Journal of Theoretical and Applied Information Technology
10th December 2014. Vol.70 No.1
© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A PERSPECTIVE OF AUTOMATED PROGRAMMING ERROR FEEDBACK APPROACHES IN PROBLEM SOLVING EXERCISES

[1]S. SUHAILAN, [2] S. ABDUL SAMAD and [3] M.A BURHANUDDIN

[1,2,3]Faculty of Information and Communication Technology,
Universiti Teknikal Malaysia Melaka, Melaka, Malaysia

[1]Faculty of Informatics and Computing,
Sultan Zainal Abidin University, Terengganu, Malaysia

E-mail: [1]suhailan@unisza.edu.my , [2]samad@utem.edu.my , [3]burhanuddin@utem.edu.my

ABSTRACT

Programming tools are meant for student to practice programming. Automated programming error feedback will be provided for students to self-construct the knowledge through their own experience. This paper has clustered current approaches in providing automated error programming feedback to the students during problem solving exercises. These include additional syntax error messages, solution template mismatches, test data comparison, assisted agent report and collaborative comment feedback. The study is conducted based on published papers for last two decades. The trends are analyzed to get the overview of latest research contributions towards eliminating programming difficulties among students. The result shows that future direction of automated programming error feedback approaches may combine agent and collaborative feedback approaches towards more interactive, dynamic, end-user oriented and specific goal oriented. Such future direction may help other researchers fill in the gap on new ways of assisting learners to better understand feedback messages provided by automated assessment tool.

Keywords: *Programming Language, Automated Programming Error Feedback, Computer Aided Learning System, e-Learning*

1. INTRODUCTION

Computer programming is a mandatory subject for computer science. It is usually offered to students in the early semester. Although as an introductory subject, nature of the subject requires students to practice programming rather than understanding and memorizing the concepts and techniques. Through practices of real programming, student will be able to constructively build up the programming skills through own experience as described in constructive learning concept. The most common outcomes from the practice sessions are syntax competencies and semantics skills capabilities. Practice sessions usually conducted in computer lab under supervision of tutor or teacher. Students will try individually to complete problem solving exercises by writing codes using Integrated Development Environment (IDE) and programming language compiler. Each time the codes are compiled, the tool will automatically provide feedback messages highlighting all the syntax errors found in the codes.

Nowadays, lots of automated programming assessment tools with automated feedbacks [1] are continuously developed. Researchers have put effort to automate codes assessment not only in term of syntax errors or runtime output, but also in coding structure, styles and performance. Feedbacks are instantly provided on each assessment when errors or mistakes are made by students. Such feedbacks are static library of messages based on language syntax errors. Instead of syntax checking, solution structure checking are also available using mismatched comparison between predefined teacher's solution and current students' solution. However, current automated feedbacks are still insufficient compared to teacher's feedback [2][3][4].Current feedbacks require self-interpretation that will be

© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

difficult among novice students. Furthermore, the feedbacks cannot be further explored in understanding the messages. Without proper feedback or guidance, students will fail to learn and construct their own knowledge on programming [5]. Continuous guidance and instant feedback by teacher is among the key factor [6] in learning programming. Teacher should first guide the students to help them self-discover the errors by applying syntax knowledge and justifying the semantic of the codes. Meanwhile, guiding students can be challenging tasks for a teacher when involving groups of students. Students' understanding of programming topics are different and varies among individuals. Teacher need to provide specific feedback on each students based on the quick assessment of students' codes. Such assessment may take longer time to be analyzed.

This paper has reviewed current approaches of automated programming error feedback in helping student completing problem solving exercises. The motivation of this paper to highlight new direction of automated error programming feedback that can support self-constructive learning in systematic and automated collaborative human feedback. Such future direction may help other researchers to fill in the gap in modern techniques for helping the programming self-learning style.

2. RESEARCH QUESTIONS AND METHODS

This study is conducted based on analysis of published papers from year 1990 to 2013. Although many researches are done on learning programming, this research is focusing on related approaches to provide automated feedback messages during real coding exercises. The trends are analyzed to get the overview of latest research contributions towards eliminating difficulties among students specifically in problem solving programming exercises.

