4th International Conference on Higher Education Advances (HEAd'18)
Universitat Politècnica de València, València, 2018
DOI: http://dx.doi.org/10.4995/HEAd18.2018.8132

Information from Automated Evaluation in an Engineering School

Serrano, Nicolas[a]; Blanco, Carmen[b]; Carias, Francisco[a] and Reina, Enrique[c]
[a]Department of Industrial Management, TECNUN Escuela de Ingenieros, University of Navarra, Spain, [b]Department of Biomedical Engineering and Science, TECNUN Escuela de Ingenieros, University of Navarra, Spain, [c]Information Systems, TECNUN Escuela de Ingenieros, University of Navarra, Spain.

Abstract

The paper introduces the need for automated evaluation and presents the experience of automating all the evaluations of a course in Computer Science in the sophomore year of an engineering degree.

First, the paper describes the features needed and developed for that course and the positive results for both professors and students. The main advantage of automated evaluation is that it allows real continuous grading for all types of activities: short answers and exercises during the class, homework, short exercises evaluated every 10 days in class, medium term evaluations and the final grade for the course.

A significant benefit of this practice is that it allows the professor, from the very beginning of the course, to monitor how the students perform each task. The professor can see in real time the marks of an exercise or evaluation, the global evolution of the class or the status of a specific student. The students also have immediate feedback from their exercises and the total points obtained at any given time providing greater involvement in the course.

Keywords: *automated evaluation, computer science, grading, self-assessment*

1. Introduction

An important element of the education process is the assessment of the students. This is because assessment is not only the end result for the student, but an indicator of the efficiency of the learning process (Williamson et al. 2006, Greiff et al. 2017). So, the sooner, and more frequently, the professor and the students know the value of this indicator, the quicker and more effectively they can act on the process.

In addition, in Europe, the European Higher Education Area and the Bologna Process promote a continuous assessment system. This continuous assessment produces an increase in the professor's workload and, as a result, requires more resources or a decrease in the number of evaluations done.

To solve this problem, a significant amount of tools have been developed for automated evaluation of students (Mittal & Devi 2012) (Alta-Mutka 2015). These tools can solve this problem and provide other benefits, such as immediate feedback for both the student and the professor.

However, the types of questions provided by such tools are not adequate for the complete assessment of the students in engineering subjects. For example, few of them allow introducing and evaluating mathematical expressions or the evaluation of algorithms written by the students, and there are not tools that include all these elements.

There are also other issues, like the lack or difficulty of connection with the academic management system and the not simple process to start writing and editing questions and tests.

The teachers of Computer Science and Mathematics of the School of Engineering of the University of Navarra wanted to essay a continuous evaluation program of nearly all the activities of a university course. With the experience of previous academic and teaching applications (Kaushal and Singh, 2012) (Manev et al. 2009), a system called “Iquest” was developed for this automated evaluation process by the professor of the course in Tecnun, the School of Engineering of University of Navarra. This paper studies the use of this application in a Computer Science course during the sophomore year of an engineering degree.

2. A tool for automated evaluation

The main goal of the tool was that it must be able to evaluate all the students activities in the subject. These activities include answering optional questions, programming a simulated microprocessor, programming in JavaScript and Java, and writing code HTML and Java programs that create HTML pages.

With this purpose in mind the Iquest application was developed with a simple interface that allowed a quick professor interaction to define new questions or activate and deactivate the questions for the students.

Figure 1 shows the main window of the application for the professor with a list of the defined questions. From this list the professor can activate or deactivate a question for the students (column Active), define if the question must be graded at the time (column Grade), if the grade must be shown at the time (column Show grade), go to the edition of a specific question (column Edit), execute of the grade (or regrade a specific question for all the students) (column Grade), show a preliminary view of the question (column Show) or show the summary of marks or the answer for a specific question (columns Results).

