ICITKM
Proceedings of the First International Conference on Information Technology and Knowledge Management pp. 65–70
DOI: 10.15439/2018KM31
ISSN 2300-5963 ACSIS, Vol. 14

E-Assessment Tools for Programming Languages: A Review

Sugandha Gupta[1], Anamika Gupta[2]
[1]*Sri Guru Gobind Singh College of Commerce, University of Delhi*
[2]*Shaheed Sukhdev College of Business Studies, University of Delhi*
[1]*sugandhagupta@sggscc.ac.in*, [2]*anamikargupta@sscbsdu.ac.in*

***Abstract*—Continuous Evaluation and feedback not only helps in improving learning of a student, but also acts as a constant motivator to put in more efforts. But then, feedback and assessment are very difficult and time consuming in practice. Thus, automating the entire system of assessment, evaluation and feedback will be highly beneficial. But, building such tools for all courses is yet not feasible. However, e-assessment tools for programming courses in Computer Science discipline can be developed. In this paper, we review various grading techniques used by these tools to assess a student's programming assignment. Further, this paper discusses various types and features of tools according to which an appropriate tool should be selected. And, in the end, we will be highlighting the extent to which students and instructors are actually benefited by these tools.**

***Index Terms*—E-Assessment; Automated Tool; Programming Languages; Static Analysis; Dynamic Analysis; LMS Based Tools**

I. Introduction

ASSESSMENT plays an integral role in any educational system. It helps in assessing the knowledge acquired by a learner at any point of time [1, 8]. However, with substantial increase in number of students getting enrolled at Universities in either conventional or distance learning environment, usual ways of assessing a student are becoming inadequate in terms of both time and effort [5].
Thus, the notion of e-Assessment was introduced to overcome all the inadequacies of traditional pen and paper assessment modes. The accessibility to automated tools [8, 9] to assist the evaluation of students' work and providing students with appropriate and timely feedback can really help in motivating the student to learn with more focus [1, 24]. The feedback provided by the tool can also be further enhanced by using some techniques to direct a student to engage in self learning aiming at improvement

Although, building such tools for evaluating students' work for all kind of courses is not yet feasible. However, courses that involve formal language of expression i.e. where every instruction can be expressed in terms of a language can be automated, such as, programming courses [8, 36] in Computer Sciencediscipline.

Programming is the core of Computer Science discipline. But, today programming [36] is not restricted to only Computer Science students. It is becoming an important part even in various other academic disciplines. The best way to learn a programming language is to practice a large set of problems. The skills will be improved through practicing many programming exercises. As a student, enormous satisfaction is attained if all the practiced problems are evaluated by some expert highlighting all insights. Getting a detailed and timely feedback serves as a motivating force for a student. This will not only help them to understand their mistakes, but also to improve their skills. But as an evaluator, it's very difficult to evaluate each and every problem for all students. Thus, building an E-assessment tool to assess a student's program and providing an appropriate feedback is much needed [1, 3].

The purpose of an e-assessment tool [8, 10] for programming languages is to improve programming skills in the students, paying exceptional attention to beginner students [1, 15]. For a student to practice more and more problems, immediate and thorough feedback plays an important role.

Both instructors and students are benefitted by these tools. As of instructor's point of sight, assessment tools can be used to automate mundane tasks that incur a lot oftime and effort manually [5]. For example, using these tools, an instructor can automatically correct assignments, grade students and provide a timely feedback [1, 3, 8].Also, the assessment results are trustworthy and much easier to comprehend [3].

As of the students' point of sight, the timely feedback imparted by the tools push them to practice more and more programming problems.Thus, their programming skills are also improved. Also, students are motivated to constantly improvise their assignments by re-submissions until an appropriate solution is obtained [4, 9].

In this paper, we focus on reviewing certain e-assessment tools available for grading a programming assignment. Further, we also discuss various grading techniques and classification schemes for a tool. Then in the end of this paper, we discuss the effectiveness of e-assessment tools with respect to students and instructors.