The following research questions are addressed in this paper:

1. What are the techniques used to provide automatic feedback on automatic assessment systems reported in the literature since 1990?

2. What are the current problems in automatic feedback messages provided by automatic assessment systems?

3. What are the future directions of automated feedback that can be concluded based on the current problems? This future directions may be possible gaps that can be filled by future research enhancement in automatic assessment.

By automated feedback on problem solving exercises we mean automatic messages that provided to students who write code and submit it for syntax and problem solution assessment. Therefore, this study will focus on approaches in providing feedback messages rather than assessment techniques. In addition, other assessment such as on programming theories, fill in the blank, error or output tracing are not included.

Data are collected by searching for phrases "(automatic or automated)" and "(assessment or grading or feedback)" and "programming" from the conference proceedings and journals through ACM Digital Library, IEEE Xplore, Springer and ScienceDirect. Current programming competitive system are also reviewed to identify the most applicable technique for automated feedback. We then applied the inclusion criteria to the abstracts and finally read all the remaining papers.

An iterative process is applied to find the consensus on compiling all the automated feedback techniques into groups (Section 3). Then problems on automated feedback are classified (Section 4) to conclude the future direction on the possible solutions (Section 5). Papers are selected then categories are made or revised. This was repeated until no significant features leading into new categories were found.

3. AUTOMATED ERROR PROGRAMMING FEEDBACK APPROACHES

Automated error programming feedback is an automated report messages to student in notifying what are the errors contains in the programming codes. The errors can be possibly in term of compilation errors, runtime errors and solution errors. Five types of analysis approach are identified in providing automated feedback to students so that the codes can be successfully compiled, executed and fulfilled to the problem specification.

Journal of Theoretical and Applied Information Technology
10th December 2014. Vol.70 No.1
© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3.1 Additional Syntax Error Messages Feedback

Inappropriate and unclear error message will lead to misinterpretation and demoralization among novice students. Researchers have put effort to enhance standard error feedback messages provided in standard programming language compiler to become more appropriate and clear.

Merr developed by [7] to mapping new additional error messages into existing syntax error analysis report using Yacc. It will supplement the standard error messages with the new added error messages. This tool can be used by compiler writer to easily add new messages of certain existing error identification.

Java Error Correction Algorithm (JECA) is proposed by [8] to detect syntax errors in the codes and provide hints to guide student on how to correct the errors. Syntax error analysis are based on auto-recovery of the possible misspelled source code tokens with the programming language reserved words ("for", "while", "break", "double", "int", and "void") and previously declared variables. For example, "forr" will be replaced by reserved word "for" if it is not declared as a variable. Another example is undefined variable of "count" will be changed to "count1" if "count1" is already defined as variable previously.

More comprehensive enhanced standard feedback of standard Sun-Java errors is provided by [9]. More clearer messages interpretation are associated on standard error feedback messages which produced by standard Sun-Java covering on: -

i. compile errors that consists of syntax errors, identifier errors, computational errors, return statements and access to static entities;

ii. runtime errors that consists of out of range, dereferencing null, computation, insufficient memory, program execution

However, the effectiveness of such additional interpretation is not yet tested since the solution is more in written notes.

This kind of approaches can be very helpful for student especially in identifying the exact location of the syntax error in the codes. However, the such error enhancements feedback are static and does not reflect to student's specific intention or goal. For example, syntax error in variable initialization such as "int msg = "10"";", will raise type mismatched error and may provide recovery hint either to change the variable type to "String" or remove the double quote of the value. The feedback does not evaluate the student's intention in creating such variable either to be manipulated as string or integer.

3.2 Solution Template Mismatches Feedback

In order to be able to assess student's code intention, researchers suggest all the problem questions to be associated with specific goals and sub-goals. Each sub-goals must be represented using specific solution template. The template is a sample programming codes solution to achieve the sub goals. Such template will provide basic structure of codes logic or semantic to be compared with student's answer.

