International Jl. on E-Learning (2011) 10 (1), 27-42.

Integrating an Automatic Judge into an Open Source LMS

KATERINA GEORGOULI
Technological Educational Institute of Athens, Greece
kgeor@teiath.gr

PEDRO GUERREIRO
Universidade do Algarve, Portugal
pjguerreiro@ualg.pt

This paper presents the successful integration of the evaluation engine of Mooshak into the open source learning management system Claroline. Mooshak is an open source online automatic judge that has been used for international and national programming competitions. Although it was originally designed for programming competitions, Mooshak has also been serving as an on-line system for competitive e-learning in a number of programming courses. In order to investigate how it could be integrated into more effective e-learning, thus solving in parallel managerial and communication problems, we incorporated its automatic evaluation engine into Claroline's Assignments tool. The enhanced Assignments tool allows remote evaluation of students' source code submissions in the same e-learning environment where the remaining educational sources and users' data exist. This integrated tool saves time spent by tutors evaluating code and enhances learning in the framework of a well elaborated instructional approach based on automatic judgment of students' programming assignments. Although the integration raised many interesting technical issues, in this paper we concentrate on the usage of the tool, rather than on its internals.

Keywords: automatic judge, e-learning, learning management systems,

INTRODUCTION

E-learning is a medium of instruction adopted by most higher education institutions for supporting their blended learning and distance learning courses. With e-learning, classical pedagogies are integrated with fresh ideas based on the achievements of Information and Communications Technologies (ICT), and this allows new successful educational strategies to flourish.

In modern e-learning environments, on-line tools have an important influence on the design of blended learning scenarios that include mixtures of different teaching strategies. Namely, new interactive communication tools efficiently support cooperative learning; educational computer online games and judges support competitive learning; and online self-assessment questionnaires and computer adaptive testing (CAT) systems support individualistic learning (Economides & Roupas 2007). In this framework, the Learning Management Systems (LMS) become progressively more functional, offering to instructors the tools they need for implementing their preferred pedagogies. Indeed, LMS tools can be made to work together in order to accomplish cooperative, competitive and individualistic educational activities under well elaborated educational scenarios. A lot of research has been focused on studying the effectiveness of ICT support to education, and especially on understanding how LMS tools can contribute to successful blended learning scenarios, that are capable of endorsing different learning e-pedagogies. Johnson and Johnson (1994) suggest that an effective classroom must have the right mix of cooperative learning and competitive learning, along with individualistic learning. Accordingly, instructors combine cooperative and competitive learning approaches to review teaching techniques employed in their classrooms, adopting motivationally balanced teaching approaches (Anderson J., 2006; Bonk et al., 2003; Kolb & Kolb 2005b; Georgouli et al., 2008).

In higher education institutions, a considerable number of attempts have been made to construct applications for automatic marking, both for computer programs written by students of computer science courses and for other forms of assignments. The most common technique for grading text-intensive tests is by matching words and phrases. Most sophisticated automatic essay grading systems, capable of intelligent performance are based on natural language processing (NLP) or artificial neural networks (ANN) (Vallenti et al, 2003).

Systems that automatically assess programming assignments have been designed and used for over forty years. The first generation of those systems was source-code oriented, the second generation was tool-oriented and the current generation is web-based, and appears in the form of independent

applications or as modules to other hypermedia educational applications (Douce et al., 2005).

Most grading tasks in higher education courses can be partially automated using LMS systems, without much effort, but in computers science courses, due to the nature of programming assignments, a further step towards automated evaluation of user submitted programs seems to be only natural (Cheang et al, 2003; English & Siviter, 2000). The idea of automatically evaluating students' programs is commonly followed in international and national programming contests, mainly for saving grading time, and for ensuring impartial and immediate program evaluation. Relevant examples of programming competitions are the International Olympiad in Informatics (IOI)[1] for secondary school students, the ACM International Collegiate Programming Contest (ICPC)[2] , a prestigious contest for teams of university students, the Google CODE JAM[3] a contest having a prize money for the top 100 finalists, the Internet Problem Solving Contest (IPSC)[4], the IEEEXtreme 24 Hour Online Competition[5], etc. Automated judges have been used in programming competitions for around 10 years now. At first, they were prototype applications, not ready for widespread use. These days, a few of them are freely available, for example, PC[2] [6], PKU JudgeOnline[7], DOMJudge[8] and Mooshak[9]. They can be downloaded and installed by all those interested, namely by educators who want to enhance their programming teaching by implementing cooperative and competitive educational strategies.