This paper is structured into five sections. Section II discusses main approaches used by a tool to assess a program. Section III reviews categorization of tools according to certain parameters. Section IV mentions few readily used e-as-

sessment tools. Section V discusses research questions that helps to identify how important are e-assessment tools to students and instructors.

II. Main Approaches For Assessment

For automatic assessment of a programming code, two major approaches have been devised i.e. Static Analysis and Dynamic Analysis [2, 5].

(1) Static Analysis

It is performed by inspecting the source code of the program without executing it. In this method, structure of the program and content are observed [2, 3]. In the old days, this was the only means to assess a programming code. Students used to submittheir programming assignment code, and teachers based their assessment on examining the code without actually executing it.ASSYST [13], CAP [15] and Expresso [7, 14] are few examples of system that only use static analysis.Within this approach, following methods have been characterized [4]:

- Programming-style analysis: According to this, a quality program is the one which is highly readable. For this, parameters like expressive variables names, use of constants, line spacing, indentation, comments, lesser global data etc are evaluated [3].
- Semantic-errordiscovery: These errors are discovered when a statement is syntactically correct, but leads to failure in program execution. Most common semantic errors for a beginner in programming are division by zero, infinite loops and terminating a loop header. Some semantic errors can even result in crashing of the system [2].
- Software metrics analysis: It is a function that produces a single output that can be interpreted as the measure of software quality. Metrics cover the size of program in terms of number of lines of code, cyclomatic complexityof the code and quality [2, 7].
- Keyword analysis: Instructor gives a list of keywords which he/she is looking for in the solution. This parameter deals with finding these keywords in the program to be assessed and matching it with those specified by the instructor [2, 5].

(2) Dynamic Analysis

Under this approach, a program is executed against many different test cases. And, the output obtained is equated against the anticipated output. [3, 6] If they are same, then the program is considered to be correct. TRY [7, 16] is an example of system that performs only dynamic analysis.
Within dynamic analysis [3, 5, 6], two different methods of evaluating a program are available:

Black-Box approach: The entire program is executed, and its output is examined against different test cases. And, then the program is considered as "correct" or "incorrect" based upon match or mismatch [2, 3]. Under this approach, the entire program is treated as an atomic entity.

Grey-Box approach: Under this approach, a program is divided into a set of functions. And, then each and every function is assessed separately. The final grading of the program is obtained by the amalgamation of the partial grades of these functions. Using this approach, a program with some function with error can also be graded [2].

In dynamic analysis, selection of test case against which a program will be executed is an important decision. Normally, the evaluator submits the test cases with the problem itself.

(3) Static Analysis v/s Dynamic Analysis

A program with syntax errorslike a omitted semi-colon, mis-match in number of opening and closing parenthesis, etc. prevents the program to compile. And thus, such a program can't be executed using dynamic analysis. In this case, a grade of zero is awarded to the program. On the other hand, a program is assessed even when it fails to execute in case of static analysis [3, 6].

In case of dynamic analysis, student just gets to know whether the program executed successfully and anticipated output is produced or not. So, it is the student's responsibility to find out all the errors and fix them. Whereas in static analysis, student gets to know about all the errors present in the program [2, 3].

Dynamic analysis approach is not suitable for the set of programming problems which can have multiple solutions, because to assess such a code, instructor will have to submit an exhaustive set of test cases. This limitation is not applicable for static analysis approach.

There are some tools which combine both the strategies to grade a program. AutoLEP [10] and Auto Grader [19] are tools which combines static and dynamic analysis approach to grade a program.

III. Tools Review

Three classification schemes [5] have been identified for the tools: (i) assessment-type, (ii) approach, and (iii) specialty.

(1) Classi cation by Assessment-Type:Under this, the assessments tools are categorized in three types based upon how the assessment process is carried:

1. Manual Assessment: Assessment of the programming assignment is done "manually by the instructor with assistance of the tool"[3].
2. Automatic Assessment: Assessment of a programming assignment is done automatically by the tool.But, the instructor will have to clearly state the parameters on which the program code should be assessed before the assessment process[3, 5].
3. Semi-Automatic Assessment: Assessment is performed automatically by the tool, but manual inspection of

the programming code is required by the instructor [3].

