Computers & Education 56 (2011) 220–226

Contents lists available at ScienceDirect

Computers & Education

journal homepage: www.elsevier.com/locate/compedu

Ability-training-oriented automated assessment in introductory programming course

Tiantian Wang*, Xiaohong Su, Peijun Ma, Yuying Wang, Kuanquan Wang

School of Computer Science and Technology, Harbin Institute of Technology, Post Box 319, No. 92 West Da-Zhi Street, Harbin 150001, China

ARTICLE INFO

Article history:
Received 21 September 2009
Received in revised form
1 August 2010
Accepted 3 August 2010

Keywords:
Intelligent tutoring systems
Interactive learning environments
Programming and programming languages

ABSTRACT

Learning to program is a difficult process for novice programmers. AutoLEP, an automated learning and assessment system, was developed by us, to aid novice programmers to obtain programming skills. AutoLEP is ability-training-oriented. It adopts a novel assessment mechanism, which combines static analysis with dynamic testing to analyze student programs. It not only helps students to sufficiently test the programs, but also evaluates whether the programs meet the specification or not. AutoLEP encourages students to find and work through bugs, by providing automatic syntactic and structural checking, and immediate detailed feedback. This can improve students' learning experience in programming and reduce the workload of the teaching staff. AutoLEP has been used in the C programming course at Harbin Institute of Technology and many other universities since 2004. The feedback on AutoLEP and its incorporation into the introductory programming course has been positive, both by students and teaching staff.

1. Introduction

Teaching the introductory programming course in which students first learn to write programs to solve problems is a great challenge. The teachers must introduce students to the discipline of computing and to the problem solving process, and help the students to learn a whole new way of thinking. Programming is a practical process. The students need to write, type in, correct and run programs, and often they need help from the teaching staff. However, with large class sizes, it is difficult for the teaching staff to synchronize their heavy schedules to provide additional help when the students need it. Therefore, when learning to program, it is essential that students are given the opportunity to practice in an environment where they can receive constructive and corrective feedback.

Widespread researches have been carried out and various automated learning and assessment systems for programming have been developed (Ala-Mutka, 2005). These systems, such as TRY (Reek, 1989), Scheme-robo (Saikkonen, Malmi, & Korhonen, 2001), BAGS (Morris, 2003), online Judge (Brenda, Andy, Andrew, & Chong, 2003), Quiver (Ellsworth, Fenwick, & Kurtz, 2004), and RoboProf (Daly & Horgan, 2004), usually adopt a dynamic testing assessment mechanism which runs a student program through a set of testing data. The only factor that can influence the mark is the success or otherwise of a test. The dynamic testing assessment mechanism does not take into account the way in which a problem has been solved. This sometimes leads to inequitable grading results, because a program producing a right output may not meet the programming specification. There is another drawback of dynamic testing based assessment systems. In many cases, a program, submitted by a novice in an examination, may not produce an output or even may not terminate when dynamically tested, which makes these systems fail to give reasonable marks.

CourseMarker (Higgins, Gray, Symeonidis, & Tsintsifas, 2005), Boss (Joy, Griffiths, & Boyatt, 2005), and GAME (Blumenstein, Green, Fogelman, Nguyen, & Muthukkumarasamy, 2008) improved the dynamic testing assessment mechanism by introducing static analysis. These tools perform quality or style checks against a set of metric tools and provide feedback. GAME also performs structure analysis by examining strategic key points in the code. However, the drawbacks of the dynamic testing assessment mechanism have not been well solved yet.

There are also some systems that adopt a static analysis mechanism by comparing student programs with the model programs provided by teachers. For example, ELP (Truong, Roe, & Bancroft, 2005), WAGS (Norshuhani et al., 2006), and the system proposed by Khirulnizam

* Corresponding author. Tel./fax: +86 451 86412824.
E-mail address: sweetwtt@126.com (T. Wang).

0360-1315/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compedu.2010.08.003

et al. (2007). These systems can evaluate how close a student program is to a given model program. However, they can't carry out semantic analysis. The teachers need to provide more than one model programs in order to facilitate all the possible answer variation by the students. So these systems only works with small or "fill in the gap" type programming exercises (Khirulnizam & Md, 2007).

