A Review of Techniques in Automatic Programming Assessment for Practical Skill Test

Adidah Lajis[1], Shahidatul Arfah Baharudin[1], Diyana Ab Kadir[1] , Nadilah Mohd Ralim[1], Haidawati Mohd Nasir[1] and Normaziah Abdul Aziz[2]
[1]*Universiti Kuala Lumpur Malaysian Institute of Information Technology*
[2]*Kulliyah of Information Technology International Islamic University Malaysia*
adidahl@unikl.edu.my

***Abstract*—Computer programming ability is a challenging competency that requires several cognitive skills and extensive practice. The increased number of students enrolled in computer and engineering courses and also the increased of failure and drop rate in programming subject is the motivational factor to this research. Due to the importance of this skill, this paper intends to study the landscape of current scenario in assisted assessment for hands-on practical programming focusing on competency-based assessment. The Bloom Taxonomy is used as a competency-based assessment platform. The review showed to-date that there are several automatic assessments for programming skills. However, there is no common grading being applied. Thus, further research is required to propose an automatic assessment that grades the student achievement based on learning taxonomy such as Bloom Cognitive Competency model.**

***Index Terms*—Cognitive Assessment; Assisted Assessment; Programming; Competency-Based;**

I. INTRODUCTION

Assessment is important for any educational organisation. It is a process to quantify the student skills and knowledge that can be implemented via various methods such as project, test, observation, assignment and others. Assessing computer programming competency is quite different from the assessment of other courses neither to math-related courses. In programming especially when focusing on practical assessment, every question can be solved in a variety of methods. It can be considered an individualised assessment, and this will consume a lot of time. The increased number of students in Higher Education may also increase the time spent by the lecturer in marking the assessment [14]. In Malaysia, the number of student enrollment for the year 2015 is 566,266 in public Higher Education Institutions(HEI) and 608,378 in private HEI [47].

Computer programming is the ability to produce working digital artefacts to the standards dictated by industrial best practice. Renumol et al. [2] quoted as "programming is the process of writing, testing and debugging of computer programs using different programming languages. The former is the knowledge of programming language syntax and semantics which in turn needs memorisation and comprehension abilities; whereas the latter is problem-solving and program design skills which in turn needs additional skills like abstraction and logical thinking, and domain knowledge". Therefore, it can be concluded that programming is a complex task that requires various skills and knowledge.

It is one of the common subjects taken by most of the students in higher education who enrol Information Technology, Computer Science and Engineering and is commonly known as highly practical subjects with the goal to develop students' understanding of the programming principles. The hard part of teaching computer programming is for beginners where they need to master the abstract concepts of programming. The drop-rate and failure rate for programming subject is relatively high [3] [4].

The assessment of computer programming is different from the assessment of math-related subjects [46]. It comprises of the high cognitive task, ranges from low level to high-level thinking skills. According to [43], a task to the program will need the skill to learn the language, create and comprehend new program, reuse and integrate programs, debugging and testing, and documenting what they code. The tasks performed during the programming process is a cognitive task that requires knowledge of syntax and semantics of the programming language [43]. Others cognitive task involve is to solve the problem, i.e. understand the problem, analyse and design the solution.

It has been identified that deficiencies in programming skills of first-year students are due to failure to recognise the main source of their difficulties [16]. Novice programmer often has difficulty in grasping the foundation level of programming concepts. In 2015, Parson et al. argued that students performed poorly in the assessment because the assessment is not testing their programming ability [5]. Some assessment methods are on paper and focus on assessment of the programming concept. The ability to solve programming problems and produce working code is considered the most important capability for computer science and engineering students. Besides, the process of manually validating student source code proves to be quite burdensome, and this may result in untimely reporting of feedback which also contributes to the high failure. A number of program assessment systems on different scales were produced over the last 15 years are either to assess the performance or the competency of the students. This paper studies the landscape of assisted assessment in hands-on practical programming focusing on cognitive competency based on Bloom taxonomy.

II. COMPETENCY-BASED EDUCATION AND BLOOM TAXONOMY

Competency-based education (CBE), the smaller concept of outcome-based education (OBE) is a measure of learning where student progress by demonstrating their competence while the educator guides them. Here competence is referring to the ability of the student to solve the problem. Competent students are those who can and want to, interact effectively

with three kinds of environments posed by the socially-ascribed, self-selected, and self-developed roles they face upon graduation [17]. Therefore, CBE is based on a set of outcome that is derived from an analysis of student task.