(2) Classi cation by Approach:Under this, three approaches were identified based on howthe assessment process is commenced.

1. Instructor Centered Approach: After the submissions of programming code by student, assessment process is initiated by the instructor. The code is assessed by the tool and results are forwarded to the instructor for review. After review, final results are formulated by the instructor and given to students.

- Student Centered Approach: The assessment process is started by the student. Programming code submitted by the student should abide certain specifications stated by the instructor, which will be used by the tool for assessment.After the completion of the assessment process, assessment results are forwarded to both the instructor and students [5].
- Hybrid Approach: These tools incorporatestrong points of both the abovementioned approaches. Under this approach, three different strategies were identified:
  –Preliminary validation [5]: Instructor starts the assessment process, but a preliminary evaluation of the solution is carried out by the tool.BOSS [9], OTO [30] and PASS [7] are few tools that implement this strategy.
  –Partial feedback [5]: Tool assesses the code partially. Instructor manually evaluates the final code. WebCAT [8] uses this strategy for assessment of programming codes.
  -Instructor review [5]: A feedback is provided to the student after evaluation of the code by the tool. But, that feedback is not final, as the result of tool's assessment is accessible and updatable by the instructor. The instructor's decision would be final and would be intimated to the student. MOOSHAK [11] implements this approach.

(3) Classi cation by Specialization:Under this, the assessment tools are classified based on their ability to perform extra functions other than assessment of a programming code.

- Tools Specialized in Competitions [5]: The assessment tool is used to evaluate codes of all students using specific test cases. As a result of the evaluation process, it specifies out of all the codes, which all codes are acceptable. ONLINE JUDGE [22, 23]implements this strategy.
- Tools Specialized in Quizzes [5]: A set of questions are formulated which should be answered by the students. And based on the result of evaluation of theses answers, winners are selected. This strategy is also used in recruitments.AutoLEP [10]uses this approach for assessment.
- Tools Specialized in Software-Testing [5]: These tools are used to automate the testing process, thus reducing the dependence on manual testing. Prog Test [17] is used for software testing purposes.

A fourth classification is also possible, i.e. classification on the basis of usage. An e-assessment tool can either be used as a standalone tool, or in integration with some IDE or Learning Management System (LMS). A standalone e-assessment tool can be deployed on either instructor's or student's system and can be used for grading. Examples of tools that fall in this category are AutoLEP [10] and INGInious [20]. An IDE integrated e-assessment tool can be used with an IDE specific to a programming language. WebCAT [8] and Petcha [18] are integrated with IDE to grade programming assignments in java. A LMS integrated e-assessment tool is used in integration with a LMS (majorly edX, Audacity, Coursera) to grade programming assignments.

IV. Description of Tools

Based on the specifications and characteristics discussed in the previous two sections, below are few readily used E-assessment tools:

(1) WebCAT:Edwards and Perez-Quinones developed Web-CAT to assess student's code and test cases [8]. The students need to submit their own test cases with their codes, enabling them to demonstrate the correctness and validity of their own programs. It uses dynamic analysis and partial Feedback [5].

*(2) BOSS:*It is a tool developed by Luck & Joy to assist the online submission and successive processing of programming assignments. This tool can be used by in two individuals: first, by students to obtain the feedback for their submitted programs. Second, tool can be used by instructors in order to grade student submissions [9]. It uses static analysis and preliminary validation [5] (Hybrid Approach).

(3) AutoLEP: AutoLEP [10] is a standalone tool that uses a combination of static and dynamic analysis approach to grade the programming assignments. That is, it not only evaluates the output of the program but also deals with the construct of the program.

*(4) Mooshak:*It is a "web-based learning system" initially designed to conduct programming contests over the Internet. It provides specific interface to every user according to its profile, i.e. it will be different for student administrator, user, guest user, instructor and student [11].