On the other hand, a considerable number of web-based judges, most of them developed by universities, are available, together with databases of programming problems. Users of these systems select problems from the data base, solve them and then submit their solutions in order to have them immediately checked for correctness in an automatic way. Among the best known systems in this category are the UVA On-line Judge[10], which is an on-line programming trainer, with great pedagogic character, Sphere Online Judge[11], offered also in Polish, Portuguese and Vietnamese, and the Zhejiang University Online Judge[12], one of the earliest and largest online judge systems in China.

1 http://www.ioinformatics.org
2 http://icpc.baylor.edu
3 http://code.google.com/intl/el/codejam.
4 http://ipsc.ksp.sk
5 http://www.ieee.org/xtreme
6 http://www.ecs.csus.edu/pc2/
7 http://acm.pku.edu.cn/JudgeOnline/
8 http://domjudge.sourceforge.net/
9 http://mooshak.dcc.fc.up.pt/
10 http://icpcres.ecs.baylor.edu/onlinejudge
11 http://www.spoj.pl/
12 http://acm.zju.edu.cn/onlinejudge

Nowadays, there are several paradigms of application of online judges and contest systems in higher education. This approach has two main benefits: it decreases the strain of the staff responsible for grading and it offers greater consistency, since all submitted assignments are marked using the same test cases and criteria (Koosowski et al., 2008; Reguers et al, 2008; Boticki et al, 2008; Cheang et al., 2003)..

In this paper, we present the approach we took for enhancing our programming courses using the automatic judge Mooshak, and which lead to the successful integration of the evaluation engine of Mooshak into the open source LMS Claroline[13]. In the section that follows this introduction, we present our experience using Mooshak as a stand-alone on-line application to support programming courses. Then, we discuss the issues related to the use of Mooshak as a new functionality of the Claroline e-learning platform. A more technical section follows for those who are interested on technical issues, where we describe the changes we had to make to the open source code of Claroline, and the technical problems we faced and solved for achieving our goal. After that, a paragraph on related work contextualizes our findings. Finally, we present our conclusions and our ideas for future work.

USING AN AUTOMATIC JUDGE FOR THE ENHANCEMENT OF PROGRAMMING COURSES

Evaluation of student assignments, in any discipline, can be tedious. If done by hand, evaluation of students programs in a programming course is particularly uninteresting and time consuming. First, the students must hand in their source code, by e-mail, for instance. The teaching assistant who is doing the grading must sort the emails, extract the attached programs and file them carefully. Then, he or she must compile the submission, and run some tests. If the program is made up of several files, things become a bit harder. If libraries are used, they must be linked. Of course, scripts can be used to automate this, but they do not work if the students do not follow the rules. If the script does not run as expected, one could simply throw out the submission. However, the teaching assistant is a human being and has feelings: he or she will in most cases try to fix that "small detail" that derailed the script. Likewise, if the program does not compile or one the tests fails "by little", the assistant's tender heart will make him or her peak into the code, to find the cause, fix it and give another try. All this takes time and is a dull job. On the other hand, the main reason for evaluating student assignments is not grading students; it is giving them feedback on their learning effort and on the progress. The feedback must be timely, otherwise it

13 http://www.claroline.org

becomes less useful. But this is almost impossible, if evaluation is done by hand and there are many students and many assignments. Traditionally, the easy way out is to cut on the number of assignments and make the students work in teams (of 2, 3, sometimes more). Actually, working in teams can then be justified as helping students build up their teamwork skills, which, of course, is just a very bad excuse in introductory programming courses.

The solution for this state of affairs is automated judges. For realising our competitive and cooperative pedagogical approach in introductory programming courses we have used Mooshak (Leal & Silva, 2003), a standalone, open source automatic judge system which was originally intended for contests, but it is increasingly being used in programming courses to give instant feedback on practical classes, to receive and validate assignments submissions, to pre-evaluate and mark assignments, etc. (Guerreiro & Georgouli, 2008).