OBE has been implemented at all levels of tertiary education since the year 2008 [1]. It covers three learning domains; Psychomotor, cognitive, and effective domain and has been implemented in various modalities. Cognitive outcomes include a demonstrable acquisition of specific knowledge and skills in solving problems. An effective educational outcome is defined as learning outcomes that focus on "individual disposition, willingness, preferences, enjoyments ….." [6]. While psychomotor includes physical movement, coordination, and use of the motor-skill areas [18]. Evidence of the outcome is required to fulfil the shortage of the soft skill of an employee in the workplace [7] [8].

Bloom's (1956) Taxonomy of Educational Objectives relating to the cognitive domain has influenced many educationists over the years – more so than the companion volumes relating to the effective and psychomotor domains respectively [9] [19]. Bloom has defined six levels of cognitive domain: knowledge, comprehension, application, analysis, synthesis, and evaluation. Figure 1 shows the different level of Bloom Taxonomy with its behaviour.

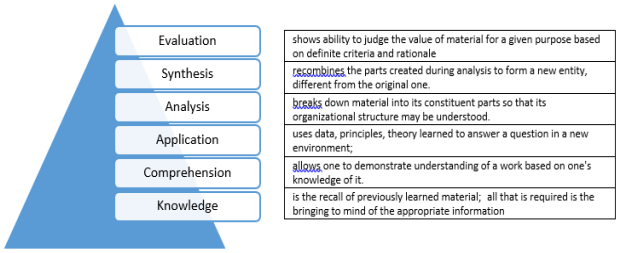

Figure 1: Bloom Taxonomy and its behaviour

This taxonomy has been taken as a basis for analysing the students' learning competence. It has been applied to structuring assessment for the computer science [20], to compare the difficulty of the cognitive level for computer science subjects [21] and also to plan of the assessment of programming [23]. In the year 2010, Alaoutinen and Smolander [50] also have studied a simple student self-assessment tool that uses Bloom's Revised Taxonomy as the base scale. This tool has help student in their learning and provides the teacher with the level of knowledge gained by the student. Thompson et al. [24] have discussed in detail in the cognitive domain in programming. The summary on the cognitive level, the programming assessment competence and the task in programming subject are shown in Table 1.

III. Reviews of Techniques in Automatic Programming Assessment

In general, there are two approaches to automate assessment in programming subjects; static and dynamic. Static approach checks and analyses the source code without executing the program [25] and being used to assess the programming style, syntax and semantic error, software metric analysis, structural similarity analysis, keyword detector, plagiarism detection and also diagram analysis. Dynamic analysis is an assessment based on the execution of the program. It is used to assess the programming errors, the design of the program, the software metrics and also to assess the style of programming. Further explanation can be found in the following topics.

Table 1
Cognitive Assessment for Programming Subjects

Cognitive Level	Competence [35]	Task in Programming [24]
Knowledge	Able to list related command or concepts	Identify a particular construct in the codes, recognise the implementation and recall any learning material learned earlier.
Comprehension	Able to explain what the command/concepts mean and able to apply an example similar problem	Able to translate an algorithm form one to another and to explain and present the concept
Application	Able to list cases when the command/concept can be used.	The algorithm and process is known and can apply to a familiar problem that has not been solved in the same data or context, or it is applied to an unfamiliar problem
Analysis	Able to explain the meaning of the command/concept in its context	The code is divided into parts and organise to achieve an objective. The critical component and unimportant component are identified.
Synthesis	Able to ensure the correct use of the command/concept.	Testing is performed to determine whether the code satisfies the requirements and able to suggest or produce better code in performing the task
Evaluation	Able to use the command/concept in problem-solving without an example.	Suggest a new algorithm or hypothesis to solve the problem

A. *Static Approach*

The semantic similarity-based approach is one of the static approaches being used to overcome drawbacks of the dynamic-based approach. The student program and the expert program are compared to calculate the semantic similarities. It evaluates how close a student's source code to an expert solution. However, it is not cost-effective when the size and the problem complexity increased, as it will consume higher processing time and memory requirement. Some examples of systems applying these approaches are FDA [55], ELP [12], SSBG [11] and AutoLEP [50], PETCHA [52].

Another static approached is the graph-based techniques. The code is represented as a graph with edges representing dependencies between different components of the program. The graph representation will provide abstract information that enables to assess the code quality by applying the software metrics. This approach has been applied in two different ways: graph transformation such as in [12][54] and graph similarity such as in [27][53].

