming profiles. *Springer Open Journal,* 2(13), pp. 1-16.

[3] Barry J, 2006. *The Effect of socio-economic status on academic achievement,* Kansas: Wichita State University.

[4] Bennedsen, J., Caspersen, M. E. & Kölling, M., 2008. *Reflections on the Teaching of Programming.* 1st ed. Berlin: Springer-Verlag Berlin Heidelberg.

[5] Cuniah, C. & Panchoo, S., 2015. *Computer Programming Difficulties at Higher Secondary School: A Content Analysis.* Dubai, Fifth International Conference on Industrial Engineering and Operations Management (IEOM 2015).

[6] Esteves, M., Fronseca, B., Leonel, M. & Martins, P., 2008. *Contextualization of Programming Learning: A Virtual Environment Study.* New York, 38th ASEE/IEEE Frontiers in Education Conference, pp. 17-22.

[7] Examiners, 2015. *Principal Examiner Report for Teachers,* Cambridge: Cambridge International Advanced Subsidiary Level and Advanced Level.

[8] Gries, D., 2008. Foreword. In: *Reflections on the Teaching of Programming: Methods and Implementations.*. Berlin: Springer, p. V.

[9] Kölling, M., 2008. Introduction to Part IV Assessment. In: *Reflections on the Teaching of Programming: Methods and Implementations.* Berlin: Springer, p. 209.

[10] Saeli, M., Perrenet, J., Jochems, W. M. G. & Zwaneveld, B., 2011. Teaching Programming in Secondary School: A Pedagogical Content Knowledge Perspective. *Informatics in Education,* 10(1), pp. 73-88.

[11] Sarkar, A., 2015. *Confidence, command, complexity: metamodels for structured interaction with machine intelligence.* Bournemouth, Psychology of Programming Interest Group.

[12] Wang, Y. et al., 2012. Assessment of programming language learning based on peer code review model: Implementation and experience report. *Elsevier,* Volume 59, pp. 412-422.