Current Practices of Dynamic-Structural Testing in Programming Assessments

Rohaida Romli, Emylda Arni Abdurahim, Musyrifah Mahmod, Mazni Omar
School of Computing, College of Arts and Sciences, Universiti Utara Malaysia, 06010 UUM Sintok, Kedah, Malaysia.
aida@uum.edu.my

***Abstract*— Automatic Programming Assessment (or APA) has been known as an important method to automatically mark and grade students' programming exercises. It has been gaining a lot of attention from many researchers either to emphasize on the aspect of static analysis or dynamic testing (functional and structural testing). To date, not many recent studies attempted to focus on the context of structural testing even though, it is key in the software testing industry. Hence it becomes one of the most critical aspects of testing to be considered. Besides that, current literatures also lack information on APA's detailed practices. Thus, we conducted a preliminary study to investigate the test adequacy criteria that have been commonly employed in the current practices of programming assessments which are applicable only to dynamic-structural testing. Specifically, this refers to testing that needs a program execution and focuses on the logic coverage of the tested program. In this paper, we reveal the means of conducting the preliminary study and its analysis and findings. From the findings, it has been discovered that most educators are commonly adopting the identified structural code coverage in programming assessments and even have a great leaning towards allowing those criteria to be considered in implementing APA.**

***Index Terms*— Automatic programming assessment; Structural testing; Structural code coverage.**

I. INTRODUCTION

Programming assignments and problems are considered as important elements in software engineering and computer science disciplines. Programming assignments contribute as a means of exposing students and getting them familiar with programming languages as well as allowing them to practise programming fundamentals and concepts effectively. Programming assessment tasks are commonly placed on educators or instructors and other resources so as to assess the level of correctness of programming assignments. The principles and techniques of software testing will be utilized to judge the quality level of each programming assignment.

The huge number of students in a single class results in a big number of programming assignments or exercises. Thus, educators or instructors need extra time to manage these programming assessments. Besides that, feedback provided to students through marking is commonly limited, and often late and outdated, particularly to the topic dealt with in the assignment [1]. Therefore, Automatic Programming Assessment (APA) would overcome such problems by providing students with assessment results immediately after submitting their programming assignments or exercises.

Nowadays, most educators have encountered that activities dealing with assessing students' programming assignments are burdensome and significantly increase their current workloads. Therefore, APA has attracted more attention from researchers in the field of teaching and learning programming [2]. APA is typically based on testing techniques [3], and requires a test data generation process to perform a dynamic testing on students' programs [4]. Dynamic testing involves the execution of a program with test data and the comparison of the results with the expected output, which must satisfy the users' requirements [5]. The correctness, execution efficiency and testing ability of students can be automatically and effectively assessed by using dynamic testing [2]. In addition, existing studies, particularly in the area of programming assessments, still have only limited discussions on current practices in conducting the assessments [6]. Thus, this study attempts to investigate the current practices of dynamic-structural testing in programming assessments. Specifically, this study mainly seeks to identify the test adequacy criteria used for dynamic-structural testing and to verify the identified criteria in the context of current practices in programming assessments. Hence, this paper discusses the preliminary study that was conducted to gauge the required details.

The content of the remaining sections are organized as follows: Section 2 details the code coverage that are commonly employed in dynamic-structural testing. Section 3 provides details of the survey conducted for the preliminary study. Section 4 reveals the analysis and findings of the study. Finally, Section 5 concludes the paper.

II. CODE COVERAGE METRICS FOR DYNAMIC-STRUCTURAL TESTING

Software testing is an important technique to measure the quality of software product assurance [7]. The two important goals of software testing are to ensure the system being developed is according to the customers' requirements and also to reveal bugs [8]. The establishment of good testing skills must begin as early as possible in the computing curricula [9]. According to Zhu[10], the central problem of software testing is "What is a test data adequacy criterion?", which can be defined as the rules that are needed in order to determine whether a software has been tested sufficiently or not.

