# *Automatic Assessment of Programming Assignments to Enable New Educational Paradigms*

José Cardoso, University of Porto, Portugal
João Carlos Pascoal Faria, University of Porto and INESC-TEC, Portugal
Bruno Miguel Carvalhido Lima, University of Porto and INESC-TEC, Portugal

The European Conference on Education 2017
Official Conference Proceedings

**Abstract**
Automating the assessment of programming assignments in higher education institutions is important to provide prompt feedback to the students, reduce teachers' workload on repetitive tasks, avoid human errors, and enable the exploration of new educational paradigms such as gamification and course adaptation based on learning analytics. However, the automatic assessment of programming assignments is challenging because of the variety of programming languages, the variety of assessment strategies, the difficulty to assess quality attributes beyond functional correctness, and the need to integrate with e-learning and students management platforms. There are several platforms for automatic assessment that are used namely in programming contests, but that support only one assessment strategy or do not integrate with students management platforms. To overcome those limitations, the authors, from the Faculty of Engineering of the University of Porto, developed an extensible web based platform for the automatic evaluation of programming assignments, integrated with the students management platform, and supporting multiple programming languages (ranging from Assembly to Java) and assessment strategies (input/out, API), as well as gamification and analytics features. The platform, in a controlled and secure environment (protected against malicious code, infinite loops, etc.), executes the code submitted by students against test suites submitted by the teacher, reporting the results to the students and relevant statistics to the teachers. The platform was successfully applied in real class environment, involving 340 students from two different courses, significantly reducing the time for feedback and teachers workload as compared to previous editions.

Keywords: Education, Programming, Automatic Assessment, Gamification, E-learning, Learning Analitycs

## Introduction

In recent times, we have witnessed the increasing use of e-learning platforms and their tools have assumed an increasingly active role in Portuguese teaching. Not only have teachers been giving preference to these tools in that they allow a more didactic and interesting environment to the student of today and emphasize learning through technology.

E-learning tools are characterized by their distance learning model based on an online web browser platform. This teaching method has a wide range of platforms, such as Moodle, FEUP's platform of choice, which has multiple forms of teaching support, such as online text-based exercises, message exchange with teachers, etc. Most of these tools support the addition of new plug-in functionality.

In the case of Portuguese higher education, namely at the University of Porto, Moodle has become a very important platform, being obvious the immense advantages that this brings to the support of teaching, providing resources of document management, provision of curricular contents, communication between students and teachers, creation of quizzes and platform for the resolution of exams and exercises. With regard to the teaching of curricular units that involve programming, there is also the need and the idea of using tools capable of supporting the development and correction of code, since this is done manually by the teacher.

The creation and promotion of programming competitions also added the need to create and develop tools capable of performing and analyze code and compare the results obtained by the program to the competition as well as the analysis of various metrics and statistics. There are many already developed tools capable of realizing all these functionalities, however, all of them have been developed given a specific need, and, therefore, may not be ideal for correcting exercises in a teaching context nor do they allow the teaching institution to manage The students and teachers.

## Motivation and Goals

In the dissertation we intend to develop a web platform integrated with Moodle, a e-learning platform, capable of assisting the teachers of Object Oriented Programming Laboratory (LPOO) and Microprocessors and Personal Computers (MPCP) of the Integrated Master in Computer Science and Computation of the FEUP in the performance of the tasks of evaluation and correction of the practical exercises of programming of the students of these same Curricular Units (UC).

Through the set of unit tests, provided by the teachers of the above mentioned UCs, it is possible to construct a safe and automatic environment that will allow the students to have their exercise evaluated without the intervention of the teachers.

For this dissertation, it was then proposed the development of a system capable of supporting the teaching of the UCs in FEUP, as it can streamline the whole process of correction of programming exercises as well as provide feedback to students when End of the financial year in question.