Structural similarity analysis also uses this approach. The code is converted into pseudocode abstract. Pseudocode abstract is a representation of the basic algorithmic structure of the program. The student's abstract representation then is compared to the expert abstract representation [10].

However, in Non-structural similarity analysis, it is done by translating the students' and expert's code into the pseudo-codes, and they are compared to find similarity percentage [32].

Moodle extension developed by Slovak University of Technology Bratislava [51] evaluates the submission of an assignment by compilation and static analysis. It also compares the functionality of the program with a model.

Machine learning algorithm such as Support Vector Machines and the decision tree are also used to classify code properties that lead to error [42]. In 2009, MacNish [38] applied breath-first search, clustering and neural network to identify logic errors of the program. Later, Matloobi [37] applied fuzzy logic to grade algorithm complexity and meaningful comments. Latest, Srikant and Aggarwal [36], developed a system to grade a programming skill by this algorithm based on assessment rubrics. The system will provide two scores on program-ability and program practices. Automata is an example of one system that is based on the hybrid approach of semantic analysis and machine learning algorithm and incorporated a taxonomy indicating basic, advanced and edge [45] in its programming assessment.

Besides all the mentioned techniques, software quality metrics also being employed in the assessment. It can be raw metric or computed metric. Raw metrics are simple counts of things like lines of code and inheritance depth. Computed metrics take one or more raw metrics and combine them to form another measurement. Table 2 shows the software quality metrics employed in the analysis.

Table 2
Software Quality Metrics in Static Analysis Approach

Computed Metric	Raw Metrics
Typographic metrics [33,64]	Percentage of the following item; blank lines, average white space per line, names with good length, comment lines, characters in comments. Average characters per line and average identifier length.
Program complexity [33]	Reserved words, assignment statements, library and function calls, operators, loops and conditional statements, maximum depth of braces and brackets.
Program structure [33] [34][49]	Unused variables, re-declared variables, the variable used before set, used of value return by a function, unused statement, unused pointer, incorrect declaration of a variable, comments compared to some functions, valid variable declaration locally or globally and some denotations that should be declared as constant.
Comments [37]	Meaningful comments reflect the code.
Algorithms [37]	Some iterations, iteration, assignments, inline comments, and arrays.
McCabe's Cyclomatic Complexity [39]	Measures and controls the number of paths through a program
Halstead Complexity Measures [40]	Some unique operators, unique operands, the total number of operators and operands.
Reference Code Value [48]	It compares the CAM, LCOM3, RFC, and CC metrics of the assignments and reference code for deviations.

B. Dynamic Approaches

Dynamic analysis is the assessment based on executing the program used to assess style, programming errors, software metrics, and even design. On top of that, the assessment process can be done by looking into a code structure (white-box) or simply based on a functional behaviour of a program (black-box) [30]. It is the most well-known approach being employed in many programming assessment techniques [31]. Several systems apply this approach, such as Ceilidh[60], Mooshak [61], HoGG[62], PSGE [63] to name a few.

C. Hybrid Approaches

This approach is a combination of static and dynamic to improve and overcome the drawbacks of both approaches. Some systems that hybrid are PECHA[52], Scheme-Robe [58], WebBot[57] and Web-CAT[56]. More recent ones are AutoLEP [50] and Quimera[59].

IV. CONCLUSION

Several studies have applied Bloom for assessment but not focusing on hands-on or practical programming test. Bloom Taxonomy has been proved to help in to guide learning as it categorises thinking skills ranging from knowledge, the most basic skills up to creating, the highest thinking skills. It enables to identify the skill level of the student being assessed which thus help to improve his/her learning. The ability to solve programming problems and produce working code is very important, especially for computer and engineering students. With the increased number of students taking these courses, automated programming assessment helps to reduce the burden of manual assessment by the teachers, and at the same time able to improve the students' programming skills [64].

Based on the review focusing on the practical or hands-on assessment of programming subject, presently, most of the automatic assessment do not have a common grading model that refers to the learning taxonomy. This issue also has been argued by Caiza and De Alamo [65]. Therefore, further research must be done with the focus on the assessment of practical programming skill based on the learning taxonomy such as Bloom Cognitive Competency as a grading model.