Software testing is commonly categorized into two parts: static testing and dynamic testing [11]. Dynamic testing falls

into two parts that are functional testing (black-box testing) and structural testing (white-box testing) [12][13]. Functional testing emphasizes inputs, outputs and principle functions of a software module [14]. Meanwhile, structural testing is a method of testing that depends on the internal structure of software applications [11]. Structural testing is the most common form of assessment to determine the coverage of the program logic or so-called coverage metrics [15]. General classifications of coverage metrics include [10][11][15][16]:

1) Statement coverage: this type of coverage needs each statement in a program to have been executed and implemented at least once.
2) Path coverage: path coverage depends on a program source code to find every way possible for each program through which it passes or executes all the possible paths.
3) Branch coverage: it requires all branches and decisions, which must be taken in a program to be passed at least once.
4) Condition coverage: condition coverage is evaluating each condition as true and as false at least once
5) Multiple condition coverage: this type of coverage reports all completed combinations of other coverage such as branch coverage, condition coverage, decision coverage and statement coverage.
6) Modified Condition/Decision Coverage (MC/DC): the decision has taken all possible outcomes at least once and, it is said that both the true and the false branches have been covered.
7) Loop coverage: loop coverage considers each loop in the control flow program will be executed in zero time, just once, or more than once in a row.

III. The Survey

In this section, a discussion on the research design of a survey conducted for the preliminary study, its respondents and the survey instruments used are detailed out. The conducted preliminary study aims to investigate the current practices of dynamic-structural testing in programming assessments. The specific objectives include:

1) to identify the test adequacy criteria used for dynamic-structural testing
2) to verify the identified criteria in the context of programming assessments current practices as well as for a future consideration if APA is implemented.

The respondents of the survey were educators who have been teaching programming courses at one of the public universities in Malaysia. The respondents were selected on the basis of their expertise in the subject investigated. Due to the time constraint and the fact that the survey required minimal interference from researchers, (particularly to understand the identified structural code coverage) only one university was selected. The survey received a total of thirteen responses.

A questionnaire was designed to collect the related data and information. The close-ended questions ask the respondents to make choices among a set of alternatives given by the researchers. The investigated structural code coverage in programming assessments were based on the information collected from literature survey.

The questionnaire consisted of thirteen questions that are divided into three parts: background (demographic of respondents), the adoption of structural code coverage in programming assessments and the future consideration for the structural code coverage in implementing APA. The questions that involved ratings used the Likert Scale format. We used four types of estimations for the Likert Scale, ranging from 1 to 5 and 1 to 4. The first type was frequency estimations which consisted of five values; *almost never*, *some of the time, about half of the time, most of the time* and *almost always*. The second type was priority estimations, which used the scale: *not a priority*, *low priority*, *moderate priority*, *high priority*, and *essential*. The third type was agree or disagree estimation which used the scale; *strongly disagree*, *somewhat disagree, neither agree nor disagree*, *somewhat agree* and *strongly disagree*. The fourth type was critical estimation which used the four values; *not critical, low critical, moderately critical* and *high critical.*

Figure 1 shows the design of the preliminary study. Since the targeted respondents were among the lecturers who have been teaching programming courses at higher learning institutions and were categorized as the specific target groups, its sampling design was based on a purposive sampling (non-probability) technique. The unit of analysis was individual response, as the study treats each lecturer's response as individual data source. This study employed the study setting known as field study that is in non-contrived settings. This means that the preliminary study was done in the natural environment where work proceeded normally and the factors to be studied were not controlled [17]. In this situation, the study requires minimal interference by the researcher. This study identified the time horizon as a cross-sectional study because the data were collected only once and no other consecutive data collection activities will be carried out. In terms of the measurement, as stated earlier, the collected data based on a survey using the questionnaire, the constructs and items were measured using scale (itemized rating scale). In order to achieve the identified objectives of the study, descriptive statistics were used as the data analysis technique. The statistical data derived from this preliminary study were analysed based on Descriptive Statistics – Frequencies (graphing frequencies) by using Microsoft Office Excel 2010.

IV. Findings And Discussion

The following sub-section discusses the analysis and findings of the conducted preliminary study that are based on: demography of respondents, adoption of structural code coverage in the current practices of programming assessments and future considerations for automated programming assessment.

A. Demography of Respondents

The demography of respondents consisted of five questions: level of appointment, experience in teaching programming courses, type of programming language applied in teaching a programming course, programming course(s) that have been taught, and the current means of marking students' programming exercises. Figure 2 shows the frequency of responses for each question.