We have been using Mooshak for more than six years now, as a pedagogical tool, and it has proven to be invaluable. Now, the students submit their programs to the judge, and have immediate feedback. The students have to be more rigorous in their programming, because Mooshak is a robot and will not try to make up for small mistakes, as the teaching assistant would have. Mooshak handles automatically all the administrative task formerly done by the assistant: filing the submission, compiling the source files, running the executable against a set of secret test cases, and reporting back the result. If all tests pass, i.e., if the output of each test is equal to the expected output, the submission is accepted. Otherwise, an indication on the nature of the error is presented: wrong answer; time limit exceeded; compile time error; etc.

With Mooshak, the amount of programming done by the students increased several times, and also the courses themselves became more joyful, and also more popular, more transparent and more spoken of. Furthermore, students became more diligent programmers, because automatically enforced deadlines are more difficult to circumvent, and more rigorous programmers, because of the greater effort required perfecting the programs so that they pass all the secret tests held by Mooshak.

The first contact of students with Mooshak tends to be turbulent. Usually students react angrily when Mooshak refuses to accept their solutions, which they could swear were bulletproof. Moreover, at the beginning of the semester, good students are striving for visibility in the class. It must be very disappointing to have your programs rejected and not being able to hide it.

Mooshak was designed for programming contests, with imperative languages, such as Pascal, C, C++ and Java. Still it can be used with functional languages, such as Haskell and Common Lisp, and logic programming

languages, such as Prolog. In all cases, the number of evaluated programs is much larger than what could be achieved using a teaching assistant. In the latest edition of our Programming Fundamentals course, there were 69 problems and exercises in the judge. Not all students submitted everything, and not all had all their submissions accepted, of course, but they all did get a lengthy exposure to the automatic judge.

In this Programming Fundamentals course, Mooshak was one of the pedagogical tools used. The other was the LMS that supports the rest of the educational activities, and runs independently from Mooshak. This did cause some overhead, because these two systems were set up with different usernames and passwords. Furthermore, partial grades obtained in Mooshak were separated from partial grades obtained in other assignments that were done in the LMS. Also, discussion about the problems and the exercises was carried out in the forums at the LMS, not within Mooshak. Ideally, we would prefer a single, integrated environment: it would be less cumbersome for students and much more useful for teachers, if all the pedagogical information of each student could be handled together, in one place.

Since Mooshak was designed for competitions, it has some competitive features that are really not necessary in a pedagogical environment. For example, sets of problems are organized as contests, and those who submit first show up in the first places in the rankings. This has no importance for the purpose of the course. Yet, although we did not stress the competitive aspect of the assignments, many students thrived in it. We observed that many of them enjoyed being on the first places on the ranking and made an effort to solve the problems as soon as possible. As anecdotal evidence on this, at one occasion, we made a mistake when setting up a new contest in Mooshak which caused the ranking to disappear. We immediately received complaints from the students that "without the ranking, it was no fun".

We have been recording the feedback from students, using surveys, at the end of classes, before the exams. Feedback is generally positive. In the latest edition, of our introductory course, with Haskell, we had 115 responses, from more than 90% of the students taking the course. For most questions we used the 5-point agree-disagree scale. Table 1 summarizes the responses to the questions relevant to the present discussion.

In the questions, "platform" means the automatic judge, Mooshak, plus the learning management system that we were using.

We observe that the students are not very assertive, except when giving advice to teachers, as in the forth question. Still, for all questions, more than half of the students either "strongly agree" or "agree". Even in the third question, whose replies reveal that many students are not comfortable with Mooshak, only less than 20% of the students consider that Mooshak is not an effective learning tool.

