Computers & Education 59 (2012) 535–550

Contents lists available at SciVerse ScienceDirect

Computers & Education

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

A new Internet tool for automatic evaluation in Control Systems and Programming

D. Muñoz de la Peña [a], F. Gómez-Estern [a,*], S. Dormido [b]

[a] Departamento de Ingeniería de Sistemas y Automática, Universidad de Sevilla, Spain
[b] Departamento de Informática y Automática, Universidad Nacional de Educación a Distancia, Madrid, Spain

ARTICLE INFO

Article history:
Received 11 May 2010
Received in revised form
14 November 2011
Accepted 13 December 2011

Keywords:
Automatic evaluation
E-learning
ICT on Education
LCMS

ABSTRACT

In this paper we present a web-based innovative education tool designed for automating the collection, evaluation and error detection in practical exercises assigned to computer programming and control engineering students. By using a student/instructor code-fusion architecture, the conceptual limits of multiple-choice tests are overcome by far. The proposed system is also able to individually parameterize the exercises for each student and allows an instructor to implement innovative self-learning techniques in which student can obtain a measure of their knowledge continuously along the whole course length. The proposed automatic evaluation system has been applied to several control engineering courses at the University of Seville and the results several case studies are presented here.

1. Introduction

The quick expansion of the Internet as a community collaborative tool in all sectors started in the nineties and has now reached maturity. The uses of today's XHTML pages little has to do with the initial, mostly static, contents of that period. Recently, the term Web 2.0 has been coined in order to denote a second generation of the Net, based on user communities which are responsible for providing most part of the content. Team collaboration, file sharing, blogging, product user ratings and forums are popular tools, implemented in content management systems such as Drupal and Joomla. Worldwide education institutions have started using packages such as Moodle and WebCT, that offer all these collaborative tools, plus some additional features, at the service of education. The use of this technological framework in education reduces the student mobility requirements (restraining expenses in education institutions and companies) and enables a closer interaction between teacher and students (using different tools such as forums, e-mail, FAQ, immediate course material delivery, warnings, grades, planning, calendars, etc…). In Kapur and Stillman (1997); Poindexter and Heck (1999); Heck, Poindexter, and García (2000), an overview of the possibilities offered to educators is presented.

In control systems education there has been great interest in web platforms that enable the users to interact with a remote laboratory system, either real (see Cedazo, Lopez, Sanchez, & Sebastian, 2007; Casini, Prattichizzo, & Vicino, 2003; Casini, Prattichizzo, & Vicino, 2004) or simulated (Maussion & Tricot, 2008; Sanchez, Dormido, Pastor, & Morilla, 2004; Sanchez, Morilla, Dormido, Aranda, & Ruiperez, 2006; Torres et al., 2006), extending to the web the crucial aspect of practical educational work in engineering.

However, learning management systems, though very effective in distributing material and collecting the student assignments, offer limited evaluation capabilities, in general reduced to multiple-choice questions (MCQs). The use of MCQs in Engineering is widespread Petridis, Kazarlis, and Kaburlasos (2003); Tartaglia and Tresso (2002) and its convenience is currently under discussion Gentil and Exel (2004). From our perspective, more sophisticated tools specific for evaluating elaborate projects from students are highly desirable. Little effort has been made in order to automatically grade complex assignments while some interesting examples in diverse fields are Jiang, Zhang, and Ye (2009); Jaaskelainen et al. (2009); Gerosa and Narayanan (2008); Chiou, Hwang, and Tseng (2009).

On Computer Science, a number of approaches have been developed whose characteristics include automatic evaluation, see for example Douce, Livingstone, and Orwell (2005); Gutiérrez, Trenas, Ramos, Corbera, and Romero (2010). In Control Engineering, student assignments

* Corresponding author.
E-mail addresses: dmunoz@us.es (D. Muñoz de la Peña), fabio@esi.us.es (F. Gómez-Estern), sdormido@dia.uned.es (S. Dormido).

0360-1315/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compedu.2011.12.016

likely to be automatically graded range from mathematical problems (e.g. eigenvalue problems, state-space matrix analysis), controller parameter tuning, simulation-based work, and algorithm programming exercises. Among the few virtual laboratories that offer automatic evaluation for programming courses we find Rodriguez, Zamorano, Rosales, Dopico, and Pedraza (2007); Rodriguez, Pedraza, Garcia, Rosales, and Mendez (2007); Rodriguez, Pedraza, Dopico, Rosales, and Mendeza (2007); Garcia, Rodriguez, Perez, and Garcia (2009), however, they provide particular solutions that cannot be used in other applications.