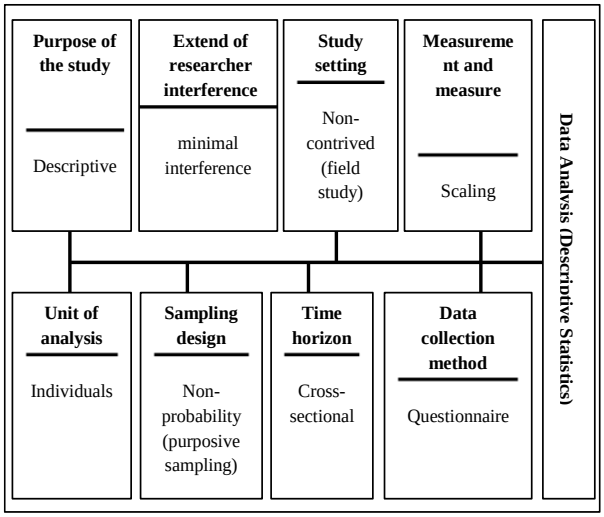

Figure 1**:** Design of the preliminary study

From the result tabulated in Figure 2, it can be seen that most of the educators or instructors in the university are lecturers (that is about 77%) and merely 23% of them are senior lecturers. This might due to the fact that programming courses commonly require a lot of effort devoted to ensuring students are able to understand very well all the concepts and principles of programming, which will be the basis for higher level courses. Hence, the younger generation of lecturers seemed likely to be more passionate in dealing with this kind of circumstances. It is also shown that most of the educators have more than three years of specific experience in teaching programming courses, and Java has become the most popular programming language applied in teaching the courses. Almost 50% of the educators are specifically focused on teaching an introductory programming course as compared to data structure and advanced programming courses. In terms of the current means of marking students' programming exercises, about 72% of them appeared to be manually marking printed documents rather than manually marking via the softcopy version of programming solutions (that is about 28%).

B. The adoption of structural code coverage in programming assessments

This sub-section reveals the findings in terms of the frequency of adopting structural code coverage in the current practices of programming assessments, level of prioritization and scoring of each code coverage metrics, and overall scoring in structural testing. This part of the investigation aims to achieve the first objective of this study.

As mentioned earlier, among the structural code coverage metrics considered in this study include statement coverage, path coverage, branch coverage, condition coverage, multiple condition coverage, MC/DC and loop coverage. As shown in Figure 3, in terms of the adoption of structural code coverage in the current practices of programming assessments, the highes response is narrowed down to the frequency of *most of the time*, particularly for the path coverage (about 46%), condition coverage (about 39%), loop coverage (about 39%) and branch coverage (about 31%). In general, a very small number of the educators responded with a frequency of *almost never* for the considered structural code coverage metrics, except for the MC/DC. This might be because they were not

