Assessment of Automated Evaluation Process Parameters

Maxwell Christian
GLS University
maxwell.christian@glsuniversity.ac.in

Prof. Bhushan Trivedi, Ph.D
GLS University
bhushan.trivedi@glsuniversity.ac.in

Abstract

In the current digital era, e-learning plays an important role in education. Many e-learning courses are widely available and offered by a large number of reputed institutes. And hence the work of assessing the work in such courses scales to a large extent. An automatic evaluation process can aid in bringing down the amount of time consumed also to have a clear and non-inclined result to the process of the assessment. This paper provides study and comparison of the existing tools for automatic evaluation of programming work. This paper focuses on the study of these existing tools on the basis of different parameters which addresses evaluation parameters of programming work.

Keywords: *Automatic Assessment, Programming Assessment, Programming Evaluation Tools, Automatic Assessment Tools, Automatic Assessment Process, Programming Assessment Process*

1. Introduction

In current era the use of online medium in education has increased at a great scale. Learners have a wide variety of choice from various online courses. Hence in also results in need of automatic assessment in such online learning environment.

The conventional method, procedural and formulated evaluation, of the practical work performed by a student, irrespective of the methodology incorporated by him, has always been a challenging task[1][2][3][5]. Continuous improvisations in evolution process has been thus observed[5][6]. Existing research enlists few ground rules for assessment[2][5].

1. There can be multiple approaches for solving a given problem and at times both student and evaluator is unaware of it[9][10]

2. Solution implemented is not exactly aligning with actual requirement and a good amount of time is infused by students in doing things not required by the assigned task.[7][8]

3. An assessment approach, dynamic in nature, is required that assesses on the grounds of adopted methodology for solving the assignment irrespective of the adopted solving approach be conventional or totally innovative.[14][16]

4. Ample amount of time gets consumed by the student in solving trivial syntactical/logical errors, generated due to non systematic approach towards solving the assigned task, which the expert evaluator can point out quickly on basis of his experience[16][18]. The same amount of time, of both students and evaluators, gets consumed in the repetitive process of solving the same errors again when overlooked by the students, hence delaying the process of final outcome of the assignment.[18][19]

5. Prevalent techniques and tools turns out to be non-beneficial due to their lacking to address diverse assessment types and also because many of such tools are on research stage.

This paper compares types of existing assessment tools and approaches relevant to automatic assessment and their comparison.

2. Existing Evaluation Approaches

Static and dynamic analysis of code are two widely classified categories of parameters for automatic evaluation of programming assignment. Also quantitate and qualitative assessment are the major ones to be addressed for programming assignments. The study provided in this section focuses on existing tools that addresses different components for evaluation[11] of programming assignments.

1. Code correctness: The program can be evaluated on the basis of the way in which the program has been coded i.e. the coding conventions and coding style. Only WebCat [7] and Virtual Programming Lab deal with code correctness and evaluate only the pre-defined set of conventions. Lacunas: Function conventions are not addressed.
Suggested: This if addressed can help the student to understand the concepts of cohesion and coupling amongst the program modules

2. Functionality: Program functionality is addressed only by CourseMarker[3] and constraints only to the input and output actions of the program.
Lacunas: Input and output operational functionality are the only parameters dealt with.
Suggested: Here if the type of inputs and possible outputs are qualitatively measured, can help the students to achieve better quality programming abilities

3. Test Validity: Standard way of submission of assignment is addressed by WebCat [7] which is widely accepted due to its web interface with possibility to be used as a plugin.
Lacunas: It is a plugin and works with grading submissions rather than the entire program itself.
Suggested: Grading submissions along with explanatory feedback would have been an addition

4. Code analysis: Code analysis is addressed by PETCHA [9] and Grading Tool from Madgeburg University [4]
Lacunas: Not a complete tool on its own. Requires to be used with different IDEs.
Suggested: Dynamic code analysis, if provided, can be measure of qualitative assessment

