Available online at www.sciencedirect.com

ScienceDirect

Procedia - Social and Behavioral Sciences 228 (2016) 457 – 461

Procedia
Social and Behavioral Sciences

2nd International Conference on Higher Education Advances, HEAd´16, 21-23 June 2016, València, Spain

The capabilities of automated functional testing of programming assignments

Aleksejs Grocevs[a]*, Natālija Prokofjeva[b]

[a]Riga Technical University, Meza str 1, Riga, LV-1048, Latvia
[b]Riga Technical University, Meza str 1, Riga, LV-1048, Latvia

Abstract

In the emerging world of information technologies, a growing number of students is choosing this specialization for their education. Therefore, the number of homework and laboratory research assignments that should be tested is also growing. The majority of these tasks is based on the necessity to implement some algorithm as a small program. This article discusses the possible solutions to the problem of automated testing of programming laboratory research assignments. The course “Algorithmization and Programming of Solutions” is offered to all the first-year students of The Faculty of Computer Science and Information Technology (~500 students) in Riga Technical University and it provides the students the basics of the algorithmization of computing processes and the technology of program design using Java programming language (the given course and the University will be considered as an example of the implementation of the automated testing). During the course eight laboratory research assignments are planned, where the student has to develop an algorithm, create a program and submit it to the education portal of the University. The VBA test program was designed as one of the solutions, the requirements for each laboratory assignment were determined and the special tests have been created. At some point, however, the VBA offered options were no longer able to meet the requirements, therefore the activities on identifying the requirements for the automation of the whole cycle of programming work reception, testing and evaluation have begun.

© 2016 Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
Peer-review under responsibility of the organizing committee of HEAd´16

Keywords: Automation; Assignment; Testing; Continuous Integration;

* Corresponding author. Tel.: +371-268-036-26; fax: +371-670-895-71.
E-mail address: aleksejs@grocevs.pro

1877-0428 © 2016 Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
Peer-review under responsibility of the organizing committee of HEAd´16
doi:10.1016/j.sbspro.2016.07.070

1. Introduction

The current development of information technologies sets the overall trend of choosing the specialization field for future IT-specialists; as well as the number of applications for IT studies is growing every year. The training programs are also keeping up with trends - if previously Pascal and C were used to explain the examples and the basic elements of programming, now the basics of programming are taught using Python, Java and other top level programming languages. Most of the homework and laboratory research assignments for the courses related to software development, discrete mathematics, algorithmization, artificial intelligence, and many others are used to check the students' mastery level: there are assignments that evolve algorithm implementation in a shape of a small program where the input value(-s) are given and the transformation of these values according to some algorithm is expected as an output. In addition to achieving its main goal the source code of the program is also checked for common mistakes, as well as it is verified for the proper implementation of the task, writing style etc. Therefore, the need to automate the processing of routine tests (if not the checking of the source code itself then at least the testing of the correct input/output data processing) appears. Although some aspects of automation have already been under study, the complete solution of this problem has never been developed completely. In order to do this, it is necessary to consider how rational implementing automation will be and what metrics can be used to describe it, as well as to find out the tools that can aid in increasing its level of compliance with the criteria mentioned above.

2. Choosing the Metrics and Evaluating Implementation Possibilities

In order to get a grade, the student has to implement the required algorithm as a program, submit the program in a binary (compiled) form as well as providing its source code. The professor has to test both the program using a subset of pre-calculated input/output data pairs, and its source code equivalence to the binary representation, including the compilation ability and the absence of errors while executing it. It is imperative to evaluate the factors affecting the whole process and to choose the metrics for intercomparison.

2.1. Identified Metrics

Algorithm implementation validation speed is criteria that reflects how fast can the professor obtain the program and its source code, execute and test it using the predefined test patterns. In the University the student has to upload the result of his work to the "Homework" section of the Moodle education portal, where the professor should run and evaluate it after downloading.

Evaluation quality – even in case of simple tasks, such as "implementing a list sorting algorithm" there might be many border cases which can lead to incorrect execution results, but sometimes may not be checked due to time limitation. When using the automated testing solutions, it is possible to pre-create the large number of different tests that check all the aspects of the performance of the given assessment.

Report preparation – it is important to put the data of successful/unsuccessful test runs together and to inform the student of the result. When using on-line solutions there is a possibility to send a test-passed-successfully notification as soon as an assessment has been uploaded and tested. It is also important to record the summary of all students' assessments in form of a table that can later be used to be uploaded to the University education portal.

Plagiarism check – even if the tasks are relatively similar it is crucial to make sure the source code was written by the student himself and not plagiarized from a colleague. This check should be made once by a professor in order to avoid the code consecutive altering so it can successfully pass the check.

Safety check – the binary representation of the program and its source code are usually tested separately in order to save time, and this is most commonly done in that particular order. Although, it is not always possible to quickly detect the malicious code, which is meant to erase or alter the test results or even the testing system itself, when the solution consists of several files or modules. Therefore, while testing the compiled programs they should be treated as potential malware or viruses that may damage the test environment. Ideally they should be executed in the sandbox environment which ensures the isolation of the potential threat.

Bug fix tracking – if the code has been partially altered (either on the professor's request or during the debugging process) the modified parts of the file are indistinguishable from the previous code in the file. Therefore, the

Download English Version:

https://daneshyari.com/en/article/1107180

Download Persian Version:

https://daneshyari.com/article/1107180

Daneshyari.com