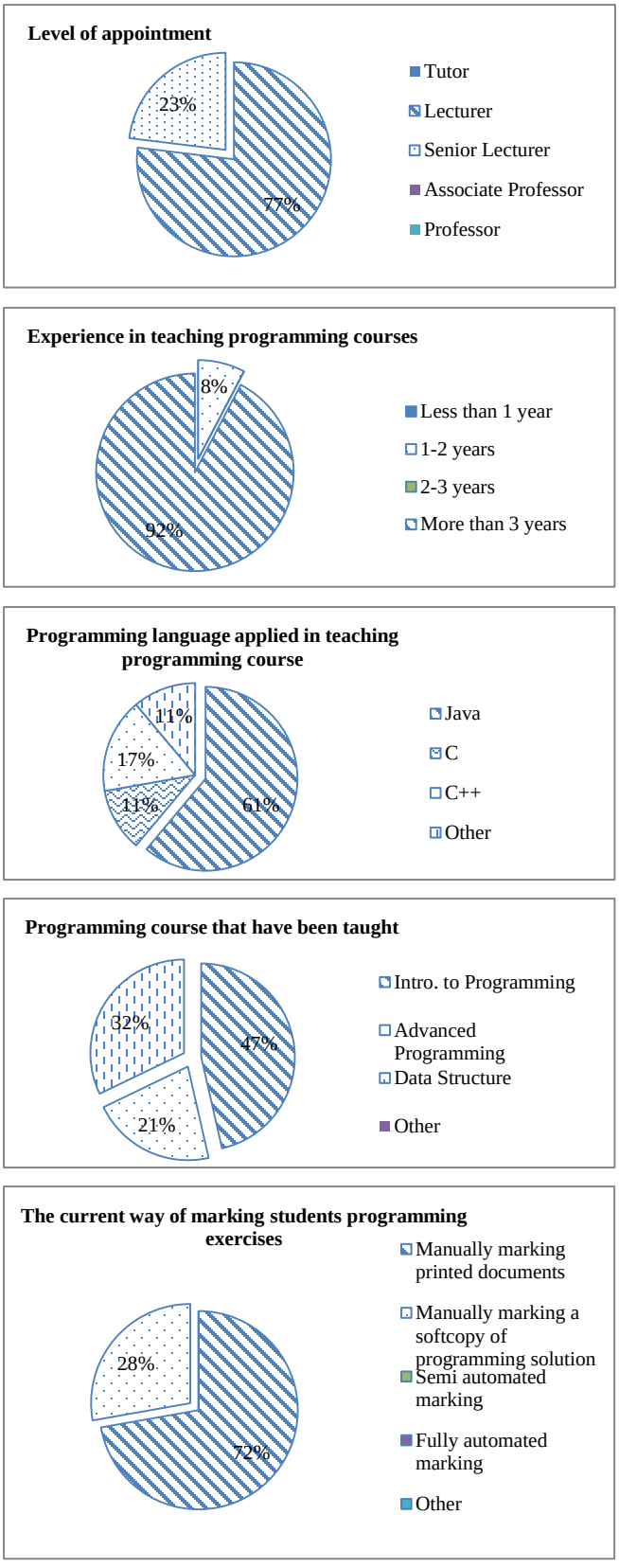

Figure 2: Demography of respondents

really familiar with that type of code coverage metrics. Overall, it can be concluded that most of the educators quite often consider structural code coverage in programming assessments.

Figure 4 shows the findings in terms of to what extent educators prioritize each of the structural code coverage metrics in programming assessments. The findings indicate that path coverage and condition coverage are among the essential coverage metrics applied in programming assessments as compared to other coverage metrics. If we put in a ranking on the prioritization level of the structural code coverage metrics, the sequence will be (1) path coverage, (2) condition coverage, (3) loop coverage, (4) statement coverage, (5) branch coverage, (6) multiple coverage, and (7) MC/DC. As shown in Figure 4, it seems likely MC/DC is one of the structural code coverage metrics that is anomalous among the educators.

Figure 5 depicts the findings in terms of the current means of scoring each of the structural code coverage metrics in programming assessments. The score was given as a range of values from 0 to 100. The score with the highest frequency is 50 marks, with path coverage at about 39% and around 29% for multiple condition coverage, branch coverage, condition coverage and loop coverage. It was also found that around 29% of educators rated the score of 60 marks for both of the branch and condition coverage types and 31% particularly for loop coverage. Overall, on average, the structural code coverage criteria contribute 60 marks or lower to the 100 marks allocated for students' programming exercises.

In terms of the overall scoring for structural testing, Figure 6 reveals the findings. The highest number of respondents (that is about 31%) provides an overall scoring of 50 marks for structural testing. Around 24% of them seemed to score 70 marks and the remaining respondents appeared to score marks lower than 50. As a conclusion, it can be said that the preferred total score allocated for structural testing is 50 marks or less.

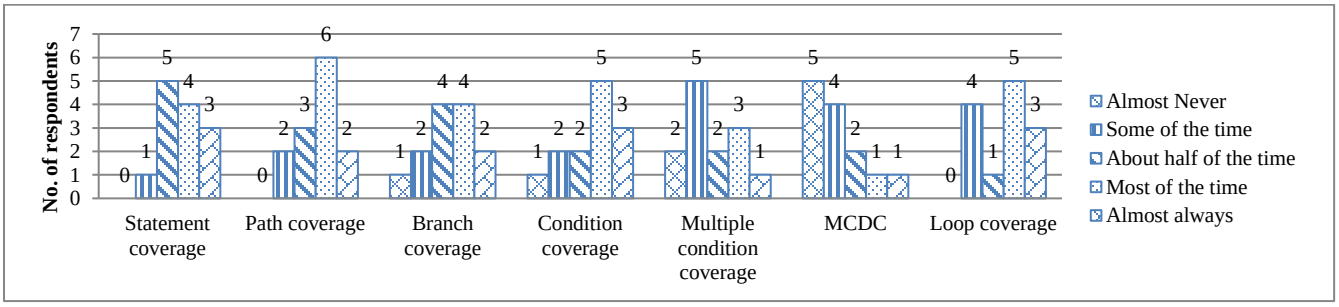

Figure 3: Frequency of adopting the structural code coverage in programming assessments