AutoLEP, an automated learning and examination system for programming, was developed by us. The purpose of AutoLEP is to automatically assess student programs and aid novice programmers to obtain programming skills. The novel aspects of AutoLEP are as follows. Firstly, AutoLEP adopts a novel assessment mechanism. It implements both dynamic testing and static analysis to complement each other. Especially, the static analysis mechanism, which is based on whole source code analysis, transformation and comparison, is different from existing works. Not only the structure but also the semantics of the programs are considered. The student programs and model programs are normalized to remove code variations, so the number of model programs needed to match with the student programs is significantly reduced. The dynamic testing assessment mechanism not only tests the correctness of student programs, but also recognizes new model program to renew the database of model programs. AutoLEP can evaluate not only the testing results but also the constructs of the programs. It can evaluate whether a student program meet a specification and give reasonable marks to programs with syntactical or logical errors, which can not be done by most of the existing automated assessment systems. Secondly, AutoLEP provides automatic syntactic and structural checking, as well as immediate detailed feedback. This encourages students to find and work through bugs. The advantages of using AutoLEP are that the students' programming ability is improved, students are assessed on what they have to do in practice, rather than merely on their theoretical knowledge, and that the workload on the teaching staff is drastically reduced.

2. Requirements of the automated assessment system for novice programmers

There are four basic requirements of the automated assistance and assessment system for novice programmers.

(1) *Sufficient testing*. Novice programmers are not proficient enough to test their programs sufficiently. They may cover the regular cases in their solutions but fail to treat special cases. Therefore, it is important for the automated assessment system to provide sufficient testing.
(2) *Checking whether the programs meet the specification or not*. A program producing a right output may not meet the programming specification. For example, the students are required to implement a program that outputs ten characters '*' by using an iteration structure. However, some students use ten output statements instead of an iteration structure. Thus automatically checking whether the programs meet the specification is an important aid for novice programmers who are still learning how to deal with precise specifications.
(3) *Dealing with programs with syntactic and semantic errors*. It is very challenging for a student to write a totally correct program in limited time. In many cases, a program, submitted by a novice in an examination, may not produce an output or even may not terminate when dynamically tested. It is not reasonable for novice programmers to be given zero marks, when there are syntactic or semantic errors in their programs. If a student always gets zero marks, he will lose faith in learning programming. So the automated assessment system should be able to point out the errors and give reasonable marks to partly correct programs.
(4) *Immediate and corrective feedback*. Timely and corrective feedback is vital in the learning process. It is especially important for novice programmers since many have not yet formed an effective internal model of a computer that they can use to construct viable knowledge. Most of the novice programmers are eager to know how much progress they have made in programming. They want to know not only whether their programs are correct, but also the details about the errors, to help point them in the right direction. Based on the feedback, they can aware what they need to further study and improve.

3. AutoLEP system

AutoLEP was developed to overcome the drawbacks of existing automated assessment systems. It aims at satisfying the requirements of the automated assessment system for novice programmers. The architecture of AutoLEP is shown in Fig. 1. AutoLEP consists of the following three components.

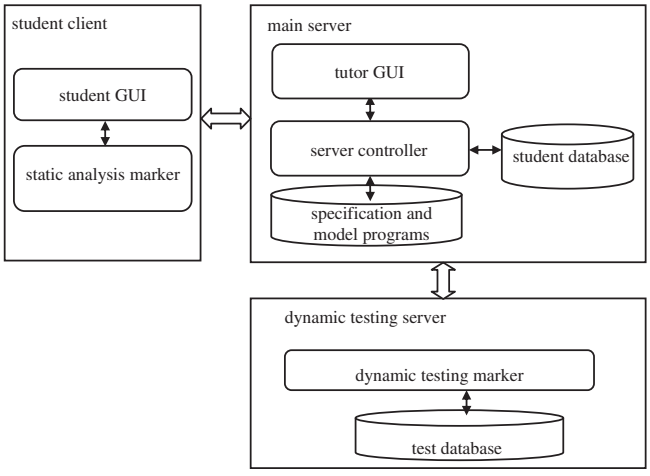

Fig. 1. The architecture of AutoLEP.