(5) CourseMarker: CourseMarker was developed at the University of Nottingham as a successor to the Ceilidh in 2003. Students use CourseMarker client on their system and login into their account. They select the course, topic and exercise they want to complete. Students write the program in response to the problem description provided to them and submit it. Program's output is matchedagainstthe expected output of the program and results are shown to students. Students can re-solve the problem if it is not satisfactory, upon teacher's permission [12].

(6) ASSYST: It practices two user views: one corresponding to student and other corresponding to the instructor. Using the student view, a student submits a programming code

for subsequent grading; and using the instructor view, instructor guides the assessment process [13].

(7) Expresso: This tool is used for an introductory Java programming courseand it processes a program in multiple passes. The first pass accepts programmer's codeas an input and removes comments and stores the resulting characters in a vector. In the second pass, white spaces are removed and the file is tokenized, and the result is stored as a vector of words. Punctuation and white space are used as delimiters to identify words. Mistakes are detected (if any) and the appropriate error messages are printed in the final pass. These messages can be used to fix the program[14].

(8)CAP: The "Code Analyzer for Pascal (CAP)" [15] analyzes syntactical, style and logical errors of a program and gives a feedback to the student stating the same. The feedback indicates the problem with the code and a measure to correct that.

(9) TRY: TRY [16] system was developed for Unix operating system which test student program with a set of hidden test data.

(10) Prog Test: It is a web-based environment for imparting a feedback to the student after successful assessment of the programming assignment submitted by the student. Prog Test evaluates a student's submission on basis of: Instructor's code and test cases, Student's code and test cases. A student has to submit program as well as test cases used by him to test his own program. The instructor's program is asignificant parameter used for assessing the student's program. [17].

(11) Petcha: It acts as an "automated Teaching Assistant in computer programming courses", helping students to learn programming more efficiently.Also, it imparts feedback to the student for the solutions of programming assignments submitted by the student [18].

(12) Auto Grader: It is a new tool designed at MIT to review faulty code and automatically provide feedback on how to fix it. It uses Error Models and Constraint-based synthesis to perform this [19].

(13)INGInious: INGInious [20] is a tool designed to automatically grade programming code submitted by the user. It can virtually cater a large set of programming languages

(14) Curator: It grades a program by a strict textual comparison of the solution provided by the instructor and the solution submitted by the student[21].

(15) Online Judge: It also textually compares the code submitted by the student and the instructor. It is used to grade programs in various languages (C, C++, Java). It is also for conducting quizzes.

(16) HackerEarth Recruit: It is an online skill assessment tool for conducting programming tests to assess developers. It saves the pain of going through hundreds of resumes, by automating the process of evaluating technical skills, which helps to quickly filter the competent candidates.

(17) LMS Based Tools: This category of tools can be used to assess a student in the course enrolled on the basis of his/her submissions during the course. Further, it can be used to detect plagiarisms, generate and evaluate test cases for a program.

V. Are E-Assessment Tools Useful In Programming Courses

Four research questions were formulated to measure the degree to which e-assessment tools have helped to students and instructors [25]:

RQ1. Have e-assessment tools laid a positive impact on student learning?
RQ2.According to students, have e-assessment tools improved their performance?
RQ3According to instructors, have e-assessment tools improved their teaching experience?
RQ4. Is the result obtained by e-assessment tools precise and useful?

RQ1. Have e-assessment Tools laid a positive impact on student learning?

In 2003, Edwards [26] presented fascinating results when he changed the e-assessment tool in a junior level course on comparative languages, i.e. Curator was replaced by Web-CAT, demonstrating more timely submission of assignments along with test cases by the students. In 2003, Woit [27] collected data of students for "five consecutive years" comparing their performance on online tests with and without e-assessment tools. He concluded that online assessment gave a more precise indication of student knowledge. In 2005, Higgins [28] conducted an experiment in which "CourseMarker substituted Ceilidh at the University of Nottingham", thus increasing the passing percentage of students. Also in 2005, "Malmi [30] showed results from students using TRAKLA and TRAKLA2", in which final exam grades improved when students were allowed to resubmit their work. In 2011, Wang [31] showed that final grades of students using AutoLEP for grading were way better than grades produced without using any tool.