Table 1
Results of survey, abridged

	Strongly agree	Agree	Neu-tral	Dis-agree	Strongly disagree	Don't know
Programming assignments are more interesting because they are supported by Mooshak	26%	50%	17%	6%	1%	1%
Assessment by the platform is generally fair	15%	62%	18%	4%	1%	1%
With Mooshak we learn more than without Mooshak	20%	35%	25%	11%	7%	2%
Other teachers should be encouraged to use a similar platform	61%	25%	7%	3%	2%	2%

INTEGRATING MOOSHAK INTO CLAROLINE LMS

Mooshak is a software system that acts as a full contest manager as well as an automatic judge for programming contests. It innovates in many aspects: it has a scalable architecture that can be used from small single server contests to complex multi-site contests with simultaneous public online contests and redundancy; it has a robust data management system favouring simple procedures for storing, replicating, backing up data and failure recovery using persistent objects; it has automatic judging capabilities to assist human judges in the evaluation of programs; it has built-in safety measures to prevent users from interfering with the normal progress of contests. Mooshak is an open system implemented on the Linux operating system using the Apache HTTP server and the Tcl scripting language.

On the other hand, Claroline is an open source e-learning and e-working platform written in PHP language, which allows teachers to build effective online courses and to manage learning and collaborative activities on the Web. Each course space provides a list of tools enabling the teacher to publish documents, administer public and private forums, prepare online exercises, publish announcements, set up assignments to be handed in online, view the statistics of user activity, etc. The Claroline platform has been used at the Department of Informatics of the Technological Educational Institute (TEI) of Athens since 2003. It has been constantly customised and enhanced and the resulting LMS, called cs e-Class, constitutes an indispensable part of all educational processes in the department, and is very well accepted by both students and teachers.

Based on the experiences reported in the previous section, we recognized the advantage of linking Mooshak with cs e-Class in a way that registered students logged in a cs e-Class course could have access to Mooshak and

take part to existent contests, and then the evaluation results should get back to students' statistics in cs e-Class. After a closer look into Mooshak's structure, we decided that there were technical obstacles that we could not reasonably circumvent. Because Mooshak is a complete application for managing programming contests, it has its own authentication system. In order for the integration of Mooshak and Claroline to work, we had two choices: either to bypass Mooshak's authentication system or to connect it with the one that comes with cs e-Class. Both of these solutions had to be rejected. Indeed, for the first technique to work we would have to make massive changes in Mooshak's source code, which would require an effort incommensurate with the intended purpose. The second technique would be easier to implement, but it would significantly increase the load on the network, and we could not afford that. We also considered using an LDAP server, since both Mooshak and Claroline support it, but our institution did not have LDAP server that was available at the time. One other difficulty we faced was that the two systems handle very differently the users and their roles: in order to connect these features from one system to the other, we would have changed Mooshak significantly, on this part, and we wanted to refrain from that. At this point, we came up with the idea of isolating Mooshak's evaluation engine from the rest of Mooshak system. This seemed less difficult to pull off, since we would not be invasive and only the necessary code from Mooshak would run during assignments' evaluation, leaving out the modules concerning those competitive features of the Mooshak's system that are not necessary in our pedagogical environment. This solution also facilitates maintenance, when newer versions of the evaluation engine become available.

Isolating the Evaluation Engine

Isolating the evaluation engine within Mooshak took place in several stages. We started by trying to understand the structure of Mooshak's source code. We installed Mooshak in an experimental environment and tried to walk through its source code, reverse engineering its features. For example, we would create contests and submit programs in order to understand which procedures were being called. The next task was to create a script file that would call Mooshak's source code procedures using specific arguments, for example a user id and a file path. Final decision was selecting the tool in the LMS into which the evaluation engine should be integrated. We originally had three possibilities: the Assignments tool, the Exercises tool or the Exams tool. The Assignments tool enables instructors to create and manage students' assignments. The Exercise tool allows you to create online self-assessment exercises composed of a list of questions from different types.

The Exams tool is a homemade tool for cs e-Class that was created from Exercises tool adding free text type questions and enhancing its functionality in order to make it suitable for in-class examinations.

In this respect, the most promising tool was the Assignments tool for the following reasons: a) the Exams tool is a newly constructed tool and it has not been fully tested and adopted by instructors; b) the Exercises tool is not a convenient environment for handling the reports from the judges; c) the Assignments tool is more flexible since the students can easily work from distance as well as in class in collaboration with their peers; d) enhancing a standard tool of the original LMS is an endeavor that can be helpful to the rest of the Claroline community.

In order to create an assignment, the teacher must access this tool and enter the parameters needed: assignment's title and description, starting and ending submission dates, submission rights, etc. When a new assignment is published, students are called to upload their work in the assignments area within the predefined time period. The instructor has to evaluate the submitted essays by selecting the appropriate grade and typing in his or her comments. Students have access only to their own submissions and marks, although they can see who from the rest of the class has already sent a submission.

