A Critical Review on Recent Proposed Automated Programming Assessment Tool

Muhammad Huzaifah Ismail¹, Muhammad Modi Lakulu²*

¹,²Department of Computing, Sultan Idris Education University, Malaysia
modi@fsik.upsi.edu.my²*

Abstract: Automated assessment tool for programming assignments has been discussed extensively in education technology ever since its existence. The tool aims to reduce the work load of instructors as well as providing an instant feedback to students in pertaining to assessing programming assignments. Myriad versions of the tool accompanied with multiple features has been proposed till this day in order to cater modern trend of education technologies. Many previous studies have analyse on these existing tools but not on recent one. This study have conducted a critical review on the recent proposed tool based on its pedagogical and technical aspect. As a result of this study, more recent proposed tool are identified and its challenging issues are also highlighted.

Keywords: Education Technology, Automatic Assessment, Programming Assignments, Tool Analysis

1. Introduction

Programming in the context of computing, is an activity of writing instructions to tell the computer how to process specific information[1]. It is an essential practical skill that needs to be obtained especially by those who want to pursue a career in Computer Science (CS) field. This is because the ability of programming contribute many things in CS field especially in the software development. Therefore, undergraduate students who enrol in CS course at University will involve with many programming assignments. Typically for the introductory programming course, the instructor will design the programming assignment that requires the students to develop a functional program where it can perform problem-solving. Therefore, the student have to develop a functional program that executes correctly as specified within the assignment outline as part of their classroom assessment task.

However, assessing these programming assignments manually arises many issues which including inaccuracy of assessment[2] and more effort and time consuming[3] especially with large number of students enrolment. The reason for this issues to occur is because the nature of programming consists of codes that have different logics and constructs but still can producing the same output[4]. In other words, student can apply multiple approaches in programming but still it can lead to the same expected solution. Therefore, it is difficult for the instructor to determine the correctness of a programming assignment and it can become a slow tedious process because the code of the student’s program have to be examine line by line and test it thoroughly whether it follows the specified requirement. To address this issues, researchers have come out a solution known as Automatic Programming Assessment Tool (APAT) which is a software tool that can evaluate student’s programming assignment automatically.

In order to cater modern trend of education technologies, myriad versions of APAT accompanied with innovative features are proposed till this day. Many previous studies have analyse on these existing APAT but not on recent one. Thus, this study have conducted a critical review on the recent proposed APAT from its pedagogical and technical aspect. The novelty of this study is that more recent proposed APAT are identified and highlighted. In addition the challenging issues with the recent trend of APAT development are also discussed throughout this study. As a result, the highlights of these details allows to determine the research gaps of APAT for future studies to address.

The remainder of the paper is organized as follows. These second section are discussing briefly on the background of APAT. While, the third section highlights the related work of this study. The fourth section on the other hand demonstrating the methodology of the study from identifying the recent proposed APAT to classifying its pedagogical and technical aspect. The fifth section presents the discussion that relates to recommended studies for future research of APAT. Lastly, the fifth section concludes the study.
2. Background

The very first APAT was introduced in 1960 by Hollingsworth where the correctness of the student program functionality is evaluated automatically [3], [5]. Fundamentally, how the proposed APAT works is that the grader program was run against the student program and produce two types of results which either “wrong answer” (for rejected solution) or “program complete” (for successful solution). The proposed APAT was specifically designed to evaluate assembly programming language. Henceforth, the automatic evaluation feature has always been the main goal of APAT and later on has been extended to other features as well such as test data generations [6]–[8], semi-automatic approach of assessment [9], [10], plagiarism detection [11], [12], automated feedback messages [13] and so forth.

The Three Types of Analysis

Automatic evaluation feature has always been the integral part of the APAT since of its introduction. Automatic evaluation feature comprises of three types of analysis which are dynamic, static and hybrid analysis. Each of the analysis evaluate student’s program on different grading criteria. The general grading criteria that instructor usually applied on assessing programming assignments include program execution, program specifications, program design, coding style, comments, and creativity [14]. So, it is crucial for the instructors aware of these types of analysis on its capability and its limitation so they able to choose the suitable APAT for their usage.

Dynamic Analysis

The dynamic analysis (DA) was based on the first proposed APAT where the student’s program is executed and the evaluation is based on the comparison between the output produced by the student’s program and the expected output [15] that either provided manually by the instructor or automatically generated [6], [16]. Simply put, DA is the same concept as applying the black-box and grey-box that has been highlighted in software testing methodology [17]. DA was designed to cater the program execution criterion whereby the student’s program should be able to compile and executes cleanly, and produce correct results. The correct results in the perspective of DA is that the student’s program was able to produce the desired output. Thus, this is how the correctness of student’s program is determined.