This system should seek to be integrated with the Moodle platform, already used and accepted in UP, promoting and maximizing the usefulness of the same. This system must, on the one hand, be able to take advantage of all the advantages that the e-learning platforms offer and at the same time allow to perform the analysis of source code as well as its execution, correction in face of the unit tests and Respective analysis of the output result.

The organization and authentication of students and teachers is one of the main characteristics of the system to be developed and will have to be implemented according to the credentials used in the SIGARRA (Information System for Aggregate Resource Management and Academic Records) also used by Moodle. Another extremely important feature is the management of the exercises in an exam or class context; These will have to be identified by occurrence, UC and examination and have associated with them a battery of tests (to be made available by the professor at the time of exam / exercise submission) in order to validate the quality, correctness and efficiency of the source code Submitted by the students in response to the exercise.

The system to be developed must be prepared for the most extreme moments of stress, that is, with a large number of students submitting their exercises. These moments will be unavoidable given the limited time in the examination context. The system should attempt to send a response within the shortest time interval about the validity of the submission and, if it is the intention of the teacher, of the process of executing the student code presenting the information regarding the correction of the source code in order to facilitate the detection of Mistakes by the student.

The entire process of executing the submitted source code must be carried out in a sandbox environment, where malicious code execution will not compromise the teacher's machine or others present on the network. It is necessary for the system to organize, in an organized way, all the source code and associated battery tests, providing a range of statistics, organized by exercise and by exam, such as: Exercise more times wrong; Battery of tests that failed more times and others still to be defined. The system should allow teachers to export, also in an organized way, the source code, the exam, and the test batteries, including previous occurrences of the Curricular Unit.

**Use of E-learning Platforms Context**

Nowadays, we are inserted in a context that is increasingly dependent on technology; Higher education is no exception, and the paradigms and references concerning the education system and its processes are steadily diverging to an online and technological model

In the case of UP, we see, more and more, the use and support in the tools of e-learning to take a predominant role as it is the case of Moodle. The purpose of this dissertation, as mentioned in the previous chapter, is to develop a system capable of streamlining the entire process of correcting programming exercises and providing feedback to students when the exercise ends. To do this, it is important to analyze which tools allow you to compile and run and evaluate the source code produced by students.

When we want to choose a tool that supports the compilation and execution of source code should be taken into account to define well what we want to obtain from the analysis of the result obtained taking into account several metrics.
These metrics consist of a wide range of points: the comparison of the result obtained with an expected result, the syntax and semantics of the code, the analysis of the source code itself in order to verify if the implemented logical structures respect the intended implementation of the concepts required in the Number of unit tests performed successfully, etc.

**Moodle**

The Moodle is an e-learning platform, Virtual Learning Environment (VLE), developed in the light of Pedagogical principles and thinking about the creation of communities focused on learning, which is constantly evolving, adapting to the new requirements and needs that arise, being the most used platforms by schools and universities, such as the University of Porto, both for online courses and in face-to-face classes.

Moodle is, therefore, a course management system, developed in PHP to create and manage courses in a way online. Its main advantage is that it is open source, allowing any user to modify and make adaptations of the environment according to their own needs, which has resulted in the development of several different objectives.

In addition to the user-developed plugins, this system has a wide range of features implemented natively. Some of these, and that are important for the context of this dissertation, are the submission of works (file submission), download of files made available by the users (teachers or students), elaboration and Multiple choice, true or false and space filling, and implementation with multiple authentication systems (such as SIGARRA in the case of UP).

After all, Moodle has the major disadvantage of not having support for performing source code analysis and execution.

Moodle and other e-learning platforms feature the basic functionality of student and teacher account management, file submission, and exercise creation. Moodle is more advantageous when compared to other tools because it is free and open source, which allows each user to develop custom functionalities. However, all these platforms suffer from the limitation of not allowing code execution.

**Automatic Program Evaluation Tools**

Unlike e-learning platforms, these tools already support the compilation and execution of source code and are used mostly in programming contests as Marcos Kirchner describes in his article.
In programming competitions, competitors must create solutions to the problems presented to them. Each solution is a source code file written in the programming language defined by the contest, which, when compiled, executes a program which, through an input, generates an output. Finally, the program is evaluated by the online contest selection tool and wins the competitor who has the best evaluation \ cite {kosa2005evaluating}.