Considering all these facts, a positive impact was inferred with use of e-assessment tools on student performance. End-of-grades or final exam scores were major measures used to measure this.

RQ2. According to students, have e-assessment tools improved their performance?

In 2003, Edwards [26] proved that using WebCAT laid a positive impact using a 20-question survey for students. In 2005, Higgins [27] proved that over 75% of students' loved the flexibility to re-submit a programming assignment due to use of an e-assessment tool by carrying a survey to test the tool CourseMarker and indicated that. In 2009, Garcia-Mateos [32] presented Mooshak, and handed over the students a survey asking agreement or disagreement with re-

spect to the tool. 77% of the students specified that “they learn better with the new methodology than with the old one,” while 91% said that “if they could choose, they would follow the continuous evaluation methodology again.” Also in 2012, Brown [33] presented a surveyon student’s opinion about the impact of using JUG automated assessment tool. Given the question “Did the automatically graded tests match your beliefs of the requirements?” the major chunk of students opted for the answer, “Sometimes.” But the question “did the reports from the automatic grader clarify how your code shouldbehave?” urged to students answering “Often.”

Unconvincing results concerning student perceptions of e-assessment tools were observed. Students had a mixed reaction on this question; but a significant number of studentsindicatedtheir dissatisfaction with the tools.

RQ3 According to instructors, have e-assessment tools improved their teaching experience?

In 1995, Schorsch [35] stated that “6 out of 12 instructors who used CAP” for grading assignments indicated that the tool helped them saving around ten hours effort incurred in grading a section earlier.In 2003, Venables [35] used SUBMIT (e-assessment tool) and stated that the feedback imparted by the tool was successful to provide answers to doubts of students which they had while solving the assignment. This feature of the tool helped by saving onto the class time that otherwise would have been spent in responding to students’ questions. In 2012, Queirós [36] claimed that “automated grading is better than manual grading in terms of efficiency, accuracy, and objectivity” as e-assessment tools eradicatefavoritisms and other biased factors from the grading process, and submissions are noticeableat a greater pace.

Majority of the instructors have reported that initially a substantial amount of their time is incurred in learning the tool and making the students familiar with the tool, but once this has been done the tool saves a lot of their time which was earlier used for grading student’s code and giving them feedback.

RQ4. Is the result obtained by e-assessment tools precise and useful?

In 2005, Higgins [37] stated that grading performed by CourseMarker tool in one section of a course was at par with the assessment done by a teaching assistant in some other section of same course. Also in 2012, Taherkhani [38] revealed that AARI (e-assessment tool) was successful at recognizing the algorithms used by the students to perform sorting on integers for about 75% of the submissions.Further in 2014, Gaudencio [39] stated that instructors who manually graded the assignments also tend to agree more with the feedback provided by the tool in comparison to assessment provided by other instructors.

Thus, it can be inferred that e-assessment tools assist the instructors in conducting the assessment process in a positive manner.

References

[1] Matthíasdóttir, Ásrún & Arnalds, Hallgrímur. (2016). e-assessment: students' point of view. 369-374

[2] S. M. Arifi, I. N. Abdellah, A. Zahi and R. Benabbou, "Automatic program assessment using static and dynamic analysis," 2015 Third World Conference on Complex Systems (WCCS), Marrakech, 2015, pp. 1-6.

[3] Kirsti M Ala-Mutka (2005),’ A Survey of Automated Assessment’,Approaches for Programming Assignments, Computer Science Education, 15:2, 83-102

[4] Rahman, Khirulnizam Abd, and Md Jan Nordin. "A review on the static analysis approach in the automated programming assessment systems." In Proceedings of the national conference on programming, vol. 7. 2007.