Adaptation of Claroline Source Code

For entering automatic judge facilities in the existing Assignments tool, we designed a new interface similar to the one already existing for assignments design, with extra fields where instructors must declare the assignment's programming language and enter the test cases containing input data and the expected output data (see Figure 1).

The corresponding environment where students can view the grading has been changed too. For each submission, the automatic judge's report has replaced the instructor's mark and feedback. Moreover, instructors can still upload their own evaluations at the end of the submission period. Also, for boosting competition learning, we decided that evaluation marks should be visible to all class members, something that doesn't happen in the standard Assignments tool.

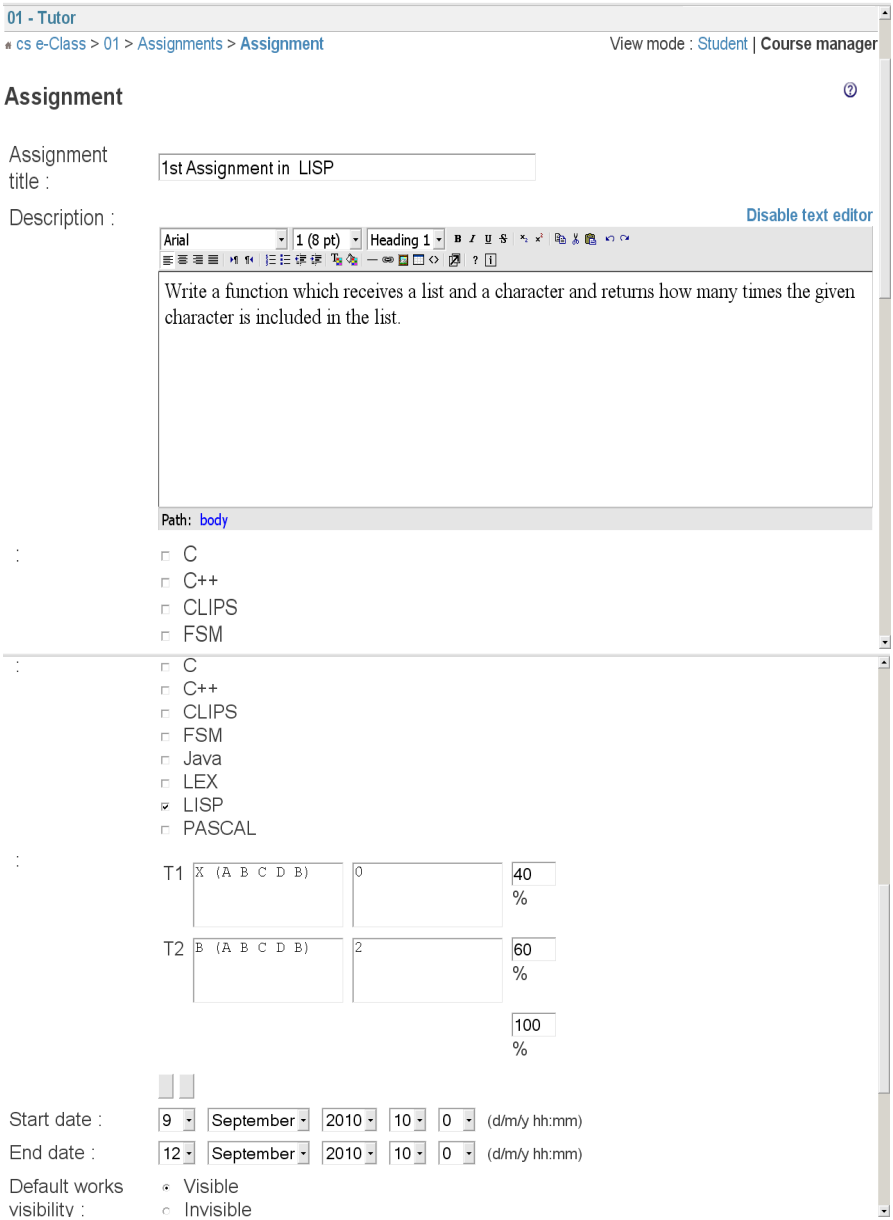

Figure 1. The assignments environment for test cases design.