PROUST developed by [10] to diagnose error based on program sub goals. It define common programming goals as data input, output and processing. Each goal will be associated with one or many of solution template alternatives. These pre-defined goals need to be selected by teacher to be associated with a new created problem exercise. Then, PROUST will analyze student's codes using representation tree to find which part of the codes matches with any of the specified template in the pre-defined sub goals. Automatic feedback will be tailored based on errors found from portion of codes that failed to match certain structure of the specific solution template. Any mismatches will be inserted into the appropriate line of codes by translating the codes into pseudo-codes statements. However, the technique may fail to assess the codes intention if the codes are written totally different from the existing pre-defined solution template. [11] create a similar approach as PROUST called GOES. As PROUST is designed for Pascal language, GOES is designed for C language.

Java Intelligent Tutoring System (JITS) is developed by [12] to provide special feedback based on specific problem exercises on variables, operators and looping. Problems have been initially designed and assigned with list of programming decision tree to assess student's answer against the trees. Programming decision tree is a specific knowledge representation for achieving goals in the given problem statement. It consists of discrete section evaluation steps on verifying a programming statement. For example, if the problem requires a loop, the system will check if there is "for" or "while" statement in the

© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

codes. If the word is found, it will assess the looping correctness using the tree. It will start to assess the looping section correctness; initial value, conditional section and update section. If this step success, it will continue to check on initial value section in depth such as is it variable has been correctly defined, is it initial value has been assigned and so on. If student's codes did not match with the discrete code structure template, it will provide feedback to guide which part of the codes need to be revised. This approach however limit the creation of new problems set as each problems need to be constructed with specific solution goal templates. Thus, it cannot be recycled for new incoming students because the answers of the exercises can be easily obtained from previous students. It also does not entertain other new solution alternatives provided by student if the structure is not exists in the system knowledge-base.

New approach to check student's codes by comparing on common mistakes template to complement the solution template approach is proposed by [13]. They have designed error model language (EML) that consists of potential correction rule-list based on common errors on the specific problem. The model is much like context free grammar (CFG) that originally used in representing syntax of a language. Alternatively, the EML are used to represent common mistakes pattern with some additional hints on how to correct the mistakes. However, the EML is not designed to be global because it just represent common mistakes on a specific exercise or problem.

Solution templates can be taken from correct codes which are submitted by students is proposed by [14]. This will eliminate burden on teacher to prepare the solution templates on each problem exercises. It also provides more solution alternatives for mismatched comparison.

Many of researchers that use pre-defined solution template strategy claim that highest success rate in helping student solving programming problems are achieved. Although it helped in term of solving problem, but it cannot be concluded as learning effectiveness because student tend to abuse the automated feedback using trial and error strategy to get closer to the solution and finish the [15]. Instead of analyzing the errors, student may tend to use try-and-error strategy with the intention of just solving the problems. If the scenarios happens, students are not contributing themselves in the learning process. Furthermore, student will become frustrated after several of the attempts failed to solve the problems. So the outcome of the programming tool usage to help student at least constructively learn programming by correcting misconception cannot be effectively achieved.

3.3 Test Data Error Feedback

Automated programming assessment tool such as [16]–[20] are among common platform for students to practice programming skills by writing codes to solve real life problem solving questions. These tools provide online programming editor for the students to write and submit the codes answer via internet browser. It will automatically check either the submitted codes are correct or not by automatically execute the codes with the specified test input and compare the output with expected output data. However, such platform are not popular among novice students because it more towards competitive programming among highly interested or expert students.

[21] develops CodeCloud to function same as the competitive programming tools but are made open source for researchers to possibility exploit their own log data using their own server. While other tool require teacher to post exercise, [22] develop CodeWrite to offer exercises and test case to be provided by students themselves. This tool can provide instant feedback for student to overcome deficiency in manual assessment [23]. However, since such tools only concern with the correctness of the final output, not much feedback detail can be provided to students in helping them correcting the mistakes. For example, the codes may be correct in dealing with input and process, but failure in formatting the output. Another example is that the codes run well in data processing and output formatting, but it has mistake in interpreting input types or formats. Furthermore, in some cases that involve number of branches, more than one possible output may need to be specified to assess the answered codes.