ACKNOWLEDGEMENT

This research is supported by the Fundamental Research Grants Scheme (FRGS/1/2015/ICT02/UNIKL/02/2) financed by Ministry of Higher Education Malaysia.

REFERENCES

[1] Mohayidin, Mohd Ghazali (2008). "Implementation of Outcome-Based Education in Universiti Putra Malaysia: A Focus on Students' Learning Outcomes". International Education Studies. 1(4). Retrieved 23 October 2014.

[2] G, R.V., & Jayaprakash (2009). Classification of Cognitive Difficulties of Students to Learn Computer Programming.

[3] A. McGettrick program-ability, (2005). "Grand challenges in Computing: Education – A Summary", The Computer Journal Vol. 48

[4] A. Robins et al., (2003). "Learning and Teaching Programming: A Review and Discussion", Computer Science Education Journal, Vol. 13

[5] Parson, D., Wood, K. & Haden, Patricia, (2015). What are we doing when we assess programming?. Proceedings of the 17th Australasian Computing Education Conference (ACE 2015), Sydney, Australia, 27 - 30 January 2015

[6] Gronlund, N.E. (2000). How to write and use instructional objectives. Toronto: Prentice-Hall

[7] Clark, D. (2005) Softskills and E-learning. London: Epic Performance Improvement Limited.

[8] MacLeod, A. (2000). The importance of soft skills in the current Canadian labour market. Sectoral and Occupational Studies Division of Human Resources Development Canada, April.

[9] Krathwohl, D. R., Bloom, B. S. & Masia, B. B. (1964). Taxonomy of Educational Objectives. [Handbook 2: Affective Domain]. London: Longman

[10] Nghi Truong, Paul Roe, & Peter Bancroft. (2004). "Static Analysis of Students' Java Programs". Paper read at 6th Australian Computing Education

[11] T. Wang, X. Su, Y. Wang and P. Ma, (2007). " Semantic Similarity-Based Grading of Student Programs," Information and Software Technology, Vol. 49, No. 2, 2007, pp. 99-107. doi:10.1016/j.infsof.2006.03.001

[12] N. Truong, P. Bancroft and P. Roe, (2002). "ELP-A Web Environment for Learning to Program," Proceeding of the 19th Annual Conference of the Australasian Society for Computers in Learning in Tertiary Education, Vol. 19, Auckland, 8-11 December 2002, pp. 661-670.

[13] Khirulnizam Abd Rahman, Syarbaini Ahmad & Md Jan Nordin (2007). The Design of an Automated C Programming Assessment Using Pseudo-code Comparison Technique. National Conference on Software Engineering and Computer Systems 2007, organized by Universiti Malaysia Pahang, Pahang, Malaysia.

[14] Stephens D., Bull, J. E. and Wade, W. (2011). Computer-assisted assessment: suggested guideline for an institutional strategy. Online Learning. Part V: Online Assessment. London: Sage

[15] Aizyl Azlee (2016). Coding to be in school curricula next year, says MDEC CEO, MalayMail Online, Retrieved: 20 July 2017, http://www.themalaymailonline.com/malaysia/article/coding-to-be-in-school-curricula-next-year-says-mdec-ceo

[16] McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B.-D., Laxer, C., Thomas, L., Utting, I., and Wilusz, T., (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. SIGCSE Bulletin 33(4), 125–180

[17] James H. Block., (1978), 'The 'C' in CBE', Educational Researcher, 7(5), pp. 13-16.

[18] Simpson E.J. (1972). The Classification of Educational Objectives in the Psychomotor Domain. Washington, DC: Gryphon House.

[19] Harrow, A. (1972) A Taxonomy of Psychomotor Domain: A Guide for Developing Behavioral Objectives. New York: David McKay.

[20] Lister, R. and Leaney, J. (2003), Introductory programming, criterion-referencing, SIGCSE '03: Proceedings of the 34th SIGCSE technical symposium on Computer science education , 143-147, ACM Press

[21] Oliver, D., Dobele, T., Greber, M. and Roberts, T. (2004), This course has a Bloom Rating of 3.9. in Proceedings of the sixth conference on Australasian computing education - Volume 30 , Dunedin, New Zealand, 227-231, Australian Computer Society Inc

[22] Lister, R., Adams, E.S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J.E., Sanders, K., Seppälä, O., Simon, B. and Thomas, L. (2004) A multi-national study of reading and tracing skills in novice programmers. Inroads - The SIGCSE Bulletin, 36(4). 119-150.