Motivated by the above discussion, we have developed a fully novel virtual education platform designed for automating the collection, evaluation and error detection in practical exercises assigned to control engineering students that allows to introduce user (instructor) defined code in some computer language (nowadays Matlab and C language are implemented), to algorithmically evaluate the student's solution. This opens the possibility of functional evaluation of complex engineering designs, such as computer algorithms, controller tuning, control system design analysis exercises and much more. In mathematics there are several web-based systems based on using computer algebra system (CAS) with the internet delivery to provide a sophisticated mechanism with which to grade students' work Klai, Kolokolnikov, and van den Bergh (2000); Sangwin (2004); Mavrikis and Maciocia (2003), albeit do not address the requirements of programming and in general are aimed at designing online courses based on a sequence of short, concise questions whose solution is a function that has to be defined using the appropriate syntax. The platform presented in this paper allows for more complex mathematical (and engineering) exercises in which the student solution is not a single function and moreover, the problem may be personalized for each student beyond a simple randomization. In addition, it allows the implementation not only of traditional assessment, but also dynamic assessment, see for example Wang (2010), based teaching methodologies in engineering courses with a large number of students. In the sequel we will describe the new evaluation tool and analyze a particular experience carried out in a control systems design course at the University of Seville. Many exercises in this field have a common algorithmic structure and a set of numerical specifications that can be automatically verified using our approach.

2. Automatic evaluation paradigms

Most virtual education platforms often include automatic evaluation tools. The most common tool is a multiple selection quiz where the student must answer some question by choosing one or more options from a list of available answers. There are a number of variations around this scheme. For instance, the questions may be chosen randomly from a database, so that no pair of exams is identical or even adapt to the latest responses of the student as proposed in Barla et al. (2010). Moreover, grading schemes offer also some degree of customization. In other available evaluation tools are endowed with some basic arithmetic capability, in order to check that the numerical solution provided by the student is precise.

Our platform is based a totally different evaluation method, which is inspired in the core architecture illustrated in Fig. 1. The key idea is that the students must submit the solution to the practical exercises following a certain programming language syntax (Matlab among other possibilities). The student's code is executed in the server along with code designed by the instructor to evaluate the work of the student. This framework will be denoted *black-box evaluation*. The evaluation code is generated by the instructor satisfying some design constraints, in particular, the order in which the students code and his code are executed/complied and the way in which the mark and possible comments have to be stored so that the server can retrieve them and save them in the database. These design constraints leave freedom to the instructor to design complex exercises and their corresponding evaluation codes using all the possibilities that the chose programming language provides, for example C code libraries and Matlab toolboxes.

Under this general structure, different evaluation paradigms, overcoming the conceptual limits of multiple-choice tests, can be implemented. In the following subsections, we will describe the following paradigms:

- Exact result matching.
  - Individual problems.
  - Cooperative tasks.
- Performance-based evaluation.
  - Specification checking.
  - Competitive evaluation.
- Algorithm design exercises.

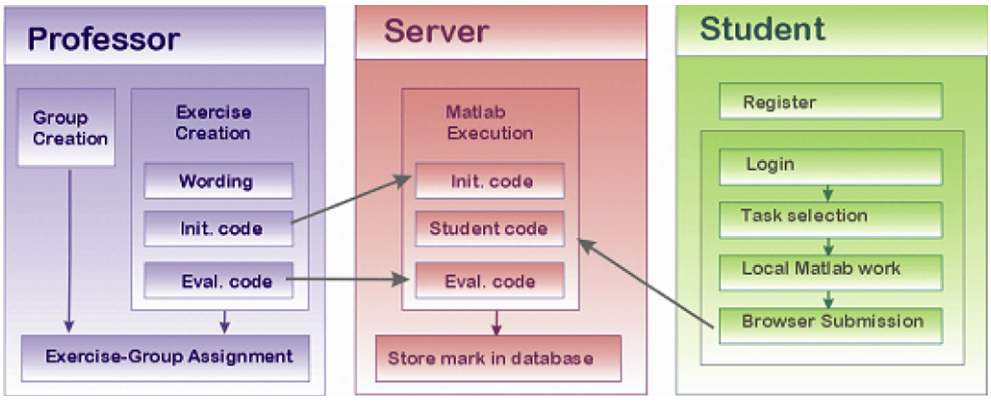

Fig. 1. Workflow for both types of users and their interactions with the system.

In order to accommodate the differences between these paradigms, small variations of the execution sequence depicted in Fig. 6 have been implemented, through a set of user options for customizing the student exercises. This will be further discussed along with specific implementations of evaluation scripts in Section 4.