[5] D. M. Souza, K. R. Felizardo and E. F. Barbosa, "A Systematic Literature Review of Assessment Tools for Programming Assignments," 2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET), Dallas, TX, 2016, pp. 147-156.

[6] Fonte, Daniela, Daniela da Cruz, Alda Lopes Gançarski, and Pedro Rangel Henriques. "A Flexible Dynamic System for Automatic Grading of Programming Exercises." In OASIcs-OpenAccess Series in Informatics, vol. 29. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.

[7] Truong, Roe, Bancroft, “Static Analysis of Students’ Java Programs”, Sixth Australian Computing Education, Conference (ACE2004), Dunedin, New Zealand, Vol. 30.

[8] S. H. Edwards and M. A. Perez-Quinones. Web-CAT: automatically grading programming assignments. In Proc. Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE), pages 328–328, 2008.

[9] Mike Joy, Nathan Griffiths, and Russell Boyatt. 2005. The boss online submission and assessment system. J. Educ. Resour. Comput. 5, 3, Article 2 (September 2005).

[10] W. Tiantian, S. Xiaohong, M. Peijun, W. Yuying and W. Kuanquan, "AutoLEP: An Automated Learning and Examination System for Programming and its Application in Programming Course," 2009 First International Workshop on Education Technology and Computer Science, Wuhan, Hubei, 2009, pp. 43-46.

[11] José Luis Fernández Alemán, ‘Automated Assessment in a Programming Tools Course IEEE TRANSACTIONS ON EDUCATION, VOL. 54, NO. 4, NOVEMBER 2011

[12] Higgins, C., Hergazy, T., Symeonidis, P., and Tsinsifas, A. The CourseMarker CBA System: Improvements over Ceilidh, Education and Information Technologies, 8(3), 2003, pp. 287– 30.

[13] David Jackson and Michelle Usher. 1997. Grading student programs using ASSYST. In Proceedings of the twenty-eighth SIGCSE technical symposium on Computer science education (SIGCSE '97), James E. Miller (Ed.). ACM, New York, NY, USA, 335-339.

[14] Maria Hristova, Ananya Misra, Megan Rutter, and Rebecca Mercuri. 2003. Identifying and correcting Java programming errors for introductory computer science students. In Proceedings of the 34th SIGCSE technical symposium on Computer science education (SIGCSE '03). ACM, New York, NY, USA, 153-156.

[15] Tom Schorsch. 1995. CAP: an automated self-assessment tool to check Pascal programs for syntax, logic and style errors. In Proceedings of the twenty-sixth SIGCSE technical symposium on Computer science education (SIGCSE '95), Curt M. White, James E. Miller, and Judy Gersting (Eds.). ACM, New York, NY, USA, 168-172.

[16] Yusofa,, Zinb , Adnana, ‘Java Programming Assessment Tool for Assignment Module in Moodle E-learning System’, International Conference on Teaching and Learning in Higher Education (ICTLHE2012) in conjunction with RCEE & RHED 2012, 1877-0428 © 2012 Published by Elsevier Ltd. Selection and/or peer-review under

responsibility of Centre of Engineering Education, Universiti Teknologi Malaysia.

[17] D. M. de Souza,J. C. Maldonado and E. F. Barbosa PROGTEST: An Environment for the Submission and Evaluation of Programming Assignments based on Testing Activities. Software Engineering Education and Training (CSEET), 2011 24th IEEE-CS Conference.

[18] Ricardo Alexandre Peixoto Queirós and José Paulo Leal. 2012. PETCHA: a programming exercises teaching assistant. In Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education (ITiCSE '12). ACM, New York, NY, USA, 192-197

[19] Auto Grader: http://www.csail.mit.edu/node/1886

[20] Guillaume Derval, Anthony Gego, Pierre Reinbold, Benjamin Frantzen and Peter Van Roy. Automatic grading of programming exercises in a MOOC using the INGInious platform. Proceedings of the European MOOC Stakeholder Summit 2015.