[23] Shneider, E. and Gladkikh, O. (2006) Designing questioning strategies for information technology courses. Mann, S. and Bridgeman, N. eds. The 19th Annual Conference of the National Advisory Committee on Computing Qualifications: Preparing for the Future — Capitalising on IT , Wellington, 243-248, National Advisory Committee on Computing Qualifications.

[24] Thompson, E., Luxton-Reilly, A., Whalley, J. L., Hu, M. and Robbins, P. (2008), Bloom's taxonomy for CS assessment. Proceedings of the Tenth Conference Australasian Computing Education (Wollongong, NSW, Australia, January 20-23, 2008), 155-161.

[25] Ala-Mutka, K. M. (2005). A survey of automated assessment approaches for programming assignments. Computer science education, 15(2), 83-102.

[26] N. Truong, P. Roe and P. Bancroft, (2004), "Static Analysis of Students' Java Programs," Proceedings of the 6th Conference on Australasian Computing Education, Vol. 30, p. 325.

[27] K. A. Naude, J. H. Greyling and D. Vogts, (2010), "Marking Student Programs Using Graph Similarity," Computers & Education, Vol. 54, No. 2, pp. 545-561

[28] Shuhida, Shuhida, Hamilton, M., D'Souza, D. (2009). A Taxonomic Study of Novice Programming Summative Assessment, Eleventh Australasian Computing Education Conference (ACE2009), Wellington, New Zealand, January 2009

[29] Alaoutinen, S, & Smolander, K. (2010). Student self-assessment in a programming course using bloom's revised taxonomy. In Proceedings of the fifteenth annual conference on Innovation and technology in computer science education (pp. 155-159). ACM, New York, NY, USA.

[30] 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

[31] Fatima Al Shamsi, Ashraf Elnagar (2012). An Intelligent Assessment Tool for Students' Java Submissions in Introductory Programming Courses. Journal of Intelligent Learning Systems and Applications, 2012, 4, 59-69

[32] Rahman, K. A., & Nordin, M. J. (2007). A review on the static analysis approach in the automated programming assessment systems.

[33] Athanasios Tsintsifas. (2002). A Framework for the Computer Based Assessment of Diagram-Based Coursework. PhD thesis.

[34] Michael Blumenstein, Steve Green, Ann Nguyen, and Vallipuram Muthukkumarasamy, (2004), GAME: A generic automated marking environment for programming assessment. In Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04) Volume 2, pages 212–, Washington, DC, USA, 2004. IEEE Computer Society.

[35] Alaoutinen, S, & Smolander, K. (2010). Student self-assessment in a programming course using bloom's revised taxonomy. In Proceedings of the fifteenth annual conference on Innovation and technology in computer science education (pp. 155-159). ACM, New York, NY, USA

[36] Srikant, S., and Aggarwal, V. (2014). A system to grade computer programming skills using machine learning. In Proc. 20th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining (Aug. 2014), 1887–1896

[37] Roozbeh Matloobi, Michael Myer Blumenstein, and Steven Green. (2009), Extensions to generic automated marking environment: Game-2+.

[38] Cara MacNish (2007). Longitudinal Syntactic Analysis of Laboratory Submission for Examining Problem – Solving Behavior, Advanced Learning Technologies , 2007, ICALT 2007, Seventh IEEE International Conference.

[39] Susan A. Mengel and Vinay Yerramilli. (1999). A case study of the static analysis of the quality of novice student programs. SIGCSE Bull., 31:78–82, March 1999

[40] Verifysoft Technology GmbH. Verifysoft (2010), Halstead metrics. http://www.verifysoft.com/en_halstead_metrics.html, June 2010.

[41] Deek, F.P., & McHugh, J. (2000), Problem-solving methodologies and the development of critical thinking skills. Journal of Computer Science Education, 14(1-2), 6-12.

[42] Brun, Y., & Ernst, M. D. (2004, May). Finding latent code errors via machine learning over program executions. In Proceedings of the 26th International Conference on Software Engineering (pp. 480-490). IEEE Computer Society.

[43] Deek F. P., McHugh J. A. (1998), "A survey and critical analysis of tools for learning programming". Computer Science Education, Vol.8, n°2, p. 130-178

[44] Secretary's Commission on Achieving Necessary Skills (SCANS). What work requires of schools A SCANS Report for America (2000). Washington DC: US Labor Department. http://wdr.doleta.gov/SCANS/whatwork/whatwork.pdf.

