www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 2 Issue 5 May, 2013 Page No. 1692-1694

AUTOMATED ASSESSMENT SYSTEM FOR SOURCE CODE

Sumeet H. Pardeshi, Amit M. Morvekar, Hemkant B. Gangurde, Mahesh D. Gangode
(amit.morvekar05@gmail.com)
(sumeet.pardeshi.online@gmail,com),
(mahesh.gangode@gmail.com)
(gangurde.hemkant@gmail.com)
Dissertation under the supervision of Prof. Jyoti R. Mankar and Prof. Dr. Shirish S. Sane
at K. K. Wagh Institute of Engineering Education and Research, Nashik, Maharashtra, India.

ABSTRACT
Generally programming contests are managed, monitored and judged manually which tends to be a major defect, thus leading contest to be more time consuming and complex to judge. The proposed project aims at developing a system wherein the participants can login to an online system and upload their codes and automated system will judge their programs. Judge aspect and Participant aspect are the two main views of system. Students can work on problem, compile their solutions and submit them to the online system. They will get the feedback about the code whether it is acceptable or specific error specifications if any. Judges can view the current status of code submission and rankings accordingly, which they can use to evaluate the work. Automated Assessment System for Source Code (AASSC) aims at providing fair evaluation of source codes be in programming contest, or online practical exams held as a part of university curriculum. Unlike other competitive products in market, which rank contest based on timing and number of submissions only, proposed system uses some benchmarking metrics for analysis of code and thus yielding a fair outcome. Plagiarism by students is the biggest emerging threat which this proposed system tries to tackle. System aims at identifying the plagiarized source code, thus leading to a more fair evaluation and helping to improve the quality of programming among the students.

KEYWORDS
Plagiarism, Static Analysis of Code, Source Code Benchmarking, Object Oriented Metrics.

INTRODUCTION

The Computer Based Assessment (CBA) systems are, currently, extremely useful tools in education, helping to reduce the distance between students and teachers and providing better methods for monitoring the learning process. These systems can play a particularly important role in programming courses, due to students needing to solve a large number of exercises while learning from their practice.

The overall learning process can be improved by the use of an automated tool, since it can present to the students the feedback they need to refine their programming skill. Moreover, teachers will benefit from not having to grade each assignment manually and replying with individual feedback, in addition to easing the process of making assessments.

GOALS

The main goal of this dissertation is the development of a system for the automatic evaluation of assignments and contests concerning programming in Computer Engineering Education. Using the advanced modules implemented in the system, it is possible to outline the static analysis of code, detect plagiarised code, evaluate the software metrics for the submitted code, resulting into a more intuitive system for training as well as self learning of programming skills.

RELATED WORK

Computer Based Assessment (CBA) systems are used in a number of universities around the world. Two other CBA systems were analysed in order to explore their functionalities and implementation methods: Mooshak and XLX [1]. Of the two automated judging systems taken into account: Mooshak and XLX, the former is the basis of the developed system. Concerning more specifically the plagiarism detection, some existing tools were studied in order to analyse the performance of different algorithms: SIM, MOSS, YAP3 and JPlag [4].

METHODOLOGY

A strategic methodology was followed in the implementation of the system. Its details are described in the following sections:

5.1. Integration of Advanced Functionalities in CBA System

Following the conclusions obtained from the analysis of the previously stated CBA systems, external tools and algorithms,

the advanced modules to be developed and their specifications were clearly identified as follows:

- **Plagiarism detection**: the main requirement of this tool is to generate a list of potential plagiarists given a submission list. The available methods for the implementation of this module are the integration of an external tool and the manual implementation of plagiarism detection algorithm such as Sherlock, since it provides the best results overall and there is a large amount of documentation on its variations and finally heuristic determination of the policy of plagiarism-detection;

- **Software Code Benchmarking**: The aim is to evaluate codes based upon certain predefined software metrics. The term software metrics is used as a collective term to describe a wide range of activities concerning measurements of source code.

- **Static analysis of code quality**: Although all metrics covered in the analysis could be implemented, the general rule of thumb is that as few metrics as possible should be used to simplify the interpretation of the results. The key to this module is evaluating the source code for an appropriate number of metrics through the balance of quantity and quality.

- **Feedback level configuration**: the feedback system is currently limited, and can be improved by offering the possibility to manually configure on a per exercise basis the output to provide to the students. With the addition of the static code analysis and software benchmarking module, the feedback can be even further configured to display the desired evaluation aspects.

- **Support for assessment creation**: The AASSC system provides support for assessment creation featuring a test creation as well as selection mechanism, where administrator can create customized tests as well as select the most appropriate one based on the intent of implementing the assignment.

5.2. Implementation

A subset of the advanced modules outlined was implemented and some functionalities of the analysed CBA systems were extended. This implementation phase comprised the following steps:

- **Optimizing the AASSC CBA system**: As expected, there were some major setbacks while optimizing the program assessment modules in prevoius CBA systems, probably due to the addition of software benchmarking and static code analysing module;
- **Implementing the static code analysing module**: This functionality required the most effort, since it was developed to support any extension of metrics and programming languages and had the greatest number of implementation steps. The McCabe's cyclomatic complexity [2], Space and Time complexities were the primary focus. A set of style metrics, namely the ones used in Submit! [4], were implemented as well;
- **Adapting a plagiarism detection algorithm to the system**: Sherlock and C Code Plagiarism Detection Algorithm (CCPDA[4]) were the algorithms of choice to be implemented in the AASSC CBA system.

5.3. Results

The main achievements were the implementation and integration of the two advanced modules covered in the other CBA systems' analysis: static analysis of code quality and plagiarism detection. Both of these modules were independently tested and details on how to improve and extend them are provided. The development of additional software metrics and the integration of more programming languages were the focus of the static analysis module. On the other hand, the plagiarism detection module blesses the system with new grading interfaces, easing the implementation of new plagiarism plugins or the improvement of the current algorithm, Sherlock and CCPDA [4].

CONCLUSIONS

The goals of this dissertation is to compile a study on advanced functionalities for CBA systems and implement a system, in order to be ready to play an active role on the courses of Computer Engineering Education. The system, with the contribution of the advanced modules, should be able to help reducing the amount of work needed for marking and grading programming assignments as well as contests, by automating and adding additional parameters to the assessment process. The work developed in this thesis is flexible enough to be extended or integrated in any other CBA system. This system can thus be used in a real working environment, providing it is installed in high capacity servers and the processing is carried out in conducive environments.

FUTURE SCOPE

The System can prove to be a powerful tool for education in the field of computer engineering and education. The future scope for this application includes implementation of varioius modules that would supplement the system for inclusion of more programming languages like C# and Python, which are more popular in today's world of Internet. Also work can be done for including the languages like SQL. The field of Plagiarism detection remains to be the most complicated as well as the most important module for this system in which lots of effort are required as it is more of a topic of artificial intelligence rather than a module which requires enormous coding effort.

REFERENCES

[1] *João Cristóvão Xavier*, "Computer-Based Assessment System for E-Learning applied to Programming Education".

[2] *Shyam R. Chidamber and Chris F.Kemerer*, "A Metrics Suite for Object Oriented Design*"* IEEE Transaction on Software Engineering, June, 1994, pp. 476-492.

[3] *Julien Rentrop, "*Software Metrics as Benchmarks for Source Code Quality of Software Systems".

[4] N. Haritha, M. Bhavani, K. Thammi Reddy, International Journal of Science and Advanced Technology (ISSN 2221-8386), "C Code Plagiarism Detection System", Volume 1 No 5, July, 2011, pp. 198-203.