- *Student clients*. A student client is composed by a student GUI and a static analysis marker. The student GUI allows students to submit programs to the automated assessment system to be checked and provides feedback to the students. The static analysis marker grades a student program based on static analysis of the whole source code. It can give reasonable marks to partly correct programs. A student program is graded by comparing its source code with those of the model programs, which represent the right way to implement the programming specification. The more similar the student program is to the model programs, the higher the mark is. Static analysis marker is run on the student's machine in order to reduce the load on the server and provide immediate feedback.
- *Dynamic testing server*. It is composed by a dynamic testing marker and a test database. The dynamic testing marker adopts a dynamic testing marking mechanism to grade student programs. Programs, which are potentially correct but not given full marks by the static analysis marker, are sufficiently tested based on the predefined testing data. The dynamic testing marker is a complement to the static analysis marker. It can avoid the problem of the low grading precision of the static analysis marker caused by lacking of model programs.
- *Main server*. It is composed by a tutor GUI, a server controller, a student database, and a specification and model program database. The tutor GUI facilitates the teaching staff to set up new assignments, view various statistics, and review students' submissions. The server controller is the control centre of the main server application. It communicates with the student clients and the dynamic testing server.

The main difference between AutoLEP and the other existing programming assessment systems is the assessment mechanism. AutoLEP combines static analysis with dynamic testing to assess student programs. The assessment mechanism of AutoLEP has the following advantages.

(1) It is more robust than the existing assessment mechanisms. Not only the totally correct programs, but also the programs with syntactic or logical errors, can be accurately graded by AutoLEP.
(2) It evaluates student programs on the aspects of syntax, structure, semantics and testing results. Therefore, it can recognize whether a program satisfies a specification or not.

4. Programming ability training using AutoLEP

4.1. Checking syntactic and structural defects

AutoLEP provides automatic syntactic and structural defects checking to help the students to eliminate syntactic errors and structural defects. When a student finishes a program and chooses to solve the next problem, the student client will automatically check the syntax and the structure of the program. If there is any syntactic error or structural defect, a message box will appear in the interface, as shown in Fig. 2, to give a warning.

The syntactic error warnings are like those given by common compilers. The structural weakness warnings include: unused variable declaration, unused assignment, unused return value, and dead code. These warnings can promote the students to check and fix their programs, which results in a lower fraction of programs with syntactic errors and structural defects.

4.2. Meeting the specification

A student program and a set of model programs are used as input of the static analysis marker of AutoLEP. However, due to syntactic variations of programming languages, a given programming specification can be implemented in various ways. It is infeasible to write all the possible solutions as model programs. To solve this problem, a semantic similarity based grading approach was developed. Firstly, the student program and model programs are transformed into system dependence graphs. Secondly, semantic-preserving transformations are performed on the system dependence graphs to eliminate code variations. Finally, a score is gotten by matching the normalized system

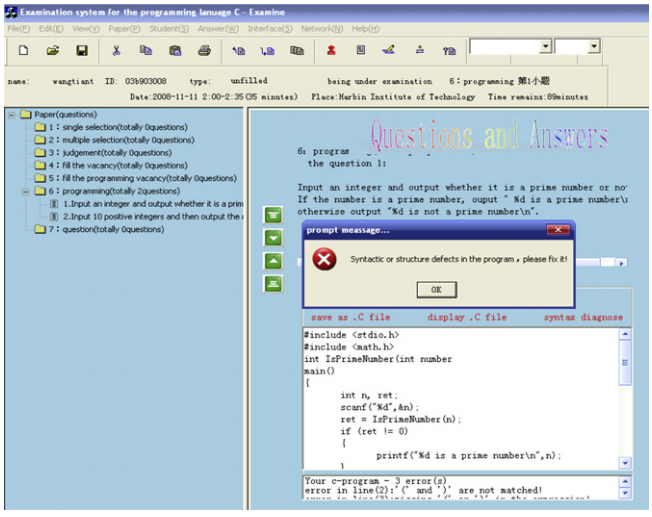

Fig. 2. Automatic syntax and structure checking.

dependence graph of the student program with those of the model programs. In this approach, the code variations in student programs and model programs are removed in advance before they are compared, so that syntactically different programs, which are implemented with the same algorithm, can match with the same model program. As a result, the number of necessary model programs is significantly reduced. The approach is described in more detail in our previous paper (Wang, Su, Wang, & Ma, 2007).