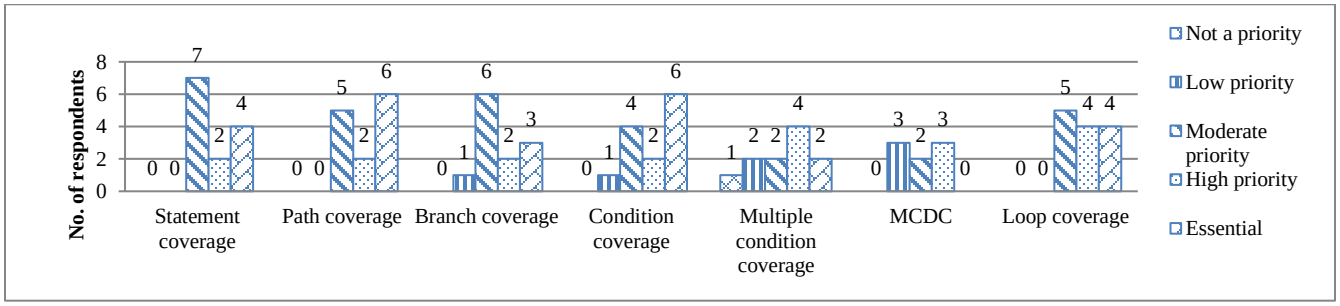

Figure 4: Level of prioritization the structural code coverage in programming assessments

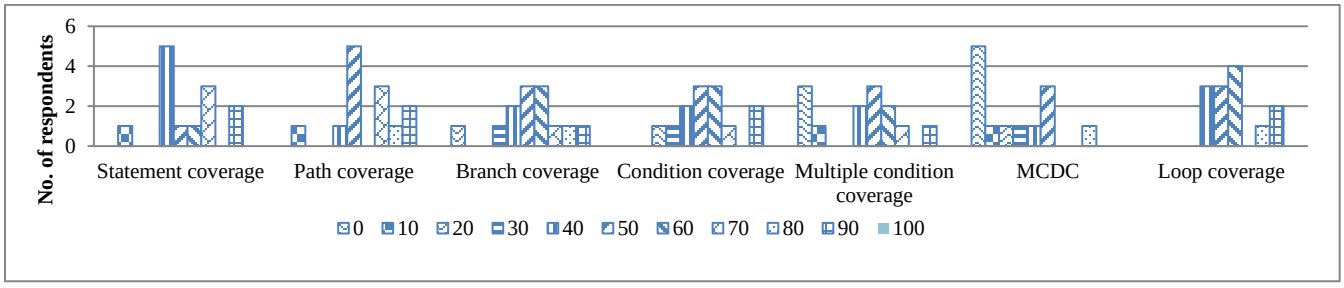

Figure 5: The current means of scoring the structural code coverage in programming assessments

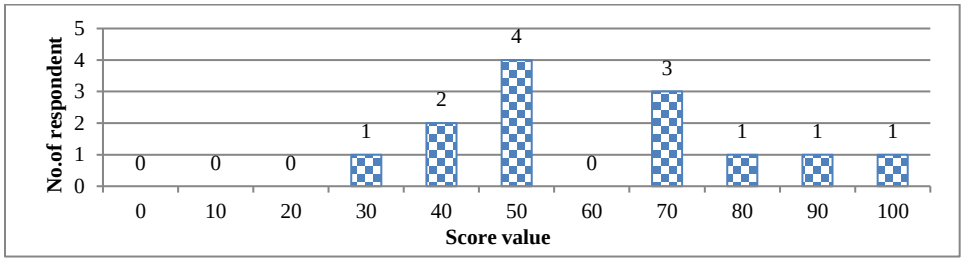

Figure 6: Overall scoring in structural testing

C. Future Consideration in Implementing APA

This sub-section discusses the future consideration for each of the structural code coverage metrics in implementing APA. The consideration includes the importance of adopting the structural code coverage metrics, weighted scoring of the critical level of adopting the structural code coverage metrics, and scoring of the structural code coverage metrics. In addition, it also relates to how to allocate the total marks for each of the testing techniques used by educators. The findings of this section will meet the second objective identified in Section 3.

Figure 7 reveals the frequency of the different level of importance of adopting the structural code coverage in implementing APA. As shown in Figure 7, for almost all the structural code coverage metrics except for the loop coverage, the highest rating is *somewhat agree* with percentage values from 39% to 62%. Branch coverage and MC/DC seem to be among the preferred structural coverage metrics. It is also depicted that about 60% of the respondents rated *strongly agree* on the loop coverage metrics. In conclusion, it seems that all structural code coverage metrics are favored by the respondents, to be included in future APA.