[45] ASPIRINGMINDS (2017). AUTOMATA. Retrieved 23 July 2017 at http://www.aspiringminds.com/technology/automata

[46] Brusilovsky, P. & Sosnovsky, S. (2005). Individualized Exercises for Self-Assessment of Programming Knowledge: An Evaluation of QuizPACK, Journal on Educational Resources in Computing (JERIC), 5(3), Article 6

[47] Ministry Higher Education (2016). Malaysia Higher Education Blueprint 2015-2025, available at https://www.acu.ac.uk/events/perspectives/datin-siti-hamisah-presentation, 21 September 2017

[48] Koyya, P., Lee, Y, & Yang, J. (2013). Feedback for Programming Assignments Using Software-Metrics and Reference Code, ISRN Software Engineering Volume 2013 (2013), Article ID 805963, 8 pages

[49] 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.

[50] Wang, T., Su, X., Ma, P., Wang, Y., Wang, K. (2011). Ability-training-oriented Automated Assessment in Introductory Programming Course. Computer. Education, Elsevier, vol. 56, pp. 220-226

[51] Jelemenská, K. Čičák, (2012). Improved Assignments Management in MOODLE Environment. INTED2012 Proceedings, pp. 1809-1817.

[52] 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

[53] Milena Vujoˇsevi´c-Janiˇci´c · Mladen Nikoli´c · Duˇsan Toˇsi´c · Viktor Kuncak (2012). Software Verification and Graph Similarity for

Automated Evaluation of Students' Assignments. Information and Software Technology Volume 55, Issue 6, June 2013, Pages 1004-1016

[54] Rivers, K. & Kenneth R. Koedinger (2013). Automatic Generation of Programming Feedback: A Data-Driven Approach, AIED 2013 Workshops Proceedings Volume 9

[55] Fonte, D., Daniela da Cruz, Alda Lopes Gancarski, && Henriques, P. R. (2013). A Flexible Dynamic System for Automatic Grading of Programming Exercise. 2nd Symposium on Languages, Application and Technologies (SLATE 13)

[56] Anuj Shah. (2003), Web-CAT: A Web-based Center for Automated Testing. Technical report, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, 2003

[57] Don Colton, Leslie Fife, and Andrew Thompson. (2006), A Web-based Automatic Program Grader. Information Systems Education Journal (ISEDJ), 4(114), November 2006.

[58] Riku Saikkonen, Lauri Malmi, and Ari Korhonen. (2001), Fully automatic assessment of programming exercises. In Proceedings of the 6th annual conference on Innovation and technology in computer science education, ITiCSE '01, pages 133–136, New York, USA, 2001. ACM

[59] Daniela Fonte, Ismael Vilas Boas, Daniela da Cruz, Alda Lopes Gançarski, and Pedro Rangel Henriques. (2012), Program analysis and evaluation using quimera. In ICEIS'2012 — 14th International Conference on Enterprise Information Systems, pages 209–219. INSTICC, June 2012.

[60] S D Benford, E K Burke, E Foxley, and C A Higgins. (1995), The Ceilidh system for the automatic grading of students on programming courses. In Proceedings of the 33rd annual on Southeast regional conference, ACM-SE 33, pages 176–182, New York, NY, USA, 1995. ACM.

[61] José Paulo Leal and Fernando Silva. (2003), Mooshak: aWeb-based multi-site programming contest system. Software: Practice and Experience, 33(6):567–581, May 2003.

[62] D.S. Morris. (2002), Automatically grading Java programming assignments via reflection, inheritance, and regular expressions. In Frontiers in Education, 2002. FIE 2002. 32nd Annual, volume 1, pages T3G–22, 2002.

[63] Edward L. Jones. (2000), Grading student programs - a software testing approach. In Proceedings of the second annual CCSC on Computing in Small Colleges Northwestern conference, pages 185–192, USA, 2000. Consortium for Computing Sciences in Colleges.

[64] Emma Enstr¨om, Gunnar Kreitz, Fredrik Niemela, Pehr Soderman, and Viggo Kann. (2011), Five Years with Kattis – Using an Automated Assessment System in Teaching. In Frontiers in Education Conference (FIE), 2011. Institute of Electrical and Electronics Engineers, Piscataway, NJ, T3J–1.

[65] Caiza, J. C., & Ramiro, Á. (2013). Programming assignments automatic grading: review of tools and implementations