Technically, DA works by taking the source code file as input and then compiles the code, a file containing a set of input test cases is fed to the program and the generated output from the program is then redirected to a file and it is compared with the file that contains output test cases [18]. If both of the output files passes the test cases comparison, then it is considered the student have successfully answered the programming assignment. Figure 1 demonstrate the flowchart diagram on how the DA works.

Figure 1. The flowchart of dynamic analysis

The main advantage of applying DA is that it is easy to implement in APAT and the testing can be non-technical [17] for the instructor to applied. What meant by this situation is that the APAT only have to analyse the student’s program by verifying the input and output matching test cases despite evaluating the whole program structure which requires more sophisticated mechanism. Nonetheless, the major drawback of implementing DA in APAT is that the risk of damaging the APAT itself and its system environment is high.
since DA requires the execution of the student program in order to proceed its evaluation process. The execution of the student’s program could lead the APAT to execute a malicious program if there is no proper security measurement implemented[19]. Consequently, it is important for the APAT to embed with proper security mechanism such as sandboxing[20] explicitly in the implementation of DA.

**Static Analysis**

The static analysis (SA) on the other hand evaluate the student’s program without executing it[21] which are then used to provide feedback based on the correctness of the student’s program structure [20]. In comparison to DA, SA is safe to apply because it does not involve with the student’s program execution. As a result, SA automatically preventing the APAT from harming itself with malicious program. In addition, student programs that possessed compilation errors are also feasible to be evaluated with SA since it does not involve program execution [19]. SA covers more grading criteria of programming as opposed to DA that includes program specifications, program design, coding style, comments, and creativity as described before.

The operation of SA performs can applied many techniques as long it is able to evaluates the program structure. One of the techniques that SA primarily applies is call as structure similarity analysis. The SA that carry out structure similarity analysis technique operates by calculating the similarity of the program structure between the student’s program and the solution model program[22]. A solution model program essentially is a file that contains source code that is written and provided by the instructor. Once SA is performed, the correctness of the student’s program structure is determined based on how similar it is to the solution model program. The intermediate representation form can be either a graph, tree, or pseudocode which depending on what technique the structure similarity analysis used for the evaluation process. The typical techniques for structure similarity analysis mostly adopting are graph matching[17], [23], abstract tree syntax [24] and pseudocode comparison[25]. Figure two shows the flowchart of the basic mechanism of SA that is implemented in APAT.

![Figure 2. The flowchart of static analysis](image)

Despite SA has more advantages over DA which is safer to be applied, covers more grading criteria and able to evaluate student’s program that has compilation errors, it still has limitation of its capability. Its major limitation is the limitation of analysing of various approaches of student’s program structure[17], [22], [23]. As previously discussed (see section 1), the nature of programming is that the program structure can consists of codes that have different logics and constructs but still it can lead to produce the desired output. Then, it is difficult for the SA mechanism to be able to evaluate all types of program structure. There is no doubt that the
design of SA mechanism is more difficult to come out with especially ensuring complex program is feasible to be evaluated. This is when DA has advantages over SA because it evaluates solely on the output comparison rather than the program structure itself. Subsequently, SA is more suitable for less complex programs due to the fact that it is easier to perform the evaluations.

Hybrid Analysis

Last but not least, hybrid analysis (HA) which is a combination of DA and SA. The perks of applying HA in automatic evaluation feature allows the complementary of the strengths and limitations of both of the analysis types. A clear of the example of HA can be seen under Zougari et al [17], [23] study where the DA is initiated at the first stage of the evaluation process and followed by SA at the second stage.

As indicated in their study, the process of DA is to ensure the students program is able to execute and produce the desired by output through automated testing process. As usual, the DA mechanism requires a set of output test cases and it applies it for the evaluation process. While the process of SA in the study, the authors has adopted graph matching technique for the structure similarity analysis to process. Clearly, the implementation of the HA helps to improve the accuracy of the assessment since is not only able to perform output comparison for the evaluation but it also inspecting the program structure as well.