5. Comparative analysis: Comparing submitted work with testing set is provided only by MOODLE Ext developed by Slovak University [11]. It is a plugin to the MOODLE system and thus handles a very limited type of question categories provisioned by MOODLE.
Lacunas: It is not a complete independent system but a plugin to MOODLE
Suggested: An adaptive approach where a new tested set is added to the learning set could enrich the assessment process

6. Grade Analysis: Dynamic grading as per given parameters is also a major component in automated evaluations. Such dynamic grade analysis and is so far supported only by AutoLEP[9].
Lacunas: Does not support test case additions
Suggested: A self learning model which can learn new test cases would be an added benefit for assessment process

7. Language Support: C, C++, JAVA, PHP, Python and Ruby language support is available so far. Only Marmoset [10] supports the listed languages while Virtual Programming Lab and WebCat [7] has support only for PHP and Python.
Lacunas: The existing tools do not have a self – learning and adaptive mechanism and works on a specific set of data.
Suggested: Self learning tool with adaptive mechanism of new cases will always be beneficial for the assessment process

3. Existing Tools

A comparative study of existing assessment assisting tools[7] is presented in this section.

1. CourseMarker[3]: It is a tool supporting the programs written in C++ and JAVA and developed using JAVA by the Nottingham University around the year of 1998 – 1999. It takes care regarding the functionalities of user authentication, courses, submission, grading the assignments and managing the grades and auditing the user actions as well as communicating with other users.

2. Marmoset[10]: Marmoset is developed by the University of Maryland and was implemented in the academic year of 2004. The prime advantage of this tool is the capturing of the entire work of the student, use of different types of tests, providing posts on the code and others. The tool also has a wide support of languages like C, Java and Ruby. The tool is largely developed using J2EE architecture and is basically a web – based system and hence has high accessibility and scalability

3. WebCat[7]: WebCat [Web based Center for Automated Testing] is a tool for self-evaluating the programming assignment by the students. It is majorly built using Java servlets and JFC. It is basically a web application with plugin style architecture and was originally conceived by Virginia Tech, as a portal for student – focused services. It has been in wide used and as of has been used with 80 different courses globally. It supports C++, Java, Prolog, Pascal, and also provides flexibility to support many other programming languages. It is widely used due to it prominent features of extensibility, flexibility, submissions, posts, suggestions, grading and grade modifications.

4. Grading tool by Madgeburg University[4]: This tool can be reached using services and demonstrates a configurable focus. It provides selection for various components like compiler to use, grading schemes and the set of different data. Also the submissions limit can be constrained. It supports a wide variety of languages like Prolog, Python, and JAVA and used XML-RPC based communication to fulfil the service oriented architecture

5. JavaBrat[7]: This tool was developed at San Jose State University and provides a support of two languages namely JAVA and Scala. The grading mechanism is developed using JAVA while the support for MOODLE is provided as a plugin. It can also work as a stand-alone system though a web interface built using JSF and the service calls are implemented using JAX-RS. This is semi – automatic tool and report generation needs to be initiated after the grading mechanism

6. AutoLEP[9]: This tool was developed at Harbin Institute of Technology and boasts of static as well as dynamic analysis for providing grade. The static analysis is all about the syntactic and semantic analysis, which is a solely available feature with this tool, while the dynamic analysis evaluates the correctness through test cases. This tool is also having a stand-alone architecture

7. Petcha[9]: This tool, developed at University of Porto, aims mainly at working as an automatic assistant in teaching programming. The prime feature is interoperability with other systems like LMS, IDEs, etc. Hence it supports almost all programming languages as supported by the IDEs. This tool has been successfully tested with well known IDEs like Eclipse and Visual Studio.

8. JAssess[7][9]: This tool was developed as a collaborative work of two different universities of Malaysia, namely, University of Technology and Tun Hussein Onn University. It is primarily as MOODLE plugin and built with the purpose to work as an assessment module. The evaluation process with this tool is not completely automatic and it supports only JAVA language.