3.4 Assisted agent feedback

Tracking and logging student e-learning activities as profile for assisting teacher is proposed by [24]. It then detect any learning conflicts and report the possible reason to teacher based on association rules. Conflicts may be alerted to teacher if students do not participate or fail to finish in the assigned exercises. The

© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

conflict detection is based the association of current student activity on the exercises such as error encountered or silent action. It provide assistance for teachers to provide instant interference to individual student who really require constructive feedback.

Individual reports may be a burden for a teacher to analyze and entertain possibly same feedback for each individual. [25] use data mining technique to compile the current errors of students into groups of errors. The error groups are categorized into field, syntax, type, import, method, constructor and visualization (output) error. The groups are processed based on compiler messages (warnings and errors) from the student's code attempt to answer programming exercises. Such reports are then can be used by teacher to analyze student understanding and help the teacher to re-emphasize on certain concept that most students failed to achieve in the classroom.

Test My Code tool was developed by [26] to provide online exercises that composed of small incremental tasks, which combine into bigger programs. It also include automatic test and assessment on the smaller parts. Thus student can self assess which parts of the exercise that has problem and ask for specific code review from teacher. Instead of studying the whole codes, teacher can save lot of time to response on specific skills.

However, assisted agent feedback may not be effective in large size of students number. If the feedback are required at the same time by many students, feedback from a teacher to individual students may be delayed and eventually distract the learning motivation.

3.5 Collaborative peer feedback

Another approach to support programming error feedback is using collaborative environment platform such as electronic forum discussion or social network.

Stack overflow [27] is a sample effective electronic forum discussion that offer student to post any questions and receive answers from experts from all over world. Compare to other electronic forum discussion that manage question into topics and subtopics tree, the platform offer easy and fast open ended text discussion with some keyword tagging techniques to index the discussion.

Java Assisted Learning System (JALs) is developed by [28] as an application running on Facebook platform, which allow students to compile Java programming codes and get assisted feedback from other Facebook's peer. It uses Facebook to enable students to discuss with other members to gain more feedback in clarifying the mistakes or errors encountered during the exercises. List of peer will be listed based on person who has previously successfully answered the problem, or if known, the teacher will be the peer to assist the student. However, instead of guiding, learning outcome may be violated if it happen that the peer distributed the whole correct answer to the students.

4. PROBLEMS WITH AUTOMATED ERROR PROGRAMMING FEEDBACK

Automated feedback are hard to be interpreted and exploiting among novice students in self-learning mode. Three types of problems or difficulties are identified as the following.

4.1 Unclear and inappropriate feedback

In certain cases, the standard feedback is not clear and inappropriate especially to novice students [9]. For example, an error of undeclared variable may generate feedback with more messages for each line of the variable occurrence. Thus, novice students will struggle to rectify the syntax errors especially when they are given with these imprecise error and inappropriate feedback[29][2]. Meanwhile, codes that do not have any syntax errors does not mean it has solved the exercise problem. It may fail in term of missing to address some of the problem specification or incorrect solution logic. Some of the researchers have also provided automated feedback to tell the missing or incorrect codes based on provided solution templates. However, student may tend to abuse the automated feedback using trial and error strategy to get closer to the solution and finish the exercises [15]. Thus constructive learning will not be achieved if the case happens.

4.2 Lacking of interactivity

Although automated feedback has covers for syntax and semantic guidelines, but still the approach is insufficient compared to teacher or peer feedbacks [2], [3][4]. Automated feedbacks

© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

are provided as static messages and no further responses are generated without changes to the current codes. Lacking of such interactibility requires students to think and figure out on their own of the meaning of the feedback. Thus, many of students tend to ignore the provided feedback since they don't understand the messages and cannot interact with the system to ask for further clarification. Contrary, experienced teacher or peer can entertain student's responses in tailoring feedback to be understood. Social network platform or other electronic forum can be used as effective human interaction platform to seek feedback and comments. Looking at the opportunities of social network in assisting learning progress, [24], [25], [28], [30]–[32] have implemented programming learning into social network setting. The platform provides an environment that enable live messages and feedback discussion among peer or teacher to be done online. It also provides mechanism to log and index the interactive feedback discussion into topics or sub topics for future references. However, such interaction platform is not designed to be responsive in helping inactive students.