In addition, changes in Claroline's code were made for managing the programming languages supported by the automatic judge. We created a simple interface which enables the system administrator to create, to process and to delete programming languages. The list of available programming languages is displayed to teachers when creating an Auto Judge Assignment.

Integration of the Two Systems

For the two systems to be integrated, we had to make changes in Claroline's Assignment tool source code, in order to add a new variation of assignments, the auto judge assignment. With this feature, the instructor can create a programming assignment by inserting the compilation parameters and the test data that the automatic judge needs in order to work correctly. In addition, changes in Claroline's code were made for managing the programming languages supported by the automatic judge. We created a simple interface which enables the system administrator to create, to process and to delete programming languages. The list of available programming languages is displayed to teachers when creating an Auto Judge Assignment.

The original idea for the integration was to use the PHP command *exec*[14] to call the script we used to run the automatic judge functions. However, we discovered that this command, as any other file of executable type, was forbidden by the central server's administration for security reasons. Therefore, we decided to use a secondary server in which we had more privileges and were allowed to use the command *exec*, and run arbitrary source programs. In this server we installed the automatic judge's evaluation engine as well as the compilers and interpreters of the programming languages used at our programming courses.

The communication between the two servers is established using the PHP command curl. This command allowed us to transmit data between the two servers. For example, for the creation of an auto judge assignment, we use curl to send to the judge's server the information about the test cases and the programming language required for each particular assignment. Also, the files that students submit are uploaded to the secondary server, where auto judge is installed, via the command curl. The judge's server sends back to the main server information about the evaluation of the submitted source code. For successful data exchanging the enhancement of curl php code was needed in several points.

14 http://gr.php.net/function.exec

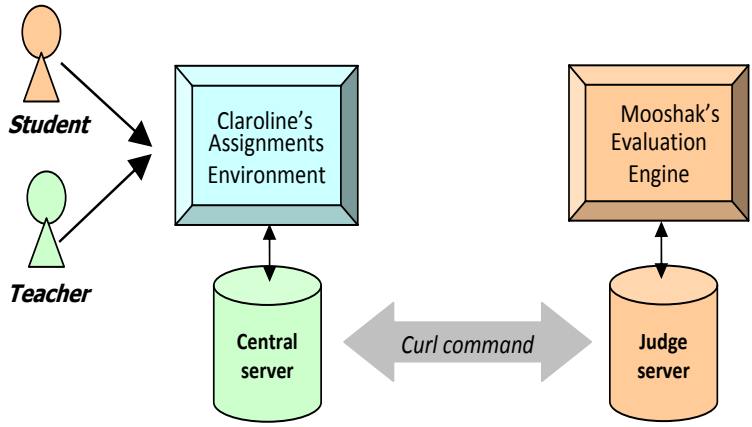

Figure 3. The Integration of Mooshak's evaluation engine with Claroline LMS.

In order to establish the communication bridge between the two servers, we created two PHP files, one in each server. The one in the main server is responsible for the compilation of data and their transmission to the judge's server and it also accepts and handles the information that the judge's server sends back. The file on the secondary server is responsible for retrieving the data that the main server sends, and forwarding them to the automatic judge.

RELATED WORK

A lot of modern online judges and contest systems have been used in higher education the recent years. SPOJ system (Kosewski et al., 2008) has been used for conducting algorithmic programming courses at Gdansk University of Technology. Most courses were accompanied by regular lectures, and some also by classroom exercises like in our case. SPOJ is an in independent platform for programming assignments alleviating much work from the teaching staff and providing a fair and educational programming course for the students. SPOJ system's creators plan to extend its functionality both in the layer of the online judge and web-based E-learning platform.

AHyCo (Adaptive Hymermedia Courseware) learning management system (Boticki et al., 2008) supports a data model for knowledge assessment where new questions for checking the student's programming abilities are

introduced. Questions in this model have corresponding answers in the form of a source program. Evaluation is done automatically through the module of automatic evaluation of programming assignments.

The automated programming assignment grader Online Judge has proved to be a useful tool for three significantly different programming courses at the National University of Singapore (Cheang et al., 2003).