9. RoboLIFT[4][7]: This is the only tool with a focus of mobile application and so it helps in grading ANDROID applications. It is based on WebCat and uses the development tools for Eclipse as provided by GOOGLE and considers unit testing as a metric to grade

10. Virtual Programming Lab[8]: The tool developed at Las Palmas University aims to provide students many programming assignments and also manage the grading process and it achieves the same by integrating itself with LMS system. It supports a variety of programming languages like C, C++, Python, PHP, Ruby, and others. It uses a browser based code editor, a MOODLE plugin for submission and grading, and a JAIL server for evaluation. For grading it considers the unit test cases and is built under the GNU/GPL license and also has plagiarism-controlling feature.

11. Moodle Extension by Slovak University[11]: The main goal of this tool is managing assignment and user type related features. The only language supported in VHDL. The submission is evaluated on the basis of compilation and syntactic testing and also supports plagiarism detection.

Table 1. Tools comparison on basis of type of architecture

Sr No	Tool	Scalability	Configurability	Feedback	Language Independence	Portability	Grade Analysis	Architecture
1	CourseMarker	Y	Y	Y				Stand alone
2	Marmoset	Y			Y			Stand alone
3	WebCat					Y		Web based
4	Virtual Programming Lab						Y	MOODLE plugin
5	Grading Tool (Madgeburg University)		Y					LMS extension
6	JavaBrat		Y					Standalone and MOODLE plugin
7	AutoLEP						Y	Standalone
8	Petcha					Y		Standalone
9	JAssess		Y					MOODLE plugin
10	RoboLIFT						Y	Standalone
11	MOODLE Ext. (Slovak Uni)		Y					MOODLE plugin

Table 2. Supported Grading Metrics

Tools	Static			Dynamic			Grade Analysis
	Typographic Assessment	Documentation Check	Lexical Analysis	Semantic Analysis	Cohesion Check	Schematic Analysis	
CourseMarker	Typography					OO Concepts	
Marmoset	Not specified	Not specified					
WebCat						Code correctness, Test Validity	
Virtual Programming Lab						Correctness	
Grading Tool (Madgeburg University)			Compilation	Execution			
JavaBrat						Correctness	
AutoLEP						Static grades	Static grading
Petcha							Grades based on test cases
JAssess			Compilation				
RoboLIFT				Unit testing			

MOODLE Ext. (Slovak Uni)			Compilation	Schema comparison			

4. Conclusion

After the comparative study of the tools mentioned in Table 1 and Table 2, it is clearly evident that there exits a wide gap of overall and qualitative first hand assessment of the student's work. Also is is found that the existing tools and approaches suffice to assessment which does require evaluator intervention on a larger scale. There also exist lacunas pertaining to quantitative and qualitative assessment of programming assignments. One more important factor identified is that the approaches and tools existing so far does have any self - learning mechanism to evolve on basis of variety of problems evaluated gradually. Hence it can be concluded that a self learning model incorporating qualitative assessment if available can be beneficiary for better assessment and evaluation of the student's programming assignment.

