Computers in Human Behavior 31 (2014) 453–460

Contents lists available at SciVerse ScienceDirect

Computers in Human Behavior

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

Student perception and usage of an automated programming assessment tool

Manuel Rubio-Sánchez [a,*], Päivi Kinnunen [b], Cristóbal Pareja-Flores [c], Ángel Velázquez-Iturbide [a]

[a] Departamento de Lenguajes y Sistemas Informáticos I, Universidad Rey Juan Carlos, c/ Tulipán s/n, 28933 Móstoles, Madrid, Spain
[b] Department of Computer Science and Engineering, Aalto University, PO Box 15400, FI-00076 Aalto, Finland
[c] Departamento de Sistemas Informáticos y Computación, Universidad Complutense de Madrid, Avda. Puerta de Hierro s/n, 28040 Madrid, Spain

ARTICLE INFO

Article history:
Available online 30 April 2013

MSC:
97C40
97C80
97U70

Keywords:
Computer science education
Automated assessment system
Mooshak
Online judge

ABSTRACT

Automated assessment systems are gaining popularity within computer programming courses. In this paper we perform an empirical evaluation of Mooshak, an online judge that verifies program correctness, in order to determine its usefulness in classroom settings. In particular, we provide a detailed study on how students use the tool, analyze their opinions and critiques about it, and measure other features like its capability to reduce dropout rates. The experience was carried out within a course on algorithm design and analysis where we collected information through several questionnaires and data generated by the tool during the course. Among the main findings we highlight: (1) the usage of the tool was adequate in relation to students' own testing; (2) its feedback needs to be richer in order to improve its acceptance among students; and (3) there was no statistical evidence to claim Mooshak reduced the dropout rate.
© 2013 Elsevier Ltd. All rights reserved.

1. Introduction

Grading computer programs is a complex task. It is time-consuming, tiresome, and prone to inconsistencies and inaccuracies, especially in large classes due to the diversity of the solutions. While certain programming skills require manual evaluations, grading can be enhanced and simplified with the use of automated assessment systems (Douce, Livingstone, & Orwell, 2005; Ala-Mutka, 2005; Joy, Griffiths, & Boyatt, 2005; Brusilovsky & Higgins, 2005; Ihantola, Ahoniemi, Karavirta, & Seppälä, 2010; Queirós & Leal, 2012), which are capable of measuring several software features automatically such as functionality, efficiency, compilation or runtime errors. Benefits of most of these tools include evaluation objectivity and consistency, immediate feedback, 24-h availability, or increased clarity of student code. However, they also present drawbacks related to plagiarism (Sheard, Dick, Markham, Macdonald, & Walsh, 2002; Chen, 2004), or poor usage, where students do not test their programs properly before submitting them (Edwards, 2003; Chen, 2004).

This paper describes an empirical evaluation of the automated assessment system Mooshak (Leal et al., 2003), which has been used in programming contests and academia. While other studies have applied Mooshak successfully in classroom settings (Leal & Silva, 2010), the literature lacks detailed analyses about the usage of the tool, and how students perceive its helpfulness in learning. Additionally, in Gárcia-Mateos and Fernández-Alemán (2009) and Montoya-Dato, Fernández-Alemán, and Garcia-Mateos (2009) the dropout rate in a first year programming course decreased considerably after using Mooshak. However, it is not clear whether this reduction was due to the assessment system, or to a different teaching methodology that was introduced simultaneously.

In this paper our major goal is to understand the possibilities and drawbacks of using Mooshak in a classroom setting, and to provide ideas for improvement. Therefore, our research questions are: (1) to what extent do students use Mooshak as a substitute for their own testing and debugging; (2) how do students experience the usage of Mooshak and how would they improve the tool; and (3) whether the usage of Mooshak influences the dropout rate.

The contribution of this paper is a detailed analysis of how students perceive Mooshak and how they use it. In particular, we found that most students used the tool appropriately regarding their own testing and debugging. However, despite acknowledging that using Mooshak was a good idea, students did not appreciate the experience as a whole, where the main reported drawback was related to its feedback. Finally, we did not find enough statistical evidence to claim that Mooshak reduced the dropout rate.

* Corresponding author. Tel.: +34 914888286.
E-mail addresses: manuel.rubio@urjc.es (M. Rubio-Sánchez), paivi.kinnunen@aalto.fi (P. Kinnunen), cpareja@sip.ucm.es (C. Pareja-Flores), angel.velazquez@urjc.es (Á. Velázquez-Iturbide).