However, given the nature and mode of use of these tools, one can extrapolate their use to a teaching context in the form of programming exercise brokers in the context of class or examination.

Mooshak, for example, is a web system that allows the management and organization of programming contests, developed by Prof. José Paulo Leal of the Faculty of Science of the University of Porto and is used in several programming competitions, such as the National Olympiad of Informatics. In addition to the use of this tool in the scope of the competitions, there have already been efforts to include the use of this tool in the area of higher education and is already used by some university as a way to support programming classes.

All the researched tools did not meet the requirements planned for the development of this dissertation, since they only allow input / output testing.
The UCs for which this platform is being developed require a very specific testing strategy. In the case of MPCP, the input is given in code form and in the case of LPOO the supplied input is unit tests in JUnit.

MOJO was a tool that already integrated Moodle with some online evaluation tools. This tool promised to be a good starting point for the elaboration of the project proposed for this dissertation since it already had an integration module. However, the idea of using this tool as a starting point was discarded for two reasons: the first consisted of time constraints, since if Moodle had been used, it would be necessary to wait for the servers responsible for managing the platform to place the platform online for the validation phase; The second was related to the discontinuation of the tool, which made it impossible for the authors to support the MOJO experimentation.

**Solution**

In order to overcome the problem presented for this dissertation, a web platform was developed that allows the automatic evaluation of programs developed in an academic environment (in the context of an exam or practical class).
As mentioned the automatic source code evaluation tools are limited to outputs comparison tests, therefore, and as a way of bridging this need for programming languages Java and Assembly, for the LPOO and MPCP, the platform supports, in addition to the simple atomic comparison of outputs, several methods of code evaluation such as executing test batteries provided by teachers, unit tests.
The web system was developed through a set of four technologies: NodeJs, MongoDB, Angular and Java.

In this work, a solution was developed that integrates the client-server architecture and a set of JAR files that constitute the test and automatic evaluation modules. In this way the work developed is basically constituted by two distinct projects, the website and the modules, being that there was a logical separation between both so that, in the future, modules for other programming languages could be developed. The platform architecture is client-server and is described in the figure below:

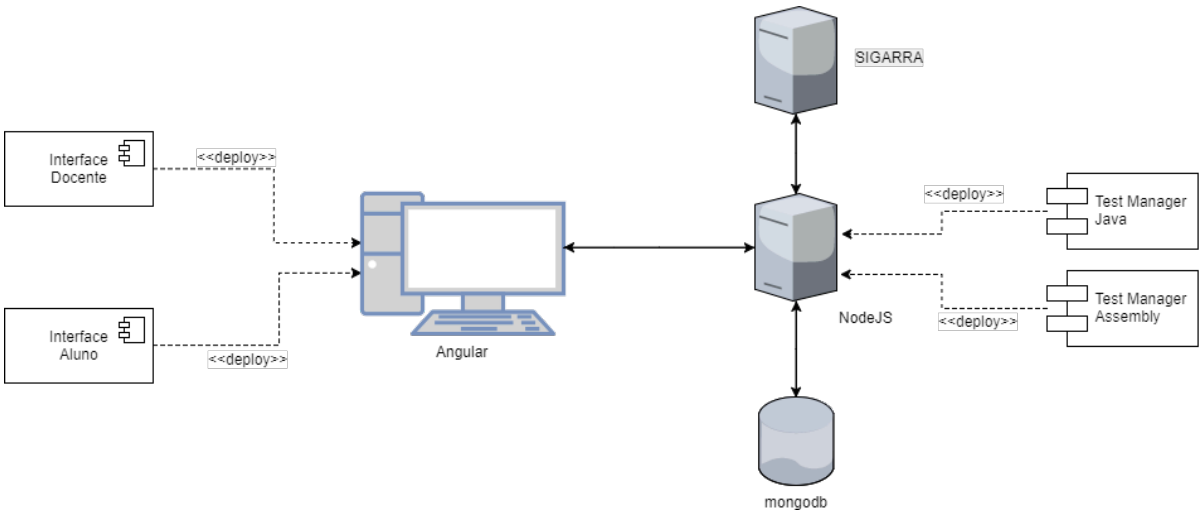


Figure 1: System architecture

The server, developed in NodeJS, stored in a virtual machine on the FEUP network and accessible through the IP 192.168.58.101, constitutes the entire backend of the application that is responsible for executing and managing all requests from the client side through REST requests. In this server are also housed the two automatic code evaluation modules. Authentication is done through a REST request to SIGARRA. The client side is developed in Angular and Typescript having separate interfaces for the teacher and the student, each with different functionalities.

As mentioned, the system includes two different types of users: the teaching user and the student user. Each type of user is authenticated in the system through its institutional credentials, however the features that are allowed to them are different given the nature of their roles. The authentication request, made to SIGARRA, allows you to identify which type of user is, by returning a {'A'} code to identify a student and {'F'} to identify an employee, who in the context This application is a teacher.

The student is an authenticated user in the system and has access to the exams and exercises corresponding to the UCs that he is carrying out with the purpose of submitting the source code. The complete list of features that a student will have in the system is found in the table below.

Table 1: List of features for the Student

| ID | Description |
|---|---|
| R001 | View exam |
| R002 | View the exercise |
| R003 | Submit source code |
| R004 | Edit submission |
| R004a | Perform new submission |
| R004b | Delete Submission |
| R005 | View feedback after submission |

The teacher, also an authenticated user, has an administrator role in the system, that is, he will be able to manage the content that will be made available to the student, delivery dates, conditions, test cases and exercises.

The complete list of functionalities that a teacher will have in the system is found in the table below

Table 1: List of features for the Professor

| ID | Description |
|---|---|
| R001 | Create LPOO exam |
| R002 | Create MPCP exam |
| R003 | Edit exam |
| R004 | Delete exam |
| R005 | Add Exercise |
| R005a | Write statement |
| R005b | Define the exercise mode (exam or exercise) which implies provisioning or not of automatic correction feedback |
| R005c | Define the programming language |
| R005d | Add test cases |
| R00 | Edit Exercise |
| R007 | Delete exercise |
| R008 | Add Student Material |
| R009 | Generate final report |
| R010 | Export exams from previous years |
| R011 | Export source code submitted by the student |

**Platform Utilization**

Initially, the teacher creates an exam, which must submit the exercise with statement and test cases and submit them on the platform. It is also possible to submit some initial code to serve as a starting point for the student to begin elaborating the exercise. In the case of UC MPCP, this material consists of 3 files: a mpcp.inc file with required dependencies, a.asm file with an example input to test your code and a file with the source code of the source code. In the case of LPOO this material consists of a Java project to serve as a basis for development.
After submission, the teacher can export the final report with the grades of all the students who made submissions.

After the creation of the exam by the teacher, the student accesses the system, through a provided url, visualizes the exercise, unloads the material provided and then develops the source code. When finished, it submits it to the system and feedback is provided.

**Automatic Evaluation**

When a source code file is submitted in response to an exercise the server will run the autocorrection module corresponding to the CU of the examination in question.

The server first checks the language corresponding to the exam through the CU by creating a separate thread in order to execute the code in sandbox in a secure way, thus preventing potential problems due to code Malicious or defective. Then it

executes the correction module corresponding to the language. There are 2 different test execution modules, one for Java and one for Assembly.

The Java automatic evaluation module was inspired by a script created by Prof. Nuno Flores. Throughout several meetings the method used was studied and what were the limitations that it offered. One of the problems that was detected was that it was not possible to continue the process when a student's code caused an exception. Hence the present module was developed.