The advantages of the static analysis marker are that it can evaluate how close the source code of the student program is to correct solutions, and even partly correct programs can get reasonable marks. Based on the assessment results, students can aware whether their programs meet the programming specification. If not, they should find another way to solve the problem.

4.3. Dynamic testing

Although the static analysis marker of AutoLEP has the above mentioned advantages, it has the disadvantage that the accuracy of grading is highly dependent on the completeness of the model program sets. To solve this problem, dynamic testing marker is introduced.

The mechanism of the dynamic testing marker is similar to that of the most of the existing dynamic testing based systems. It runs a student program through a set of testing data, and then compares the output with the predefined answer. It can correctly give marks to the specific student programs that do not have matching model programs in the database. Meanwhile, new model programs can be identified to renew the database of model programs. Therefore, the dynamic testing and the static analysis mechanism complements to each other very well.

The dynamic testing marker also has the advantage that it can sufficiently test the student programs. It points out whether a student program passed the dynamic testing or not. If a test fails, detailed information on that test is included. This information includes a copy of the input supplied to the program and the correct output the program should have generated. Because the students see the input data that resulted in errors, they have the opportunity to learn something about the nature of good testing data.

4.4. Immediate and detailed feedback

AutoLEP provides an instant feedback after a student submits his programs. It provides the student with the assessment results in detail, as shown in Fig. 3.

- The “syntactic and structural defects” part points out the syntactic and structural defects in the student program, including the number and details of syntactic errors and structural defects if there are any.
- The “dynamic testing” part points out whether the student program passed the dynamic testing or not. If a test failed, detailed information on that test is included.
- The “semantic analysis” part describes the matching results of the student program with each model program on the level of size, structure, statement and key node.

From the feedback, the students become aware of their own limitations and difficulties. For example, if a program does not pass the dynamic testing, it means that the function of the program is not correct and there may be some logical errors. If a student program gets a low matching mark in the semantic analysis part, it means that the program has defects in structure or logic, so that the student should pay attention to improve the skill in writing good structured or logical programs. As a result, the detailed feedback information often helps point students in the right direction. The students learn to pay better attention to their coding practices, and they develop themselves good basic programming habits already on the first course.

5. Ability-training practice

AutoLEP is practical in both programming laboratory courses and examinations.

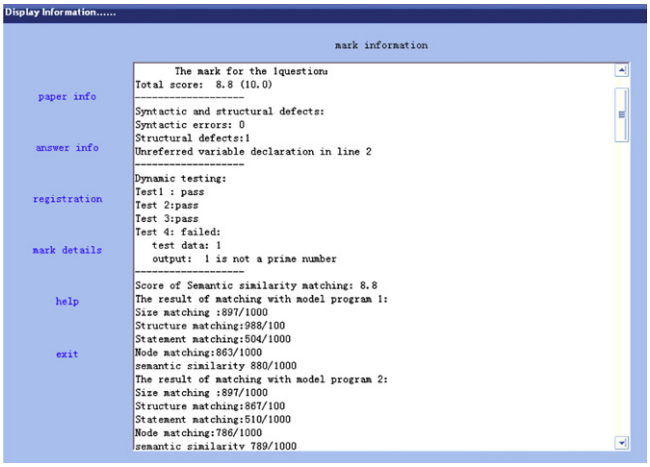

Fig. 3. Grading result feedback.