4.3 Non systematic and static feedback

Assisted feedback on programming error need to be provided in real time so that student can construct their own knowledge. Otherwise, students tend to forget the encountered misunderstanding concepts. Learning motivation may also decrease after the lab learning session ended. Previous discussion on the same programming errors need to be re-utilized in enhancing automated feedback in programming exercises on the same occurring problems or errors. Currently, many of automated feedback is designed based on set of pre-defined static messages library. Not many research have been done in utilizing social discussion as a way to improvise automated programming error feedback in becoming more sufficient and interactive. Such electronic discussion logs are currently not systematically designed to be intelligent especially in self-extracting previous related discussion content of the same programming errors occurrence.

5. DISCUSSION AND FUTURE DIRECTION

Based on the review, current automated feedback of the first three approaches are implemented in one way interaction where the feedback cannot be further explored without any modification or correction done to the codes. Changing the codes may create more errors without assessing and correcting student's code intention. Thus, if students themselves failed to construct the knowledge after experiencing errors, they will start to become clueless and frustrated. Although solution template try to guide student to modify the codes towards the answer schema, it may lead to try and error strategy and constructive learning may not in the place. It also does not entertain other solution alternative based on individual's creativity. Meanwhile, working alone with one way automated feedback interaction may discourage motivation among weak students who do not know what else need to be done to proceed.

On the other hand, assisted and peer feedback approaches can offer much richer, specific and support two ways communication. By having two ways communication, student can be guided specifically towards self-constructing understanding and avoid misconception in programming. [33] shows that new trend in online learning using social network interaction can motivate unmotivated students to learn programming.

It is likely that new approach of automated programming error feedback in social network will become an important element in future developments. However, involving human to complement automated programming error feedback can be very challenging in term of resources and knowledge management. Instead of burdening teachers, it may be impossible to provide instant feedback to individuals when the situation involves with many students. It also seems that content of feedback discussion may be violated with answer sharing or redundant with existing or unnecessary content. New specific ways of social network communication need to be addressed to become more responsive towards automatically detecting programming difficulties among students. The text discussion structure of the social feedback must be reinvented systematically so that it can be reutilized in self-improvising the static library of automated error feedback messages. Filtering agents must be designed to avoid unnecessary comments. The library must also be associated with template of

© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

common programming goal concepts such as static looping, dynamic looping, sentinel looping, array, input types, string manipulation and etc. By having programming goal, it can guide students towards answering the question and getting more appropriate automated feedback.

There are other review papers on the types of automatic assessment in learning programming and its experimental results of using certain types such as in [34][35][36][37][38][39][40][41][42]. However, the papers are more concern on the automated assessment and does not focus on the automated feedback. Thus, most of the papers have highlighted that the automated feedback need to be improved towards more clearer and understandable messages after the automated assessment are successfully executed. This paper has contribute towards classification of automated error programming feedback messages and highlight the future direction for researchers who want to improve the automated feedback.

6. CONCLUSION

Various automated programming error feedback approaches in helping student completing problem solving exercises have been reviewed. The approaches provide instant programming error feedback after automatically analyze student's solutions which serve as an important means for improving programming skills. Types of feedback approaches are identified as additional of syntax error messages, solution template mismatches, test data comparison, assisted agent report and collaborative comment feedback. There are some drawback that need to be addressed in the current approaches such as too much burden on teacher, lacking of interaction for clarification, unsystematic discussion and insufficient programming goal guideline. The reviewed result shows that future direction of e-learning feedback approaches are now embarking towards collaborative learning such as in social network environment. It may involve agents in assisting collaborative feedback approaches towards more interactive, dynamic, end-user oriented and specific goal oriented. Such future direction may help other researchers fill in the gap on new ways of assisting learners to better understand feedback messages provided by automated assessment tool.

REFRENCES:

[1] N. Le, S. Strickroth, S. Gross, and N. Pinkwart, *A Review of AI-Supported Tutoring Approaches for Learning Programming*, vol. 479. Heidelberg: Springer International Publishing, 2013, pp. 267–279.

[2] M. Rubio-sánchez, P. Kinnunen, C. Pareja-flores, and Á. Velázquez-iturbide, "Student perception and usage of an automated programming assessment tool," *Computers in Human Behavior*, vol. 31, pp. 453–460, 2014.