3. Related Work

Many previous studies have analysed the existing APAT and come out an interesting discussions on it. Ala-Mutka [21] have surveyed existing APAT based on its analysis types that consists of DA and SA (read section 2.1 for details). In addition, Ala-Mutka have mentions the limitation of DA which prefers to apply semi-automatic assessment approach as an alternative for the issues. Ala-Mutka also recommended the direction of APAT studies should be developed on globally scale instead of locally so that other universities courses can adopted it as well.

Douce et al [3] on the other hand, have classified three main generations of APAT development that are specifically applied DA for its automatic evaluation feature. The first generation is an initial attempt to automate the assessment of programming assignments activity. Later, the second generation is where the command-line-based of APAT started to existing. Lastly, the third generation of APAT are the web-based type.

Ihantola et al [20] study presents the key features of APAT that supported by the tools and the different approaches used both from pedagogical and technical point of view. Somehow, Ihantola et al works does not list out the tools aligning with the identified key features. Due to that, Souza et al [26] have address this gap by listing out the existing tools and perform classifications based on its features, interface type and programming languages supported. However, all of these related studies seems have not focusing on identifying and highlighting the specifications of more recent proposed APAT. Thus, this study would like to address this gap by analysing on recent proposed APAT as well as promoting recommended studies that should be focus in the future. The next sections explaining the on how this study applied its methodology.

4. Methodology

This study have selected 10 studies that are specifically describing the proposed APAT based on its characteristics such as mentioning the tool name and its features. As aforementioned before, this study are targeting on the recent proposed APAT since no other studies have covered on it. Therefore, the search for the studies has been restricted to only from year 2014 till 2019. These identified studies are later referred as the selected tools which are used in table two. Table one presents the summarization of the identified recent proposed APAT according to the tool’s name, description, and the year of the article was published.

<table>
<thead>
<tr>
<th>Label</th>
<th>Tool Names</th>
<th>Tool Description</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Judge (UOJ)[27]</td>
<td>System UOJ is an online web-based of APAT that is developed to assists instructors in terms of assessing programming assignments in addition to allows the students practices C++ programming language in UniversitiTeknologi PETRONAS. The UOJ was also built in with a swarm inspired automatic test case generation feature which are known as Particle Swarm Optimization (PSO) algorithm.</td>
<td>2014</td>
</tr>
<tr>
<td><strong>T2</strong></td>
<td>Automata[28]</td>
<td>Automata is a machine learning based of APAT that uses Latent Semantic Analysis (LSA) technique to assess the student program. To facilitate the automatic grading feature of the tool, a rubric was designed and integrated. The rubric outline the grading based on the correctness of the student’s program functionality.</td>
<td>2014</td>
</tr>
<tr>
<td><strong>T3</strong></td>
<td>FaSt-generator[29]</td>
<td>FaSt-generator is an APAT that is specialized in automatic test case generation. The tool built on top of test data generation framework which was based on their previous work[8], [30]. The tool purpose is to generate an adequate set of test data that are used to execute both functional and structural testing for the evaluation process.</td>
<td>2015</td>
</tr>
<tr>
<td><strong>T4</strong></td>
<td>SAUCE[31]</td>
<td>SAUCE as it stands for “System for AUtomated Code Evaluation” is a web-based of APAT that assists instructors on assessing parallel programming assignments. It applies a lightweight queue system for running the submission tests on assessing the assignment submission. SAUCE has been tested on various examples of parallel programming which includes OpenMP, MPI and CUDA.</td>
<td>2015</td>
</tr>
<tr>
<td><strong>T5</strong></td>
<td>Ask-Elle[32]</td>
<td>Ask-Elle is a tutor type of tool that support the stepwise development of simple functional program in Haskell. Ask-Elle is consider as an APAT due to its ability of evaluating the correctness of incomplete students programs, giving hints and allows instructors add particular feedback information on the programming exercises.</td>
<td>2015</td>
</tr>
<tr>
<td><strong>T6</strong></td>
<td>Programing Grading Assistant (PGA)[33]</td>
<td>PGA essentially is a mobile apps version of APAT that uses Qr-Code Scanner and Optical Character Recognition (OCR) mechanism to evaluate handwritten programming exercises. The researchers also design a framework namely Mobile Grading Framework (MGF)which later has been adopted in PGA system architecture.</td>
<td>2016</td>
</tr>
<tr>
<td><strong>T7</strong></td>
<td>Online Learning System[34]</td>
<td>The authors somehow does not specify the name of the proposed APAT and simply mention it as an online learning system. The online learning system is implemented on open source learning platform call Moodle and its automatic evaluation feature is based on Virtual Programming Lab (VPL) plugin.</td>
<td>2017</td>
</tr>
<tr>
<td><strong>T8</strong></td>
<td>Aristotle[35]</td>
<td>Aristotle is an APAT that applies semi-automated assessment with the capability of assessing any types of programming languages. At the same it is integrated with GitHub Classroom which is a free web-based tool for the usage of instructors which was basically use for managing the assignments.</td>
<td>2017</td>
</tr>
<tr>
<td><strong>T9</strong></td>
<td>ASys[36]</td>
<td>The authors of the articles has presented two major contributions in their studywhichare developing a web-based APAT that they called as ASys and a new automatic evaluation feature that they refer as JavAssess. ASys purpose is to assists instructors for assessing final exams and permits student using the tool for self-evaluate learning on Java programming. The JavAssess is basically a java library that applies a HA of evaluation mechanism for evaluating java codes.</td>
<td>2018</td>
</tr>
<tr>
<td><strong>T10</strong></td>
<td>Paprika[37]</td>
<td>Paprika which is the name of their proposed APAT applies HAfor its automatic evaluation feature. The usage of paprika have proven that it can produced correct scores and accurate feedback even in student’s program that has weak output. Weak output as describe by the authors is an output format that is dissimilar from the output required by the test case.</td>
<td>2019</td>
</tr>
</tbody>
</table>
In table two, this study has perform a classifications from the 10 selected tools that are list out in table one. The classifications of these tools are based on two aspects which are the technical and pedagogical aspect. In the context of this study, technical aspect of APAT are elements that covers more on its functionality. The specifications of these elements helps in determining the recent trend of proposed APAT functionality and also provides details of improvement for new development of APAT in the future. Therefore, the technical aspect this study are emphasising on its novelty feature, analysis types that uses for its automatic evaluation feature, programming languages that supported and its assessment approach type whether its automatic or semi-automatic. In other words, the technical aspects that this study are focusing are mostly involving with the APAT mechanism. As for the pedagogical aspect of APAT, this study have specified whether the tools usesrubric for its grading and also the involvement of learning taxonomy. Both of these elements are significant to be applied in APAT especially for its pedagogical aspect. Essentially, rubric is a documentation that lays out the specific expectations of grading [38]. The importance of applying rubric for grading assignments is because of it capable of providing consistency of the grades [39]. So unfair grading towards the student’s assignment can be prevented if a rubric is use. While, learning taxonomy can be refer as the scheme for classifying educational goals, objectives, standards[40] and grading as well [4], [41]. The significant of using learning taxonomy in assessment helps in measuring the students learning performances.