5.1. Laboratory course

In the laboratory course of programming, teachers usually set a set of programming exercises. The students are expected to complete these exercises before the date deadline. Correcting this amount of practical work manually would load a heavy burden on the teachers when the class is large; with AutoLEP, the corrections and feedback do not depend on the number of students. The teachers only need to set programming specification, corresponding model programs and testing data used in the AutoLEP system. Therefore, the workload of teachers is greatly reduced.

AutoLEP also helps students to achieve their objectives. The characters as follows make AutoLEP very popular with the students. Actual learning happens when students design and implement their programs, trying to apply good coding practices in the process.

(1) With AutoLEP a student can check if there are any syntactic errors or structural defects in his programs whenever he wants to. This is easy to accomplish only by clicking the "syntax diagnosis" button on the student GUI. If he forgets or bothers to do so, a compulsory automatic checking is performed when he finishes a program. A message box will pop up to give warnings if syntactic errors or structural defects are detected. These warnings prompt the student to modify his program. From the detailed warnings, the student can aware where his weaknesses exist. He may review related knowledge, and then fix the program. After that, he can check the program again. This process can iterate several times until the student gets a right answer. During this process the student learns from errors to improve programming skills.
(2) The instant and corrective feedback also contributes to helping a student to build around a correct solution. As stated in Section 4.4, a student submits programs and receives immediate feedback on how well he has succeeded and which issues he should pay more attention to. He is free to fix and resubmit the programs. This iterative process encourages him to find and work through bugs, which results in a higher fraction of correct programs.

5.2. Final examination

Assessing programming skills is notoriously difficult under examination conditions. In many cases, a program, submitted by a novice programmer in an examination, may not produce an output or even may not terminate when executed. The static analysis and dynamic testing combined assessment mechanism of AutoLEP makes it suitable for automatic grading of student programs. AutoLEP can evaluate the student programs on the aspects of syntax, semantics and functionality. It can accurately grade programs with syntactic or logical errors. This is an advantage of AutoLEP over the other existing automated assessment systems.

For the introductory programming class, an online examination was introduced as part of the final examination in Harbin Institute of Technology (HIT) to evaluate the students' coding and debugging skills. Fig. 4 shows a picture of online examination using AutoLEP.

The advantages of using AutoLEP in the final examination are as follows.

(1) The students are assessed on what they have to do in practice. This urges the students to practice more in programming. The number of students, who are reciting without understanding of the theoretical knowledge, is greatly reduced.
(2) The teaching staff do not need to mark the answers manually, so the workload is drastically reduced.

6. Impact on students

This section presents some findings from using AutoLEP in the introductory C programming course. A quantitative analysis of students' performance and a quantitative analysis of the robustness of the assessment mechanism are presented. Also, a qualitative analysis of student and faculty feedback provides further insight into AutoLEP.

6.1. Quantitative analysis of impact on students' performance

In order to evaluate the efficiency of AutoLEP, we chose four classes of first-year undergraduates in HIT to do the experiment. Each class consists of 30 students. The students in the four classes had similar entrance scores, and they were taught by the same teacher with the same

Fig. 4. Online examination using AutoLEP.

Table 1
Evaluation on students' performance.

	Group1	Group2
Syntactic and structural defects rate	5.6%	8.7%
Functional error rate	10.1%	13.5%
Average grade (0–100)	79.3	73.2

book. We divided the 4 classes into two groups, with each group consisting of two classes. The students in Group1 were introduced to AutoLEP and were told to use it for their laboratory work, while the students in Group2 were told to do their laboratory work manually and to e-mail their solutions to their respective lab assistants. The lab assistants would then mark the students' work by hand. The workflows were similar for the students in the two groups. All the students needed to write and submit the programs. After a semester's learning and practicing, all the students in the two groups took part in the same final examination. They were given an hour for writing the answers for five specified programming tasks. All the answers were marked firstly by AutoLEP, and then verified by the teacher.

The evaluation of how the students' usage of AutoLEP during the laboratory course affected their performance in the programming examination is shown in Table 1.