Figure 8 illustrates the findings on the weighted scoring of the critical level of adopting the structural code coverage in implementing APA. As shown in the figure, the highest rating with the critical level of *moderately critical* belongs to MC/DC that is about 54% and about 46% for the path coverage, condition coverage, branch coverage. It is also shown that none of the respondents rated on the critical level of *not critical* for all structural code coverage metrics. Thus, it can be summarized that each of the individual educators prefer to have full authority to assign the weighted value in identifying the critical level of the structural code coverage if APA is implemented.

Figure 9 shows the scoring value for each of the structural code coverage metrics for future implementation of APA. It seems likely the finding shows a similar trend as the one shown in Figure 6, which emphasizes on the scoring value as applied in current practices of programming assessments. If APA is implentented, the highest rating is shown to focus on the score of 60, particularly for the condition coverage, loop coverage and multiple condition coverage with their respective percentage values ranging from 23% to 31%. From this finding, it can be concluded that the educators or lecturers desire to allocate a score of between 50 and 90 marks for structural code coverage if APA is implemented.

Figure 10 illustrates the result of overall scoring for programming assessment. The overall scoring for programming assessment is based on testing techiques. The testing techniques involved are static analysis, dynamic testing (functional or black box testing) and dynamic testing (structural or white box testing). Based on Figure 10, it can be concluded that the educators wish to allocate more marks to structural or white box testing as compared to functional or black box testing, and static analysis. This implies that the educator will give marks depending on the structure of program execution. In addition, the educators gave a slightly lower scoring for static analysis where static aspects of a program basically refers to the syntax or lexical aspect of a code [6]. For future implementation of APA, the various scoring values assigned by respondents show that educators wish to allocate the total marks for each of the testing techniques according to their own preferences.

Commonly, programming exercises are constructed based on objectives of each topic in a course syllabus [6]. Regarding the adoption of structural code coverage in programming assessments, educators may employ structural testing criteria in programming assessments in terms of statement coverage, path coverage, branch coverage, condition coverage, multiple condition coverage, MC/DC and loop coverage. Findings from the survey reveal that, the path coverage, statement coverage, branch coverage and loop coverage are among the coverage metrics that were ranked high by the respondents. This is because most of the content of introductory programming syllabi consists of sequential, selection, and iteration control structures. For novice students who are learning programming, they must at least understand and acquire related principles and concepts of these control structures so as to ensure they would be able to master the skills of programming well at the end the course.

In terms of the overall scoring of structural testing, most of the respondents agreed to allocate about half of the total marks. The remaining marks are for static analysis and functional testing. This implies that the aspect of considering structural testing in programming assessments has become an important criterion in judging the level of students programs' correctness. This survey also included the future consideration of each of the structural code coverage metrics for implementing APA. The findings reveal that the importance of adopting the structural code coverage metrics in implementing APA can be ranked by order of importance, as (1) loop coverage, (2) Statement coverage, (3) Path coverage, (4) MC/DC, (5) Multiple coverage, (6) Condition coverage, and (7) Branch coverage.