0747-5632/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.chb.2013.04.001

2. The automated assessment system Mooshak

Mooshak is an online judge that checks whether computer programs have been implemented properly and work correctly. Given a set of predefined instances of some computational problem consisting of input–output pairs, it compiles and runs source code in order to verify whether the program generates the desired outputs given the initial inputs. After testing a submitted program, Mooshak returns a brief feedback message indicating if it has produced correct outputs for all of the instances (A: accepted), or whether it causes some error (PE: presentation error; WA: wrong answer; RE: runtime error; CTE: compile time error; IF: invalid function; TLE: time limit exceeded).[1]

Despite being initially designed for use in programming contests,[2] Mooshak has been successfully applied in computer programming courses. It supports a variety of programming languages such as C/C++, Java, or Pascal, which is appropriate when students have different programming experience. However, allowing several languages can limit the use of anti-plagiarism software (since these tools generally do not compare codes written in different languages). In Mooshak the inputs and outputs that define the test sets are specified through text files. The judge compares each character of the desired outputs with those obtained by students' programs on particular inputs, see also (Reek, 1989). Other systems like CourseMarker (Higgins, Hegazy, Symeonidis, & Tsintsifas, 2003) or HoGG use more sophisticated strategies like pattern matching or regular expressions to allow a greater flexibility for the outputs. Students submit only one file to the server per exercise. Thus, if an assignment needs to use several files these can be compressed into a single one, which can later be processed by some script (Montoya-Dato et al., 2009). Finally, after each submission the feedback message (which is often instantaneous) together with the identity of the corresponding student is publicly shown on a list.

3. Methods

In order to answer our research questions, we carried out an exploratory analysis related to data collected from questionnaires and information provided by Mooshak, within an algorithm design and analysis course.

3.1. Algorithm design and analysis course

We used Mooshak in a 13-week course on algorithm design and analysis, for 55 (6 female and 49 male) second year students (where the median age[3] was 19.5, with mode 19, and standard deviation 1.47) of a computer science degree at Universidad Rey Juan Carlos (Madrid, Spain) in 2010. About one third of the course was dedicated to computational complexity, while the rest covered design paradigms such as divide and conquer, greedy algorithms, dynamic programming, and backtracking. It comprised seven programming exercises arranged in three assignments related each paradigm except the greedy approach. Students used recursive or divide and conquer strategies to solve the first four exercises: power in logarithmic time, root finding, matrix block multiplication, and a variant of merge-sort. The fifth and sixth exercises consisted in implementing the pseudocode described in Cormen, Stein, Rivest, and Leiserson (2003) for the top-down and bottom-up approaches for the "Assembly-line scheduling" problem related to dynamic programming. Finally, the seventh exercise involved creating a Sudoku solver by using backtracking.

We introduced Mooshak in a 1 h lecture since students did not have any previous experience with it (or any other automated assessment system). We also dedicated a 2 h lab class to Mooshak where students implemented and submitted very basic programs, but with which they could observe the various types of error messages the judge returns.

3.2. Research design

We collected data associated with the questionnaires described in Rubio-Sánchez, Kinnunen, Pareja-Flores, and Ángel Velázquez-Iturbide (2012) and Supplemental material, which were filled in during the course and would grant students an extra 5% on their final grade. A first 14-item questionnaire was aimed at acquiring background information regarding their previous grades and programming skills. A second larger 64-item (pre-test) questionnaire was developed to measure variables related to background knowledge, self-efficacy, goal orientation and motivation, study habits, and feedback. Both questionnaires were filled in at the beginning of the course by 52 students, prior to any exposure to Mooshak. Three of the items were boolean (q1, q2, and q51), while the remaining 61 were measured on a 5-point Likert scale (1: strongly disagree, 5: strongly agree). At the end of the course the 64-item (post-test) questionnaire was filled in again by 34 students (who had all filled in the initial questionnaires, see Section 4.1 regarding dropout rates), together with another 25-item questionnaire that we elaborated in order to measure students' opinion and acceptance of Mooshak[4]. This last questionnaire was entirely based on the same 5-point Likert scale, except for an extra open question that asked students how the tool could be enhanced.

3.3. Grades and submission information from Mooshak

From a research point of view, a distinguishing feature of Mooshak is the possibility to export data related to its usage (in XML or tab-separated text files) that contain a wide array of information regarding submissions. Thus, we were able to automatically gather data associated with submission dates or the type of feedback messages, and compute related variables such as average rankings on exercises (according to the date of the first accepted submission), the number of submissions until a first acceptance is obtained, or the average number of errors. In particular, this data was collected for 51 students that used the judge at some point during the course.

Finally, we have also included in our analyses the grade on homeworks and on the algorithm design exercises of the final exam.

4. Results

4.1. Dropout rates

In the Spanish public academic system students have two opportunities to pass a course. If they do not pass at the end of a semester they are allowed to take new exams and hand in late or failed assignments at the end of a regular school year. In this scenario, students rarely drop out of courses officially. Therefore, we consider both students that do not hand in any assignments for credit nor take exams (early dropouts), and the ones that having submitted at least one homework during the semester do not take the final exam (late dropouts). Table 1 shows the percentages and

[1] There are other possible feedback messages, but these are the only ones we have observed in our data.

[2] Mooshak has been used in ACM-ICP regional contests (SWERC), IOI, or IEEEXtreme.

[3] Computed for 52 students that filled in a background information questionnaire (see Section 3.2).

[4] We did not find validated scales in the literature for our needs.

Download English Version:

https://daneshyari.com/en/article/6839541

Download Persian Version:

https://daneshyari.com/article/6839541

Daneshyari.com