[3] R. Queirós, I. La, D. Ipp, and J. P. Leal, "PETCHA A Programming Exercises Teaching Assistant," in *17th ACM Annual Conference on Innovation and Technology in Computer Science Education*, 2012, pp. 192–197.

[4] H. Mungunsukh and Z. Cheng, "An Agent Based Programming Language Learning Support System," in *International Conference on Computers in Education (ICCE)*, 2002, pp. 148–152.

[5] A. Yadin, "Reducing the dropout rate in an introductory programming course," *ACM Inroads*, vol. 2, no. 4, pp. 71–76, 2011.

[6] M. A. Brito and F. De Sá-soares, "Assessment frequency in introductory computer programming disciplines," *Computers in Human Behavior*, vol. 30, no. 30, pp. 623–628, 2013.

[7] C. L. Jeffery, "Generating LR syntax error messages from examples," *ACM Transactions on Programming Languages and Systems*, vol. 25, no. 5, pp. 631–640, Sep. 2003.

[8] E. R. Sykes, "Qualitative Evaluation of the Java Intelligent Tutoring System," *Journal of Systemics, Cybernetics and Informatics*, vol. 3, no. 5, pp. 49–60, 2005.

[9] M. M. Ben-ari, "Compile and Runtime Errors in Java," Israel, 2007.

[10] W. L. Johnson, "Understanding and debugging novice programs," *Artificial Intelligence*, vol. 42, no. 1, pp. 51–97, Feb. 1990.

[11] J. S. Song, "AN INTELLIGENT TUTORING SYSTEM FOR INTRODUCTORY C LANGUAGE COURSE," *Computers & Education*, vol. 28, no. 2, pp. 93–102, 1997.

[12] E. R. Sykes and F. Franek, "An Intelligent Tutoring System Prototype for Learning to

© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Program Java TM," in *IEEE International Conference on Advanced Learning Technologies (ICALT 2003)*, 2003, no. figure 2, p. 7695.

[13] R. Singh, S. Gulwani, and A. Solar-lezama, "Automated Feedback Generation for Introductory Programming Assignments," in *ACM Programming Language Design and Implementation (PLDI)*, 2013, pp. 15–26.

[14] M. Gaudencio, A. Dantas, and D. D. S. Guerrero, "Can Computers Compare Student Code Solutions as Well as Teachers ?," in *45th ACM Technical Symposium on Computer Science Education*, 2014, pp. 21–26.

[15] B. E. Vaessen, F. J. Prins, and J. Jeuring, "Computers & Education University students ' achievement goals and help-seeking strategies in an intelligent tutoring system," *Computers & Education*, vol. 72, pp. 196–208, 2014.

[16] "Programmr," 2014. [Online]. Available: http://www.programmr.com/. [Accessed: 10-Apr-2014].

[17] "Codeforces," 2014. [Online]. Available: http://codeforces.com/. [Accessed: 10-Apr-2014].

[18] "Codechef," 2014. [Online]. Available: http://www.codechef.com/. [Accessed: 10-Apr-2014].

[19] "UVa Online Judge," 2014. [Online]. Available: http://uva.onlinejudge.org/. [Accessed: 31-Mar-2014].

[20] "Topcoder," 2014. [Online]. Available: http://www.topcoder.com/. [Accessed: 10-Apr-2014].

[21] A. Papancea, J. Spacco, and D. Hovemeyer, "An open platform for managing short programming exercises," in *Proceedings of the ninth annual international ACM conference on International computing education research - ICER '13*, 2013, pp. 47–51.

[22] P. Denny, A. Luxton-reilly, E. Tempero, and J. Hendrickx, "CodeWrite : Supporting Student-Driven Practice of Java," in *Proceedings of the 42nd ACM technical symposium on Computer Science education*, 2011, pp. 471–476.

[23] A. Kurnia, A. Lim, and B. Cheang, "Online Judge," *Computers & Education*, vol. 36, no. 4, pp. 299–315, May 2001.

[24] A. Casamayor, A. Amandi, and M. Campo, "Intelligent assistance for teachers in collaborative e-learning environments," *Computers & Education*, vol. 53, no. 4, pp. 1147–1154, 2009.