The EduJudge project[15] aims to develop an innovative system based on ICT that can be incorporated into the learning processes in the mathematical and programming field and is addressed to higher education students and and also to secondary education students. The main goal of the EduJudge project is to give a greater pedagogic character to the UVA Online Judge (online-judge.uva.es), which is an online programming trainer and to adapt it to an effective educational environment for higher and secondary education (Regueras et al., 2008). For this reason, the project team promises to integrate the online judge into the open source e-learning platform Moodle and the QUESTOURnament system which is implemented as a module of Moodle. They claim that integration of Online Judge into Moodle/QUESTOURnament seems to be the solution for some of the required pedagogical functionalities for the current system. QUESTOURNnament explores the idea of a variable scoring system, whereby students replying earlier might get more points than students replying later, and the number of points may be a function of the number of wrong answers, as this measures the difficulty of the assignment. This is an interesting idea, but not fully applicable to higher education students, who are encouraged to manage their time, and work mostly outside the classroom. As we write this article we are not aware of any results concerning the integration of UVA Online Judge into Moodle and the QUESTOURnament system.

According to our research, the automatic marker presented by (Suleman, 2008) is the only system that accomplishes the integration of an automatic judge with a modern learning management system. Suleiman (2008) reports on the integration of their experimental automation system for assessing programming assignments with the open source Sakai[16] learning management system. The main difference between this approach and ours is that we have not developed a home-made application for automatic marking. Instead, we relied on the evaluation engine of an already existing open-source automatic judge. Additionally, we tried to use the existing tools on the LMS for registration of users, for submission of files and assignments, and for supporting different kind of communication. This was necessary in order to handle assignments smoothly, and in order to be able to show marks and

15 http://www.edujudge.eu/project_des.html
16 http://www.sakaiproject.org/

statistics in an integrated way. Furthermore it enhanced the global security of the system.

CONCLUSIONS

The successful integration of the evaluation engine of the automatic judge Mooshak into the learning management system Claroline allows us to seamlessly insert competitive and cooperation aspects into our pedagogical approach in teaching introductory programming. The integration significantly increases the effectiveness of our approach, in relation to the previous state of affairs, in which both students and instructors had to constantly move back and forth between the automatic judge and the LMS.

Both Mooshak and Claroline are open source systems, but they have completely different origins and were developed using different technologies. Thus, having them work together required careful consideration of the goals of the project, namely figuring out which subsystem should handle which part of the new functionality. Also, as the LMS is being used across the institutions, changes in its source code had to be guaranteed neither to disrupt the normal operations nor to cause any security risks.

We believe we were able to solve all the technical problems that arose in this project, and that in the end we built an integrated environment where students registered in our programming courses can use the LMS to submit their source code assignments, receiving immediate feedback from the automatic evaluation engine of Mooshak. The evaluation marks and other relevant information are transmitted to other educational tools in the LMS, like the Statistics tool, for creating various reports, and the Marks tool, a home-made tool for bringing together the marks obtained from different assignments.

The enhanced Assignments tool of Claroline has been thoroughly tested in Artificial Intelligent Course during the last semester to support introductory instruction on Common Lisp programming language. This has helped us detect some minor defects in assignments' design environment, and also a few bugs, that have been promptly corrected.

The technical issues that were raised by this project and the solutions that we describe may be useful to other researchers carrying out similar combinations of existing LMSs with automatic judges. Our next challenge is to integrate Mooshak within Moodle, another popular LMS. A further issue that we want to explore is how to restore the spirit of competitive programming that is somewhat downgraded when the evaluation engine is hidden inside the LMS.

A complementary task is to carry out a formal survey to study the immediate impact on our pedagogy that the successful combination of Mooshak

with Claroline has caused. Unfortunately, most of our colleagues, working with programming languages in introductory level, were not so willing to try to enhance their traditional instructional methods adopting the new e-learning tool, a phenomenon common in the majority of similar situations. Moreover, those who were familiar with e-learning tools, mostly young assistants, were enthusiastic with the idea of using this new type of assignments and promised to support our formal survey efforts.