As shown in Table 1, both the syntactic and structural defects rate and the functional error rate of Group1 are significantly lower than those of Group2. The main reason for this is that AutoLEP system enables more rigorous enforcement in what students is expected to build. AutoLEP performs automatic checking on syntax and structure defects, gives immediate feedback of the dynamic testing and semantic analysis, during the students' practice of programming. The teaching assistants can give the feedback too. However, it is time consuming and can not be done in an instant way. Without immediate feedback, students are prone to neglect trivial errors that result in large grading penalties. With feedback, however, students find out immediately if their program has bugs, and they can then fix the bugs and resubmit the program. The AutoLEP system thus allows students to be held to a higher standard and enables students to meet this standard.

The average grade of Group1 is higher than that of Group2 by 6.1 marks (The full mark for a correct answer is 100.). This also shows that AutoLEP helps students to achieve the main laboratory work objective—to learn the programming techniques.

6.2. Quantitative analysis of the robustness of the assessment mechanism

In order to analyze the robustness of the assessment mechanism of AutoLEP, we divided the programs submitted by the students into five categories: not compile, not producing output, not completely right, not meeting the specification, and completely right.

Through the result in Fig. 5, we can see that it was challenging for the novice programmers to get completely correct within limited time. In the test, an average of 7.2 percent of the programs had syntactic errors; an average of 3.8 percent of the programs compiled and linked properly, but did not produce any output; an average of 8.0 percent of the programs could produce output but the results were not completely right. Such programs could not be graded by the purely dynamic testing based grading systems, but were given reasonable marks by the static analysis marker of AutoLEP. In the test, an average of 1.6 percent of the programs gave the right output in the dynamic test but did not meet the programming specification. They got full marks from the purely dynamic testing based grading systems, which is improper. AutoLEP avoided such problem by limiting model programs.

To sum up, the assessment mechanism of AutoLEP is robust and it has the advantages over the purely dynamic testing-based grading systems in the ability to deal with the programs that can not produce output, as well as the ability to recognize whether a program satisfies a programming specification or not.

6.3. Qualitative analysis

AutoLEP has been used in the introductory programming courses in HIT and many other universities since the year 2004 and has been used by about 2000 students each year. Experiences and feedback on the system usage have been collected by different kinds of questionnaires from students and in personal discussions with them over the years. The teaching staff have also been interviewed about their

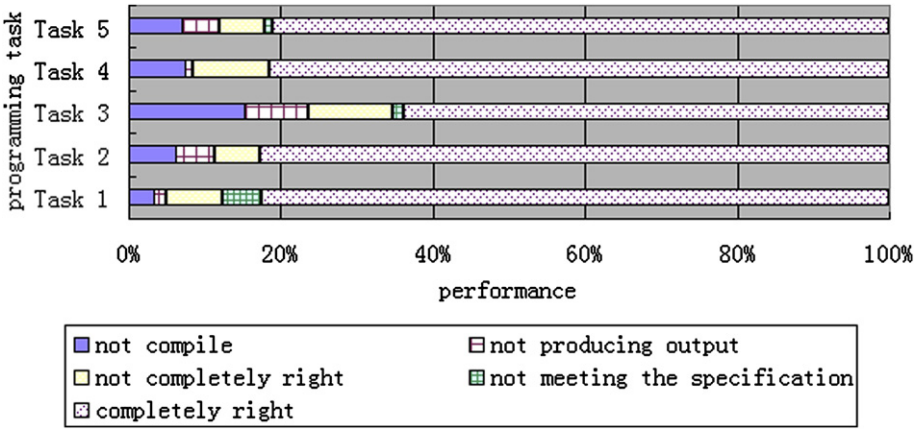

Fig. 5. Students' performance on programming tasks.

experiences in using the system and in tutoring students who have used it. Generally, the feedback on AutoLEP has been positive, both by students and teaching staff.

All students reported that they had enjoyed the experience of using AutoLEP. Most of the students were in agreement that AutoLEP was very useful and interesting. They were very happy to receive immediate feedback on how well they had succeeded and which issues they should pay more attention to. They also appreciated that AutoLEP treated everyone equally, and it could give objective marks to their programs. Particularly the fact that not totally correct programs could get reasonable marks from AutoLEP, was appreciated by unskilled novice programmers.

