Computer Languages, Systems & Structures 53 (2018) 59–72

Contents lists available at ScienceDirect

Computer Languages, Systems & Structures

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

Automatic assessment of Java code☆

David Insa*, Josep Silva

Departamento de Sistemas Informáticos y Computación, Universitat Politècnica de València, Camino de Vera s/n, Valencia 46022, Spain

ARTICLE INFO

Article history:
Received 13 July 2017
Revised 15 January 2018
Accepted 17 January 2018

MSC:
97Q70
97P40

Keywords:
Assessment
Java

ABSTRACT

Assessment is an integral part of education often used to evaluate students, but also to provide them with feedback. It is essential to ensure that assessment is fair, objective, and equally applied to all students. This holds, for instance, in multiple-choice tests, but, unfortunately, it is not ensured in the assessment of source code, which is still a manual and error-prone task. In this paper, we present JavAssess , a Java library with an API composed of around 200 methods to automatically inspect, test, mark, and correct Java code. It can be used to produce both black-box (based on output comparison) and white-box (based on the internal properties of the code) assessment tools. This means that it allows for marking the code even if it is only partially correct. We describe the library, how to use it, and we provide a complete example to automatically mark and correct a student's code. We also report the use of this system in a real university context to compare manual and automatic assessment in university courses. The study reports the average error in the marks produced by teachers when assessing source code manually, and it shows that the system automatically assesses around 50% of the work.

1. Introduction

Assessment is a fundamental part of education, and it is useful both for students, who receive diagnostic feedback about their learning process, and for teachers, who can determine whether or not the goals of education are being met. In the psychology education area, it has been proved that most students often direct their efforts based on what is assessed and how it affects the final course mark (see, e.g., [5], chapter 9). As a consequence, continuous assessment during a course can be used to direct and enhance the learning process. However, providing quality manual assessment for even a small class requires an important effort. When the size of the class grows, the amount of assessed work has to be limited or rationalized in some way.

Moreover, interiorizing all the assessment criteria for all possible situations is almost impossible, and, in practice, two teachers of the same subject very rarely apply the same assessment criteria in all cases. This is clearly unfair, because it means that a student's mark can depend on the teacher who assesses their solution, and not only on the student's

☆ This work has been partially supported by MINECO/AEI/FEDER (EU) under grants TIN2013-44742-C4-1-R and TIN2016-76843-C4-1-R, by the *Generalitat Valenciana* under grant PROMETEO-II/2015/013 (SmartLogic), and by the Universitat Politècnica de València under grant PIME B18 and EICE HEGEA.

* Corresponding author.
E-mail addresses: dinsa@dsic.upv.es (D. Insa), jsilva@dsic.upv.es (J. Silva).
URL: http://www.dsic.upv.es/~dinsa/ (D. Insa), http://www.dsic.upv.es/~jsilva/ (J. Silva)

https://doi.org/10.1016/j.cl.2018.01.004
1477-8424/

		Automatic Assessment		Manual Assessment		Difference	
	#Students	Average Mark	Standard Deviation	Average Mark	Standard Deviation	%	Absolute
Exam 1	129	6,49	3,71	6,66	3,77	2,62%	0,27
Exam 2	129	3,88	3,47	3,99	3,26	2,84%	0,23
Exam 3	123	5,7	2,97	4,45	3,09	-21,93%	1,52
Exam 4	63	7,25	2,23	7,39	2,37	1,93%	0,49
Exam 5	31	5,24	3,36	5,31	3,29	1,34%	0,15
Exam 6	60	6,02	3,18	6,28	3,03	4,32%	0,29
Total/Average	535	5,76	3,15	5,68	3,14	5,83%	0,49

Fig. 1. Comparison of automatic and manual assessment of real university Java exams.

solution itself. This happens, e.g., in the assessment of programming exercises. Often, there exist infinite possible solutions to the same problem because there can be variations in the programming style, the documentation, the functionality, the efficiency, the maintainability of the code, etc. Therefore, an assessment template can provide guidelines, but often, it cannot be exhaustive. This leaves room for interpretation.

1.1. Manual versus automatic assessment

We measured the errors made by teachers when assessing Java exercises. For that, based on the library presented in this paper, we implemented a tool able to automatically and precisely mark some specific properties of the code. A property is a requirement an exercise must fulfil (e.g., “a class must extend another class”, “a field must be private”, etc.). With this tool, we carried out an experiment in which we marked several Java exams of real university programming courses at Universitat Politècnica de València . Concretely, the experiment was performed in a second-year Java course whose curricula includes inheritance, abstract classes, interfaces, packages, polymorphism, etc., and whose exams consist in the development of a class model with around 10 classes. The experiment collected data from two academic years: 5 teachers (not ourselves) manually, as usual, marked three exams along the first year, and three exams along the second year. 535 students participated (381 the first year, and 154 the second year). Then, we marked the exams with our tool and compared the results. They are summarized in Fig. 1.

In the table, each row represents a different exam. The meaning of the columns is the following: Column `#Students` contains the amount of assessed exams. Column `Average Mark` contains the average mark of each exam (this is shown for both the manual and the automatic assessment). The standard deviations of the marks are shown in column `Standard Deviation`. Finally, column `Difference` compares both average marks, where % shows the difference in the final mark between the manual and the automatic assessment. The total (absolute) error per exam is shown in column `Absolute`.

To the best of our knowledge, this is the first study that compares manual and automatic assessments of programming exercises. The study is large enough (2 subjects, 5 teachers, 535 students, 6 exams) to produce statistically valid results. The error shown in column % is the observable error made in the manual assessment, but it hides the real total amount of errors made by the teachers. To compute the total absolute error (0.49 points, in absolute value, out of 10) it is important to consider that errors made by teachers were 75% of the times positive (they benefitted the student) and 25% of the times negative (they harmed the student), thus many times they compensated themselves. This compensation happens both when marking one single exam, and also when computing the average of different exams. Therefore, the absolute error is computed as the average of the errors made in each exam being the errors an absolute value.

We studied with the teachers the causes of these marking errors. In almost all cases, they were due to teachers' mistakes in the manual assessment, and in a few cases they were due to small differences in the interpretation of the assessment criteria (e.g., unspecified details that were penalized with -0.1 the first time and with -0.2 the second time).

The errors made in the manual assessment were due to:

1. Wrong code introduced by the students in classes not involved in the exercise (and thus, not revised by the teacher and not penalized),
2. type errors not affecting the result,
3. incorrect use of interfaces,
4. code that is correct but it is marked as wrong because it is surrounded by wrong code,
5. messy code very difficult to understand even though it is correct,