Table 2. The classification of recent proposed APAT from its pedagogical and technical aspect

<table>
<thead>
<tr>
<th>Tools</th>
<th>Novelty feature</th>
<th>Analysis type</th>
<th>Programming Languages supports</th>
<th>Assessment approach</th>
<th>Applying rubric</th>
<th>Applying Learning Taxonomy</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Automatic test cases generations</td>
<td>DA</td>
<td>C++</td>
<td>Automatic</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>T2</td>
<td>Machine learning approach</td>
<td>SA</td>
<td>More than one languages</td>
<td>Automatic</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>T3</td>
<td>Automatic test cases generations</td>
<td>HA</td>
<td>Java</td>
<td>Automatic</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>T4</td>
<td>Assessing non-introductory programming course</td>
<td>DA</td>
<td>More than one languages</td>
<td>Automatic</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>T5</td>
<td>Assessing non-typical programming languages</td>
<td>DA</td>
<td>Haskell</td>
<td>Automatic</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>T6</td>
<td>Assessing handwritten programming paper based</td>
<td>SA</td>
<td>Java</td>
<td>Automatic</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>T7</td>
<td>Plugin system based</td>
<td>DA</td>
<td>More than one languages</td>
<td>Automatic</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>T8</td>
<td>Plugin system based</td>
<td>DA</td>
<td>More than one languages</td>
<td>Semi-automatic</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>T9</td>
<td>HA approach</td>
<td>HA</td>
<td>Java</td>
<td>Automatic</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>T10</td>
<td>HA approach</td>
<td>HA</td>
<td>More than one languages</td>
<td>Automatic</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

5. Results and Discussions

From previous section, this study have classified the ten identified recent proposed APAT according to its pedagogical and technical aspects. The elements of both aspects have been specified throughout the classifications process. The followings are the findings based on the classifications that have been outline from the previous section:

- The findings of recent proposed APAT based on its novelty feature
In the context of this study, novelty feature of APAT can be considered as a feature that is designed to facilitate a particular scope of the study. Usually, the authors specify the scope of study and the proposed APAT need to be able to cater it and this can be referred as the novelty feature. From the findings of this study, seven categories of novelty feature are discovered and the results are presented in Figure 3. HA approach as previously discussed, are the type of analysis that combines both static and dynamic and from Figure 3, two out of ten tools which are T9 and T10 possess this novelty feature. T7 and T8 on the other hand are specialize in plugin based architecture. Only one tool that is T6 has the capability of assessing handwritten programming paper based which is very unique as opposed to the rest which are more focusing on the digitalization. Besides that, one particular tool that is T5 is built specifically to assess Haskell programming language. It seems the tool aiming on assessing the non-typical programming language as comparing to the typical one such as C, C++, C#, Java and python. Another tool that has unique traits in relation to novelty feature that is T4 has the ability assessing of parallel programming which is the tool is focusing on more advance programming course. Last but not least, one tool that is T2 has incorporating a machine learning approach within their study.

- The findings of recent proposed APAT based on its analysis type

![Bar chart showing the total number of tools based on the adoption of analysis type.](image)

**Figure 4.** Total number of tools based on the adoption of analysis type

Based on Figure 4, clearly it can be seen that DA is highly adopted within their proposed tool as its automatic evaluation feature which is at 50%. The main issue of these tools that have adopted DA is that none of these are including any type of security mechanism such as sandboxing, CPU usage limitation, and so forth. This is crucial for the sake of these tools system environment since the potential of executing a malicious code is high. However, the tools that adopted SA and HA is at 20% and 30% which is lower compared to DA.

- The findings of recent proposed APAT based on its programming languages that the tools supported
It seems that the recent trend of the tools are striving for supporting more than one type of programming languages as depicted in figure 5. The results in figure five shown that 50% of the tools has the ability of evaluating more than one programming languages. As for the tools that supported only one programming languages, Java has most applied within the tools which is at 30% and C++ only at 10% and same goes for Haskell which is at 10%.

- The findings of recent proposed APAT based on assessment type of approach

The types of assessment approach in APAT can be implemented in two ways that is either fully automatic or semi-automatic. The semi-automatic assessment is basically the types of assessment approach that only performs partial assessment. This approach was presented to eliminate the limitation of the DA type of automatic evaluation feature where the evaluation was only based on the output comparison of test cases. Based on figure six, only one tool that is T8 have applied the semi-automatic approach mechanism.

- The findings of recent proposed APAT based on rubric usage

From the results depicted in figure 6, there is no doubt that rubric seems very poorly applied in the recent proposed APAT since only T2 does include it in its automatic grading feature. Meanwhile, a rubric is an essential instrument for ensuring the grades are given consistently to any type of level of assignment as previously mentioned. The implementation of rubric in APAT helps to make the grades non-binary as oppose to previous grading mechanism especially the first APAT that was introduced (read section 2 for further details).
Moreover, the proposed APAT should have not consider non-binary grading since it is difficult for student to perform self-assessment.

- The findings of recent proposed APAT based on the application of learning taxonomy

![Figure 8. Total number of tools that applied any learning taxonomy](image)

Same goes for this situation, the awareness of applying learning taxonomy within the APAT seems very low. In fact, it is much worse since none of these tools have applied it as presented in figure seven. The importance of applying the learning taxonomy in APAT supports instructor in terms of monitoring students learning achievement. This results shows that these recent tools have neglected the essential element of an assessment which is using learning taxonomy as a guidance for monitoring students learning achievement.

6. Conclusion

This study have conducted a critical review based on ten recent proposed APAT (2014-2019) by classifying its pedagogical and technical aspect of the tool. The findings of this study discovered that most these recent proposed APAT are very weak in the pedagogical aspect compared to its technical. Thus, for future studies of APAT, researchers should focus more on incorporating the pedagogical aspect especially the usage of rubric for grading and the application of learning taxonomy in highlighting students learning achievement.

As for the technical aspects of APAT, researchers should strive on proposing tools that uses hybrid analysis for its automatic evaluation feature as oppose to static and dynamic. This is due to the fact that the perks of having hybrid analysis as its automatic evaluation feature allows the complementary of the strengths and limitations between static and dynamic. Nevertheless, there are no studies highlighted the overall performances between these three analysis type which researchers should focus as well.

References


35. M. D. Adams, “Aristotle: A flexible open-source software toolkit for semi-automated marking of