The teaching staff were also enthusiastic of the ability-training-oriented automated learning and assessment approach. After the application of AutoLEP, the teaching staff felt that the students were more interested in programming in the laboratory than ever before. After a period of time, the students' programming skills had been greatly improved.

7. Conclusions

This paper has analyzed the requirements and important roles of the automated assessment in the teaching of introductory programming course, and then presented AutoLEP, an ability-training-oriented automated learning and assessment system. The novel assessment mechanism and the immediate and corrective feedback of AutoLEP make it very popular with the students and the teaching staff. As for the teaching staff, AutoLEP can improve the quality of teaching programming, and reduce the workload. As for the students, AutoLEP can promote their interests in programming, and improve their programming skills and capability of self-learning.

Acknowledgements

This research is supported by the National Natural Science Foundation of China under Grant No.60673035 and 61073052, as well as the Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant No.20092302110040.

References

Ala-Mutka, K. (2005). A survey of automated assessment approaches for programming assignments. *Computer Science Education, 16*(2), 83–102.

Blumenstein, M., Green, S., Fogelman, S., Nguyen, A., & Muthukkumarasamy, V. (2008). Performance analysis of GAME: a generic automated marking environment. *Computers & Education, 50*, 1203–1216.

Brenda, C., Andy, K., Andrew, L., & Chong, O. W. (2003). On automated grading of programming assignments in an academic institution. *Computers and Education, 41*, 121–131.

Daly, C., & Horgan, J. M. (2004). An automated learning system for Java programming. *IEEE Transactions on Education, 47*, 10–17.

Ellsworth, C. E., Fenwick, J. B., & Kurtz, B. L. (2004). The Quiver system. In: *Proc. 35th SIGCSE Technical Symp. on Computer Science Education, (Virginia, 2004)* pp. 205–209.

Higgins, C. A., Gray, G., Symeonidis, P., & Tsintsifas, A. (2005). Automated assessment and experiences of teaching programming. *ACM Journal of Educational Resources in Computing, 5*(5), 1–21.

Joy, M., Griffiths, N., & Boyatt, R. (2005). The boss online submission and assessment system. *ACM Journal on Educational Resources in Computing, 5*(2), 1–28.

Khirulnizam, A. R., & Md, J. N. (2007). A review on the static analysis approach in the automated programming assessment systems. In: *National Conference on Software Engineering and Computer Systems, (Pahang, Malaysia, 2007)*.

Khirulnizam, A. R., Syarbaini, A., & Md, J. N. (2007). The design of an automated C programming assessment using pseudo-code comparison technique. In: *National Conference on Software Engineering and Computer Systems, (Pahang, Malaysia, 2007)*.

Morris, D. S. (2003). Automatic grading of student's programming assignments: an interactive process and suit of programs. In: Conf. 33rd Annu. Frontiers in Education, (Boulder, 2003), pp. 1–6.

Norshuhani, Z., Emy, E. M., Savita, K. S., Mazlina, M., Ellia, A. (2006). Development of a web-based automated grading system for programming assignments using static analysis approach. In: *International Conference on Electrical and Informatics, (Bandung, Indonesia, 2006)*.

Reek, K. A. (1989). The TRY system -or- how to avoid testing student programs. *SIGCSE Bulletin, 21*(1), 112–116.

Saikkonen R., Malmi L., & Korhonen, A. (2001). Fully automatic assessment of programming exercises. In: *Proc. 6th Annu. Conf. on Innovation and Technology in Computer Science Education, (Canterbury, 2001)* pp. 133–136.

Truong, N., Roe, P., & Bancroft, P. (2005). Automated feedback for "fill in the gap" programming exercises. In: *Proceedings of the 7th Australasian Computing Education Conference, (Newcastle, Australia, 2005)* pp. 117–126.

Wang, T. T., Su, X. H., Wang, Y. Y., & Ma, P. J. (2007). Semantic similarity-based grading of student programs. *Information and Software Technology, 49*, 99–107.