This module consists of a JAR file consisting of 4 classes: BulkProcessing.java, Configuration.java, Main.java, and SystemCommand.java. The Configuration class is responsible for managing system-specific commands. It populates a Hash Table with all the commands that are used in managing and handling files.

The SystemCommand class takes care of assembling the commands and executing them in a secure and controlled way by creating threads whenever a command is executed. The class BulkProcessing is the main class of the module. It is in this class that the whole process is defined and chained, as well as all the control of errors and writing of the final evaluation grid. This module has 2 modes of execution: analysis of a single submission and analysis of the complete set.

First, the module looks in the base directory of the exam for all the .zip files that represent student submissions. Next, a new temporary directory called correction is created for which the student code is extracted and the teacher's test class copied. After that, the code is all compiled and the test class is executed. Finally, a .csv file is generated on which the classifications of each student are written.

The mode of analysis of a single submission is analogous to the first, but is done for each individual student when submitting the source code

The Assembly automatic evaluation module consists of a single class. First, the module looks for all the test cases included by the teacher within the exam base directory. Then the contents of the test case folder are copied to the student directory and the code is then compiled generating a exe file. After the executable is generated, it is executed by redirecting the output to a text file by comparing it with the expected output. This method is repeated for all test cases, finally generating a .csv file in which the classifications of each student are written.

**Validation**

To validate the developed platform, it was conducted 2 different experiments, one for each UC. For LPOO the experiment we selected a group of random submissions from a test of past years, submitted then to the platform to see if we achieve the same classification. For MPCP it was gathered 5 students to resolve an assignment and submit it to the platform, after which an actual professor validated the platforms' results.

After gathering all the results of the two experiments it is possible to conclude that they had a certain success, being able to determine that there is value in the project

developed and that the results were very similar to the manual process used by the teachers.
The platform had a very good adhesion rate and the students present in the experience recognized the value and usefulness in the same considering it an added value in the teaching process.
One of the points to note was the weak feedback of the assembly module and this is a point to improve in the future.

Although these experiences show a promising future for the platform, I recognize that the sample is very small compared to the desired one. This was due to the fact that it was only possible to do this validation during the exam period, which meant that the number of students willing to participate in the experiment was very small.

**Conclusion**

The objective of this dissertation was the development of a web system that allows the analysis and automatic correction of source code developed in response to programming exercises. In order to do this, several existing platforms and tools have been studied that somehow try to fill this need.

In the early stages of development, a number of approaches and technologies were tested that, at first glance, seemed to respond to the needs and requirements outlined, which led to a delay in that not all approaches were compatible with each other, leading to the initial work being redone. However, with the help and advice of my advisors it was possible to come up with a good solution, described in chapter \ ref {chap: chap3}.

The presented solution is in a good state of maturation that allows to respond to the basic needs exposed contemplating all the control and security restrictions, being very evident the success and the promising future of this tool. I think that in the context of this dissertation all the main objectives of automatic evaluation were fulfilled. However, it should be noted that not all lower priority objectives have been met. In addition to the aforementioned reason, the computer attack that occurred in \ Feup \ prevented the agile development of the platform from delaying it for a significant time.

The tests and experiments carried out on the solution allowed to show the viability of the platform in a real context and the adherence to it.

In short, we can recognize that the priority objectives of this dissertation have been successfully met and that it can serve as a proof of concept of the viability of each technology regarding feasibility and the expected purpose.

This dissertation opens up vast possibilities for future work. One of the first steps is to improve the platform in terms of performance and code validation as well as the improvement of the feedback provided in the case of the MPCP module.

An unfinished business is about learning analytics, that is, creating checks of which test cases students are failing most often in order to reinforce these contents in practice classes.

Another point to consider as future work would be the possibility of creating online W3Schools style courses to serve as a form of self-taught learning for students as well as the introduction of Gamification in order to foster the competitive spirit among students in a didactic context.

Finally, an obvious point would be the inclusion of more automatic correction modules in order to cover a greater number of languages and UCs in FEUP.