Finally, we are considering other models of program submission that more closely resemble the "real world": instead of requiring a single, monolithic, input-output application, as is the common case now, we would ask for a collection of "services". In this way, students would have to consider explicitly the sub-problems, writing functions for each of those sub-problems, which could be evaluated autonomously by the automatic judge. Therefore, tutors could at least suspect that the programs were structured in a plausible way, without having to inspect the code by hand. Furthermore, using this model, tutors could set up a number of milestones within the assignment, translated to submission intervals in the judge, perhaps with bonuses for early completion and penalties for delays. This would yield a form of variable scoring, adjusting the competitive nature of the programming assignment to more general software engineering concerns.

References

Anderson, R.J. (2006). On Cooperative and Competitive Learning in The Management Classroom, *Mountain Plains Journal of Business and Economics, Pedagogy, 7*.

Bonk, C. J., Wisher, R. A. & Lee, J. Y. (2003). Moderating learner-centered e-learning: Problems and solutions, benefits and implications. In T.S. Roberts (Ed.) *Online collaborative learning: Theory and practice,* Hershey, Pennsylvania: Idea Group Publishing, 54-85.

Boticki, I,. Budiscak, I, & Hoic-Bozic, N, (2008). Module for online assessment in AHyCo learning management system, *Novi Sad Journal of Mathematics, 38* (2), 115-131.

Cheang, B., Kurnia, A., Lim, A., & Oin, W.-C (2003). On automated grading of programming assignments in an academic institution. *Computers and Education, 41*, 121-131.

Douce, C., Livingstone, D., & Orwell, J. (2005). Automatic test-based assessment of programming: A review. *ACM Journal of Educational. Resources in Computing, 5*(3) (Sep. 2005).

Economides, A., & Roupas, C. (2007). Evaluation of computer adaptive testing systems. *International Journal of Web, Web-Based Learning and Teaching Technologies, 2* (1), 70-87.

English, J., & Siviter, P. (2000). Experience with an automatically assessed course. In Proceedings of the 5th Annual SIGCSE/SIGCUE ITiCSE Conference on Innovation and Technology in Computer Science Education, 168-171.

Georgouli, K., Skalkidis, I., & Guerreiro, P. (2008). A framework for introducing e-learning in a traditional course. In: *International Journal of Educational Technology & Society, 11* (2), 227-240.

Guerreiro, P., & Georgouli, K. (2008). Teaching Programming with a Competitive Attitude to Foster Group Spirit. *The Web Information Systems and Technologies (WEBIST) 2008 International conference*, Madeira, Portugal, May, 414-421.

Johnson, D. W., & Johnson, R.T. (1998). *Learning together and alone:cooperative, competitive, and individualistic learning* (fifth edition ed.). Needham Heights, MA: Allyn & Bacon.

Kolb, D.A., & Kolb, A.Y. (2005b). Learning styles and learning spaces: Enhancing experimental learning in higher education. *Academy of Management Learning & Education, 4*(2), 193-212.

Kosowski, A., Malafiejski, M., & Noinski, T.(2008). In . Leung et al. (Eds.): ICWL 2007, LNCS 4823, 343–354.

Leal, J. P., Silva, F. (2003). Mooshak: a Web-based multi-site programming contest system, *Software Practice & Experience, 33*(6), 567-581.

Regueras, L.M., V erdú, E., de Castro, J.P., Pérez, M.A., & Verdú, M.J. (2008). Design of a Distributed and Asynchronous System for Remote Evaluation of Students' Submissions in Competitive E-learning. *Presented at the International Conference on Engineering Education "New Challanges in Engineering Education and Research in the 21st Century"*, Hungury.

Ribeiro, P.,& Guerreiro, P. (2008). Early Introduction of Competitive Programming, *Olympiads in Informatics, 2,* 149-162.

Suleman, H. (2008). Automatic marking with Sakai. In *Proceedings of the 2008 Annual Research Conference of the South African institute of Computer Scientists and information Technologists on IT Research in Developing Countries: Riding the Wave of Technology* (Wilderness, South Africa, October 06 - 08, 2008). SAICSIT '08, vol. 338. ACM, New York, NY, pp 229-236.

Valenti, S, Neri, F., & Cucchiarelli, C. (2003). An Overview of Current Research on Automated Essay Grading, *Journal of Information Technology Education*, vol. 2.