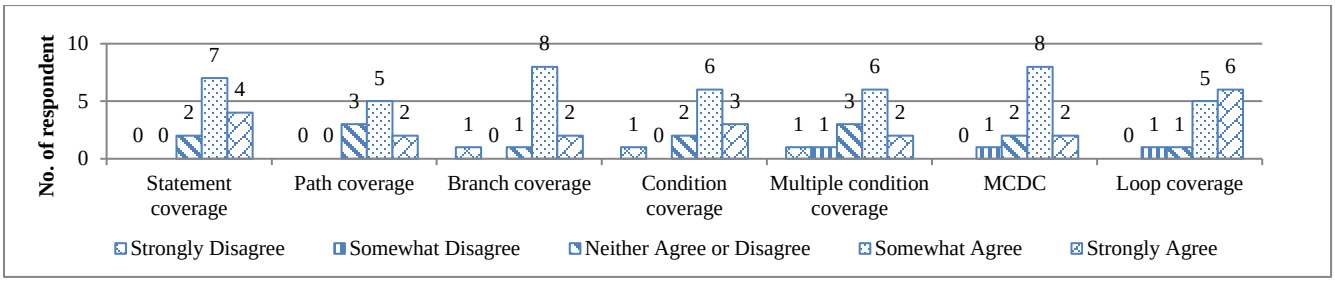

Figure 7: Importance of adopting the structural code coverage

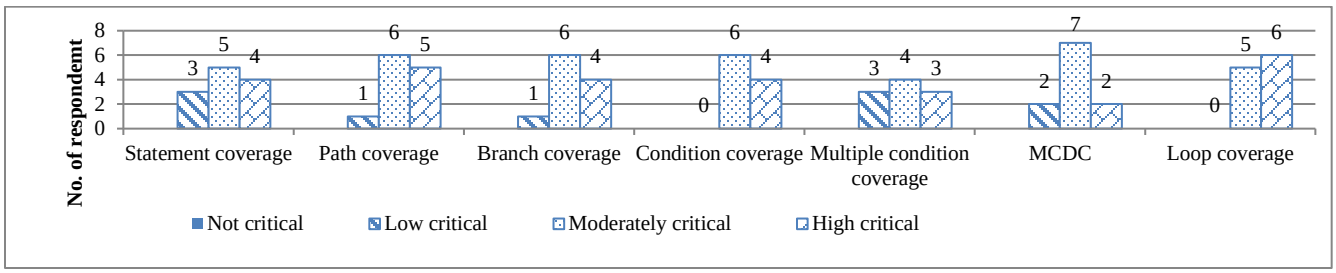

Figure 8: Weighted scoring the critical level of adopting the structural code coverage

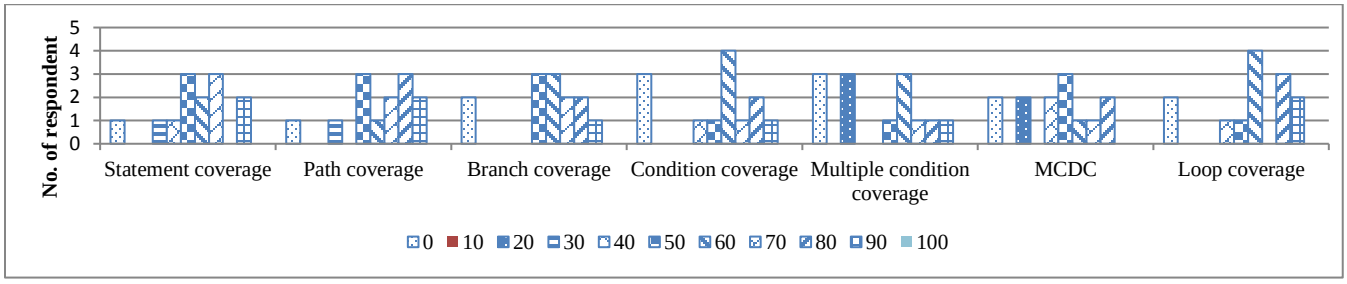

Figure 9: Scoring for each of the structural code coverage

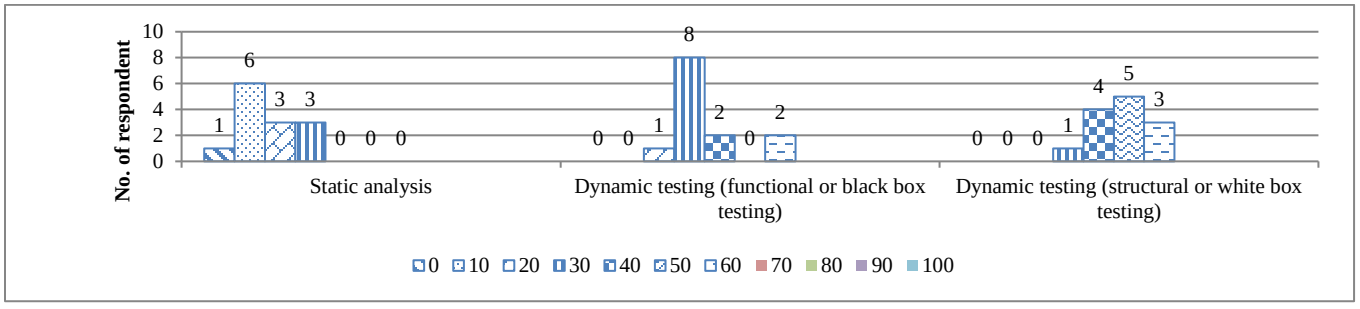

Figure 10: Overall scoring for programming assessment for each of the testing techniques