References

1. Forsythe, G. E., Wirth, N. (1965). Automatic Grading Programs. Commun ACM, vol. 8, pp. 275-278.
2. Higgins, C. A., Gray, G., Symeonidis, P., Tsintsifas, A. (2005). Automated Assessment and Experiences of Teaching Programming. Journal on Educational Resources in Computing (JERIC), vol. 5, pp. 5.
3. Douce, C., Livingstone, D., Orwell, J. (2005). Automatic Test-based Assessment of Programming: A Review. Journal on Educational Resources in Computing (JERIC), vol. 5, pp. 4.
4. Ihantola, P., Ahoniemi, T.,Karavirta, V., Seppälä, O. (2010). Review of Recent Systems for Automatic Asessment of Programming Assignments. Proceedings of the 10th Koli Calling International Conference on Computing Education Research, pp. 86-93.
5. Romli, R., Sulaiman, S., Zamli, K. Z. (2010). Automatic Programming Assessment and Test Data Generation a Review on its Approaches. Information Technology (ITSim), 2010 International Symposium, pp. 1186-1192.
6. Caiza, J. C.; Del Alamo, J. M. Programming Assignments Automatic Grading: Review of Tools and Implementations. Inted 2013 Proceedings, 2013, 5691-5700
7. Rodríguez-del-Pino, J. C., Rubio-Royo, E., Hernández-Figueroa, Z. J. (2012). A Virtual Programming Lab for Moodle with Automatic Assessment and Anti-plagiarism Features.
8. Queirós, R. A. P., Leal, J. P. (2012). PETCHA: A Programming Exercises Teaching Assistant. Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education, pp. 192-197.
9. Spacco, J., Hovemeyer, D., Pugh, W., Emad, F., Hollingsworth, J. K., Padua-Perez, N. (2006). Experiences with Marmoset: Designing and Using an Advanced Submission and Testing System for Programming Courses. ACM SIGCSE Bulletin, vol. 38, pp. 13-17
10. Jelemenská, K. Čičák, (2012). Improved Assignments Management in MOODLE Environment. INTED2012 Proceedings, pp. 1809-1817.
11. Jackson, D., & Usher, M. (1997). Grading student programs using ASSYST. Proceedings of the Twenty- Eighth SIGCSE Technical Symposium on Computer Science Education, USA, 335-339.
12. Schorsch, T. (1995). CAP: An automatic self-assessment tool to check Pascal programs for syntax, logic and style errors. Proceedings of the 26th SIGCSE technical symposium on Computer science education, USA, 168-172.
13. Emma Enstro¨m, Gunnar Kreitz, Fredrik Niemela, Pehr Soderman, and Viggo Kann. 2011. Five Years with Kattis – Using an Automated Assess- ment System in Teaching. In Frontiers in Education Conference (FIE), 2011. Institute of Electrical and Electronics Engineers, Piscataway, NJ, T3J–1.
14. Mike Joy, Nathan Griffiths, and Russell Boyatt. 2005. The BOSS on- line submission and assessment system. ACM Journal on Educational Resources in Computing 5, 3, Article 2 (Sept. 2005), 28pages.DOI:http://dx.doi.org/10.1145/1163405.1163407
15. Lauri Malmi, Ari Korhonen, and Riku Saikkonen. 2002. Experiences in automatic assessment on mass courses and issues for designing virtual courses. ACM SIGCSE Bulletin 34, 3 (2002), 55–59.

16. Jacques Philippe Sauve´, Oso´rio Lopes Abath Neto, and Walfredo Cirne. 2006. EasyAccept: a tool to easily create, run and drive development with automated acceptance tests. In Proceedings of the 2006 international workshop on Automation of software test (AST '06). ACM, New York, NY, USA, 111–117. DOI:http://dx.doi.org/10.1145/1138929.1138951
17. Brad Vander Zanden, David Anderson, Curtis Taylor, Will Davis, and Michael W. Berry. 2012. CodeAssessor: An Interactive, Web-Based Tool for Introductory Programming. The Journal of Computing Sciences in Colleges 28, 2 (2012), 73 – 80.
18. Yuen Tak. Yu, Chung Keung Poon, and Marian Choy. 2006. Experiences with PASS: Developing and Using a Programming Assignment Assess- ment System. In Quality Software, 2006. QSIC 2006. Sixth International Conference on. Institute of Electrical and Electronics Engineers, Piscat- away, NJ, 360–368. DOI:http://dx.doi.org/10.1109/QSIC.2006.28
19. A.T. Chamillard and J.K. Joiner, "Using lab practica to evaluate programming ability". In Proceedings of the 32nd ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2001), ACM Press, pages 159-163, 2001.
20. Andrew Sutton and Bjarne Stroustrup, "Design of Concept Libraries for C++", Texas A&M University Department of Computer Science and Engineering {asutton, bs}@cse.tamu.edu
21. Stepanov, A., McJones, P.: Elements of Programming. Addison Wesley, Boston, Massachusetts (2009)