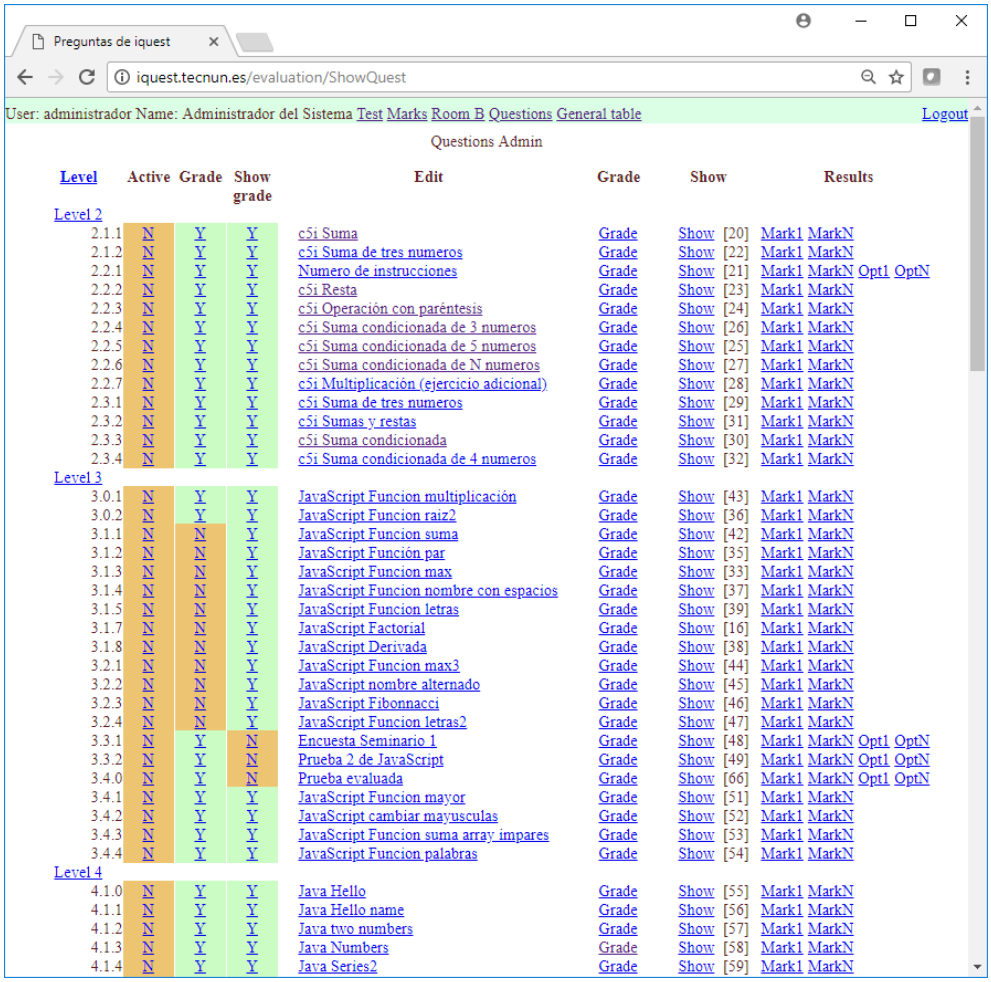

Figure 1. Management of the course questions.

This is the usual process: the professor creates the questions based on previous available exercises in the course and enables them for automatic evaluation. In the class the professor activates the desired questions and can subsequently monitor the results during the test.