V. CONCLUSION

The conducted survey of the preliminary study reveals that most of the identified structural code coverage metrics have been employed in the current practices of programming assessments. In addition, they are being favored to be included in implementing APA in future research. In terms of the allocation of total marks for each of the testing techniques, the findings show that most of the educators prioritize white-box testing (dynamic testing) criteria more than static analysis and black-box testing (dynamic testing) criteria. This can justify the fact that structural testing plays an important role in programming assessments. Even in APA, most of the focus is more towards functional/black-box testing. Overall, we can deduce that the statement, condition, path and loop coverage are among the popular code coverage metrics employed by educators in the current practices of programming assessments. Also, it is depicted that almost all the identified structural code coverage metrics contribute as promising test adequacy criteria to realize APA. However, the promising results could be generalized if bigger samples are taken into consideration.

ACKNOWLEDGMENTS

The authors acknowledge Ministry Higher Education FRGS Fund (Code SO: 12821) of Universiti Utara Malaysia for supporting this work.

REFERENCES

[1] G. Tremblay, and E. Labonte, "Semi-Automatic Marking of Java Programs using Junit", *Proceeding of International Conference on Education and Information Systems: Technologies and Applications (EISTA '03)*, Orlando, Florida, 2003, pp. 42-47.

[2] Y. Liang, Q. Liu, J. Xu, and D. Wang, "The Recent Development of Automated Programming Assessment", *Proceeding of International Conference on Computational Intelligent and Software Engineering*, Wuhan, China, 2009, pp. 1-5.

[3] D. A. Jackson, "Software System for Grading Student Computer Programs", *Computers and Education*, 27 (3-4), 1996, pp. 171-180.

[4] R. Romli, S. Sulaiman, and K. Z. Zamli, "Designing a Test Set for Structural Testing in Automatic Programming Assessment", *International Journal of Advanced Soft Computing and Application*, vol. 5, no.3, 2013, pp. 1- 24.

[5] H. D. Chu, J. E. Dobson and I. C. Liu, FAST: A Framework for Automating Statistical-based Testing, *Software Quality Journal*, vol. 6, no. 1, 1997, pp. 13-36.

[6] R. Romli, S. Sulaiman and K. Z. Zamli., "Current Practices of Programming Assessment at Higher Learning Institutions", *CCIS 179 (Springer Berlin/Heidelberg)*. Part 1, pp. 471-485, 2011.

[7] J. Wegener, "Evolutionary Testing Techniques, Stochastic Algorithms: Foundations and Applications, Lecture Notes in Computer Science", Vol. 3777/2005, 2005, pp. 82-94.

[8] M. E. Khan, "Different Approaches to White Box Testing Technique for Finding Errors", *International Journal of Software Engineering and its Applications*, vol. 5, no. 3, 2011, pp. 1-14.

[9] J. Collofello and K. Vehathiri, "An environment for training computer science students on software testing," *Proceedings Frontiers in Education 35th Annual Conference*, Indianopolis, IN, 2005, pp. T3E-6

[10] H. Zhu, "Axiomatic assessment of control flow-based software test adequacy criteria," in *Software Engineering Journal*, vol. 10, no. 5, pp. 194-204, Sept. 1995

[11] G. I. Latiu, O. A. Cret and L. Vacariu, "Automatic Test Data Generation for Software Path Testing Using Evolutionary Algorithms," *Emerging Intelligent Data and Web Technologies (EIDWT), 2012 Third International Conference on*, Bucharest, 2012, pp. 1-8.

[12] M. Roper, *Software Testing*. London, McGraw-Hill Book Company, 1994.

[13] W. E. Perry, *Effective Methods for Software Testing*, 2nd Edition, John Wiley & Sons, Inc, USA, 2000.

[14] I. Sommerville, *Software Engineering*. 7th Edition. Pearson-Addison Wesley, USA, 2004.

[15] B. Beizer, *Software Testing Technique*, 2nd Edition. Van Nostrand Reinhold, New York, 1990.

[16] J. K. Hayhurst, D. S. Veerhusen, J. Chilenski, and L. K. Rierson, "A Practical Tutorial on Modified Condition/Decision Coverage" *NASA STI Perogram* , 2001, pp. 7-9.

[17] U. Sekara, *Research Methods for Bussiness: A Skill Building Approach*, 4th Edition, John Wiley & Sons, Singapore, 2003.