[25] C. Fernandez-Medina, J. R. Pérez-Pérez, V. M. Álvarez-García, and M. D. P. Paule-Ruiz, "Assistance in computer programming learning using educational data mining and learning analytics," in *Proceedings of the 18th ACM conference on Innovation and technology in computer science education - ITiCSE '13*, 2013, pp. 237–242.

[26] A. Vihavainen, T. Vikberg, M. Luukkainen, and M. Pärtel, "Scaffolding Students ' Learning using Test My Code," in *Annual Conference on Innovation and Technology in Computer Science Education*, 2013, vol. 68, pp. 117–122.

[27] "Stack Overflow," 2014. [Online]. Available: https://stackoverflow.com/tour. [Accessed: 22-May-2014].

[28] C.-H. Lai, W.-C. Lin, B.-S. Jong, and Y.-T. Hsia, "Java Assist Learning System for Assisted Learning on Facebook," *IEEE Learning and Teaching in Computing and Engineering*, pp. 77–82, Mar. 2013.

[29] P. Denny, A. Luxton-Reilly, E. Tempero, and J. Hendrickx, "Understanding the syntax barrier for novices," *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education - ITiCSE '11*, p. 208, 2011.

[30] M. S. Othman, S. M. Suhaimi, L. M. Yusuf, N. Yusof, and N. Mohamad, "An Analysis of Social Network Categories: Social Learning and Social Friendship," *Procedia - Social and Behavioral Sciences*, vol. 56, no. Ictlhe, pp. 441–447, Oct. 2012.

[31] E. Verdú, L. M. Regueras, M. J. Verdú, J. P. Leal, J. P. de Castro, and R. Queirós, "A distributed system for learning programming on-line," *Computers & Education*, vol. 58, no. 1, pp. 1–10, Jan. 2012.

[32] K. M. Y. Law, V. C. S. Lee, and Y. T. Yu, "Computers & Education Learning motivation in e-learning facilitated computer programming courses," *Computers & Education*, vol. 55, no. 1, pp. 218–228, 2010.

[33] N. Drljevic and I. Boticki, "Leveraging Social Networks to Increase Motivation in Learning Programming," in *54th International Symposium ELMAR*, 2012, no. September, pp. 12–14.

[34] P. Ihantola, T. Ahoniemi, V. Karavirta, and O. Seppälä, "Review of recent systems for automatic assessment of programming

© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

assignments," in *Proceedings of the 10th Koli Calling International Conference on Computing Education Research - Koli Calling '10*, 2010, pp. 86–93.

[35] V. Pieterse, "Automated Assessment of Programming Assignments," in *ACM Computer Science Education Research Conference*, 2013, no. April, pp. 4–5.

[36] N. Truong, P. Roe, and P. Bancroft, "Automated Feedback for ' Fill in the Gap ' Programming Exercises," vol. 42, pp. 117–126, 2005.

[37] R. Romli, S. Sulaiman, and K. Z. Zamli, "Automatic Programming Assessment and Test Data Generation," in *IEEE International Symposium in Information Technology (ITSim)*, 2010, pp. 1186–1192.

[38] P. Reimann, M. Kickmeier-rust, and D. Albert, "Problem solving learning environments and assessment : A knowledge space theory approach," *Computers & Education*, vol. 64, pp. 183–193, 2013.

[39] M. Joy, N. Griffiths, and R. Boyatt, "The BOSS Online Submission and Assessment System," vol. 5, no. 3, 2005.

[40] C. Douce, "Automatic Test-Based Assessment of Programming : A Review," *ACM Journal of Educational Resources in Computing*, vol. 5, no. 3, pp. 1–13, 2006.

[41] T. Wang, X. Su, P. Ma, Y. Wang, and K. Wang, "Computers & Education Ability-training-oriented automated assessment in introductory programming course," *Computers & Education*, vol. 56, no. 1, pp. 220–226, 2011.

[42] S. A. Naser, "AN AGENT BASED INTELLIGENT TUTORING SYSTEM," *Journal of Theoretical and Applied Information Technology*, vol. 4, no. 7, pp. 585–589, 2008.