The edition of a question is done with the editor shown in figure 2

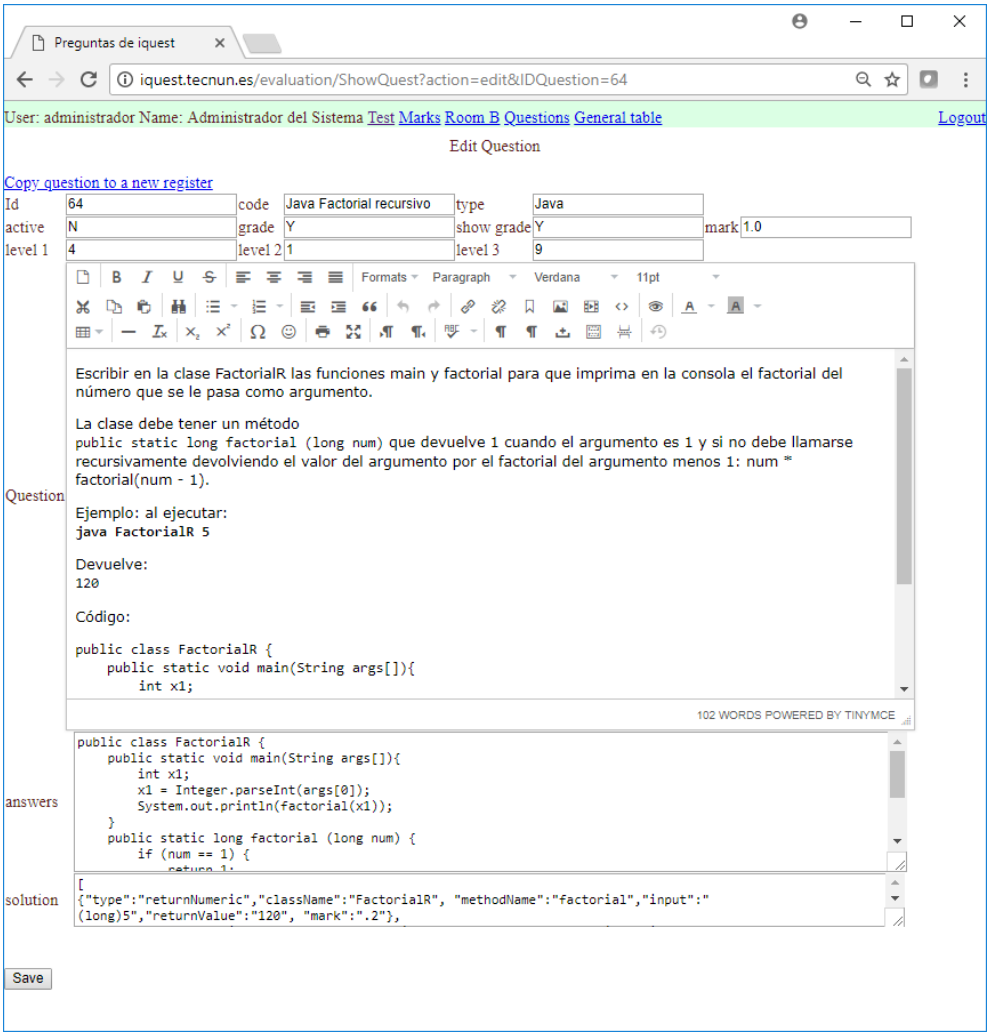

Figure 2. Edition of a question.

The editor has fields for question management, a rich web editor to type them and fields to define the answers and the question assessment method.

The student view is simpler. When the student logs in, the application shows the available activated questions for the course and can introduce the answer and check the results.

Figure 3 shows a view with an active question where the student must introduce a program written in a pseudo assembly language. When the student saves the answer, the system evaluates the program by running a specific script for this type of question and comparing the expected solution with the result of executing the student's code, and then shows the mark to the student. The student can test it again, clicking the edit button and introducing a new answer while the question is still active.