[21] Virginia Polytechnic Institute and State University. Curator: an electronic submission management environment. http://ei.cs.vt.edu/~eags/Curator.html.

[22] A. Kurnia, A. Lim, and B. Cheang. OnLine Judge. Computers & Education, 36(4):299–315, May 2001.

[23] B. Cheang, A. Kurnia, A. Lim, and W.-C. Oon. On automated grading of programming assignments in an academic institution. Computers & Education, 41(2):121–131, September 2003

[24] C. Douce, D. Livingstone, and J. Orwell, "Automatic test-based assessment of programming: A review," J. Educ. Resources Comput., vol. 5, no. 3, pp. 1–13, 2005.

[25] Pettit, R. S., & Homer, J. D., & Holcomb, K. M., & Simone, N., & Mengel, S. A. (2015, June), Are Automated Assessment Tools Helpful in Programming Courses? Paper presented at 2015 ASEE Annual Conference & Exposition, Seattle, Washington. 10.18260/p.23569

[26] Stephen H. Edwards. 2003. Improving student performance by evaluating how well students test their own programs. J. Educ. Resour. Comput. 3, 3, Article 1 (September 2003).

[27] Denise Woit and David Mason. 2003. Effectiveness of online assessment. In Proceedings of the 34th SIGCSE technical symposium on Computer science education (SIGCSE '03). ACM, New York, NY, USA, 137-141.

[28] Colin A. Higgins, Geoffrey Gray, Pavlos Symeonidis, and Athanasios Tsintsifas. 2005. Automated assessment and experiences of teaching programming. J. Educ. Resour. Comput. 5, 3, Article 5 (September 2005).

[29] Amruth N. Kumar. 2005. Generation of problems, answers, grade, and feedback---case study of a fully automated tutor. J. Educ. Resour. Comput. 5, 3, Article 3 (September 2005).

[30] Lauri Malmi, Ville Karavirta, Ari Korhonen, and Jussi Nikander. 2005. Experiences on automatically assessed algorithm simulation exercises with different resubmission policies. J. Educ. Resour. Comput. 5, 3, Article 7 (September 2005).

[31] Tiantian Wang, Xiaohong Su, Peijun Ma, Yuying Wang, and Kuanquan Wang. 2011. Ability-training-oriented automated assessment in introductory programming course. Comput. Educ. 56, 1 (January 2011), 220-226.

[32] García-Mateos, Ginés & Fernández-Alemán, José. (2009). A Course on Algorithms and Data Structures Using On-line Judging ABSTRACT. ACM SIGCSE Bulletin. 41. 45-49.

[33] Christopher Brown, Robert Pastel, Bill Siever, and John Earnest., JUG: a JUnit generation, time complexity analysis and reporting tool to streamline grading, 17th ACM annual conference on Innovation and technology in computer science education (ITiCSE '12).

[34] Tom Schorsch, Cap: An Automated Self-Assessment Tool To Check Pascal Programs For Syntax, Logic And Style Errors, SIGSCE 1995.

[35] Anne Venables and Liz Haywood. 2003. Programming students NEED instant feedback, Fifth Australasian conference on Computing education - Volume 20 (ACE '03), Tony Greening and Raymond Lister (Eds.), Vol. 20

[36] Queirós, R., & Leal, J. P., "Programming exercises evaluation systems: An interoperability survey", 4th international conference on computer supported education, 2012, (pp. 83–90).

[37] Colin A. Higgins, Geoffery Gray, Pavlos Symeonidis, and Athanasios Tsintsifas., Automated assessment and experiences of teaching programming. Journal on Educational Resources in Computing (JERIC), 2005.

[38] Ahmad Taherkhani, Ari Korhonen, and Lauri Malmi. 2012. Automatic recognition of students' sorting algorithm implementations in a data structures and algorithms course, 12th Koli Calling International Conference on Computing Education Research.

[39] Matheus Gaudencio, Ayla Dantas, and Dalton D. S. Guerrero. Can Computers Compare Student Code Solutions as Well as Teachers,SIGCSE.