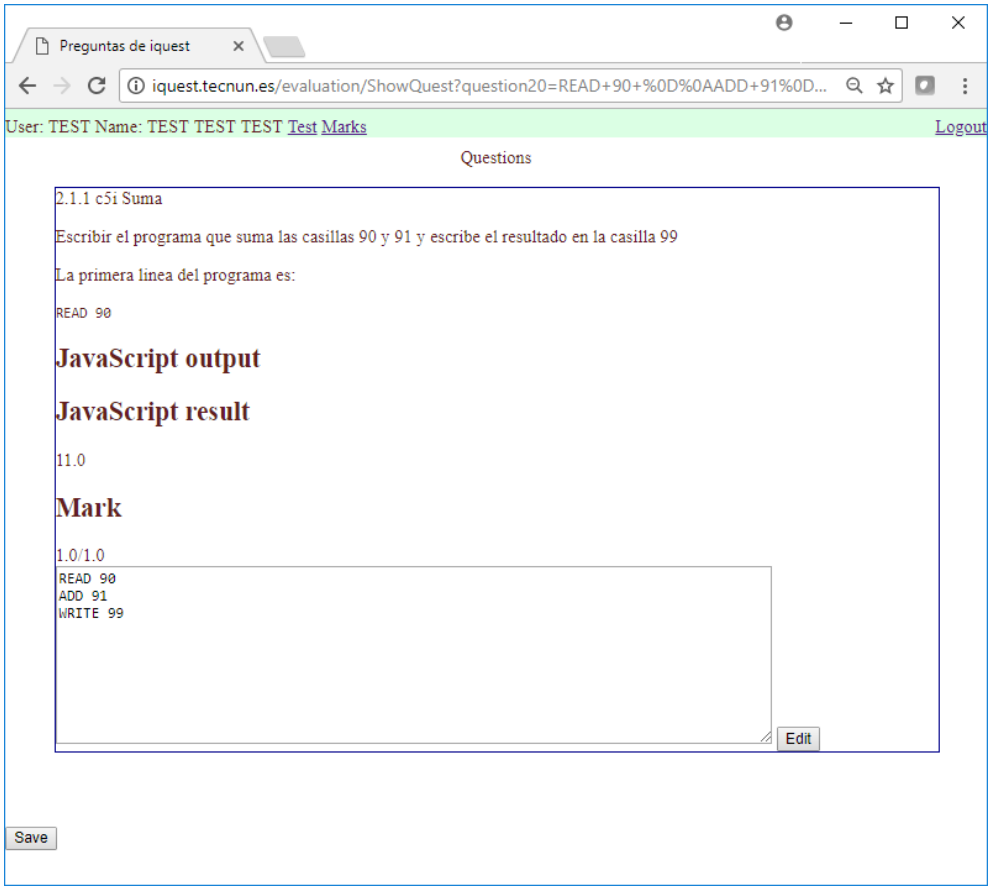

Figure 3. Student view and evaluation of a question.

The student also has another view to see previous questions and their marks.

The platform can also evaluate Mathematical expressions, JavaScript programs, HTML pages, and Java applications. A typical problem with evaluating programming applications is the presence of syntax errors and runtime errors of infinite loops. This application can solve these problems with an architecture based on threads. The thread can be stopped if it doesn't respond after some tenths of a second.

3. Information output from the tool

The use of the platform generates many data. The information is provided to the professors and students through several graphs.

Figure 4 shows partial information of the complete status of the class. Each row corresponds to a student and each column to a question. The different sizes and colors represent the different values and marks.

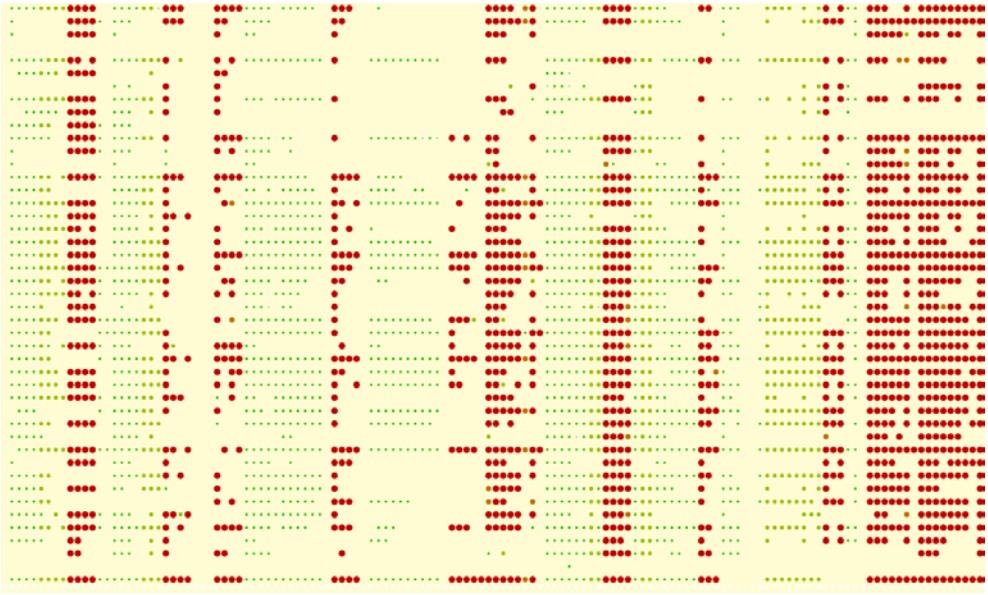

Figure 4. Global information of all students and all evaluated items

With this image, the professor can read in a row the status of a specific student (references have been deleted in the image), in one column appears the difficulty of a specific question and in a group of columns the difficulty and performance of a set of activities or a test.

The tool provides information about the evolution of an exam while the students are doing it. Figure 5 shows the evolution of a three question exam until the minute 52 (horizontal axis shows minutes from the beginning of the test). Each graph corresponds to a question and each blue block to the student's answer. The height of the block represents the mark obtained. When the mouse hovers over one block, the system shows in green all the blocks corresponding to the same student. The figure shows a student who answered the first question at minute 12, the second at minute 30 and the third at minute 32. This graph can also be used to see the performance of the students with their homework or to see all the activities done during the course, as the horizontal axis can be scaled to adequate units (minutes, hours or days).

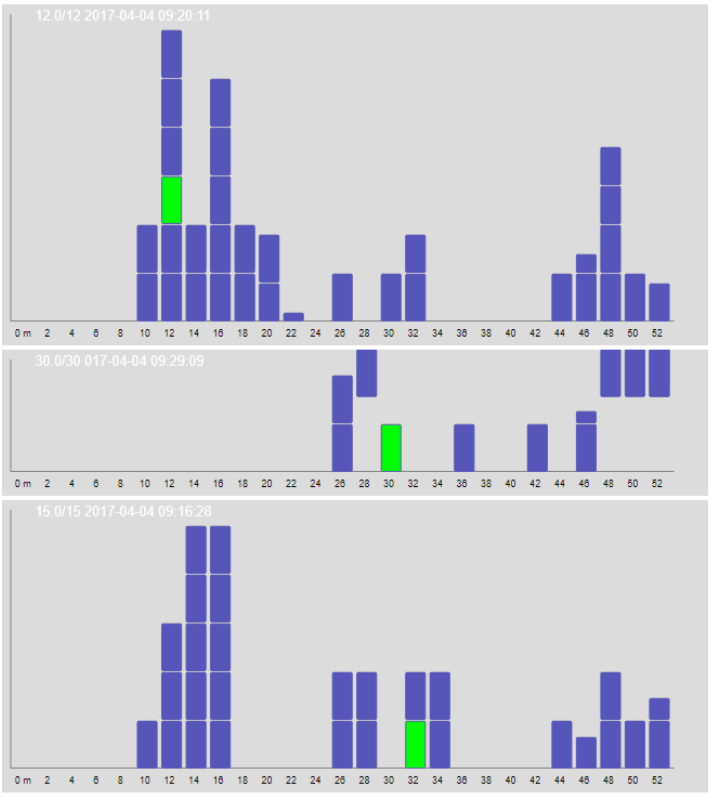

Figure 5. Information during tests.

Additional information of interest is the graph status for a specific student. Professors can see the overall performance with the graph in figure 6. Each block corresponds to a question presented to the student. Blue shows the points obtained by the student in this question and grey the points the student missed.

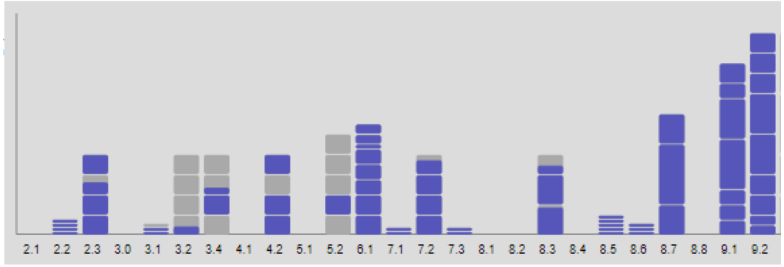

Figure 6. Overall performance of a student.

4. Conclusion

The tool has been used for a complete semester evaluating all the activities of the Computer Science course. These include short answers and exercises during the class, homework exercises and self-assessment, short exercises evaluated every 10 days in class (typically in the first 30 minutes of class), medium term exam and the final exam of the course.

The initial goal of the project was to automate the evaluation to have a continuous evaluation system without an overload of work. The use of the system has proved to be feasible for some courses of engineering, but the main advantage of the system is the information that it provides to both the professor and the student. The professor can see from the first days of the course if the students follow the subject seeing their homework results and evaluations in class. The professor can modify the time invested in each area, reinforcing some points or providing more exercises. The students have an objective measure of their development which works as a kind of gamification in which all activities, when possible, receive a number of points. The result has been quite satisfactory for professors and students alike and it is currently being applied to additional courses.

References

Alta-Mutka, K.M., (2015) A Survey of Automated Assessment Approaches for Programming Assignments. *Computer science education*, 15(2), 83-102.

Greiffa, S., Schererb, R., & Kirschner, P.A. (2017) Some critical reflections on the special issue: Current innovations in computer-based assessments, *Computers in Human Behavior*, 76 715-718

Mittal, H., & Devi, M.S. (2012) Review of Computerized Evaluation Tools in Education. *International Journal of Artificial Intelligence and Computational Research*, 4(2), 111–117

Kaushal, R., & Singh, A. (2012) Automated evaluation of programming assignments. *Conference: Engineering Education: Innovative Practices and Future Trends (AICERA), IEEE*

Manev, K. N., Sredkov, M., & Bogdanov, T. (2009) Grading Systems for Competitions in Programming, *Proceedings of the Thirty Eighth Spring Conference of the Union of Bulgarian Mathematicians, Borovetz, April 1- 5, 2009*

Williamson, D. M., Mislevy, R. J., & Bejar, I. I. (2006). *Automated scoring of complex tasks in computer-based testing.* Mahwah: Erlbaum