The claim that this architecture, which carries its own risks and difficulties, is secure and efficient, is based on experimental facts: it has already been put into practice for two years in 6 different courses at the Department of Control System Engineering, giving precise and reliable evaluation results affecting more than 2000 students (see statistics on Section 6.2). It must be noted that, through that period, a considerable amount of fine-tuning and robustness adjustments have been carried out until the present stable implementation.

2.1. Exact result matching

In this paradigm, the students submit the numerical results to a set of problems proposed by the instructor. To this end, the solution must give values to a specific set of variables following the syntax rules of the programming language. Note that the students may be unaware of this fact if a solution template is provided. In order to give the grade, the exercise must contain an automatic evaluation code produced by the instructor. In principle, this paradigm can be implemented using available multiple selection quiz, however, using the proposed black-box approach it is possible to personalize the numerical parameters of the exercises as a function of the digits of a personal unique ID or enrollment number (that may be provided by the instructor). In that case, each student is given a unique problem to solve and they will be unable to share their solutions. This architecture requires an exchange between the database, containing the student ID number, and the evaluation code. In order to illustrate this, the following second order equation problem is proposed.

Q: Provide the solutions of the second order equation $ax^2 + bx + c = 0$ where (a, b, c) are defined by the three first digits of your student ID number. Assign the solutions to variables `sol1` and `sol2` sorted by increasing real part.

To define the solution the student must fill the following template:

```
sol1 = ; % Introduce the first root
sol2 = ; % Introduce the second root
```

In order to check that this results is correct, the server will access the database to retrieve the students ID, define (a, b, c), generate the correct solutions, execute the student code to obtain the students solution, and check whether both solutions are equal (here, a tolerance must be accepted in order to accept different ways of computing the result).

In our example, assuming that the ID is stored in a vector, the instructor's Matlab code could be:

```
sol = roots(ID(1:3));
grade = 0;
if(norm(sol(1)-sol1)<eps)
   grade = grade + 1;
end
if(norm(sol(2)-sol2)<eps)
   grade = grade + 1;
end
```

In this evaluation code, `grade` would take a value between 0 and 2 and `eps` is an error tolerance. The server automatically retrieves the value of the variable `grade` and stores it in the database. Note that in order to execute this code, the variable `ID` must be created in Matlab's workspace. This is done by the server, which access the database, retrieves the `ID` and automatically adds the following code:

```
ID = [x x x x x x x];
```

In this paradigm, the solution can be split in cross-dependent parts (e.g., a C++ piece of source code divided into modules), each of them to be submitted by a different student in a group. Then, the server must merge all parts in order to evaluate the results. This opens the possibility of implementing collaborative schemes.

2.2. Performance-based evaluation

Another paradigm, which opens an interesting set of possibilities, is the one where *the student solution is not unique*. In this case, the students produce a solution that again will be executed in the server as source code (without student's awareness of that fact). However, instead of comparing the generated workspace variables with a pre-specified correct answer, they will be processed by a set of simulations that are parameterized using their values. The simulation outputs will eventually serve to judge the quality of the solution. The evaluation can be performed in different ways:

2.2.1. Specification checking

The results (the time-series of the system output) are analyzed in order to check that the closed-loop requirements specified in the wording are properly met. This will result in a binary evaluation result that is independent for each student.

2.2.2. Competitive evaluation

On top of the previous procedure, a figure of merit obtained from simulations (e.g. closed-loop the rise time in control system design) is stored in the database for each student. Then, the students are sorted in terms of that value, and graded depending on their position within the rest of the group. This is implemented with the variation from the standard execution sequence (Fig. 1) illustrated in Fig. 2.

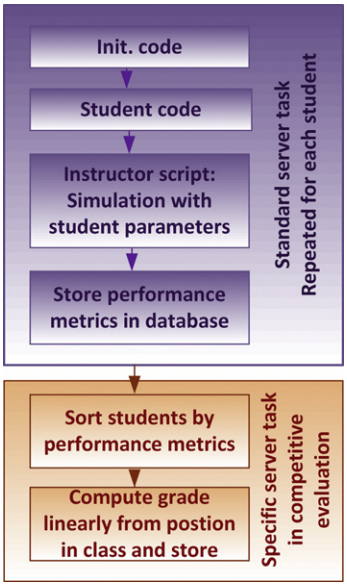

Fig. 2. Server execution sequence in competitive evaluation.

A scenario where we deem it very convenient to make use the proposed architecture for evaluation (either in competitive form or not) would be in controller design and tuning exercises. In this context we propose the following interactive model:

1. The course instructor proposes a transfer function $G(s)$ (with individualized parameters for each student) and a controller structure, for instance a PID or a lead-lag network. Note that the complexity of the system model and the controller structure is by no means bounded by the evaluation platform specifications, which can be used to teach advanced control techniques in post-graduate courses.
2. The instructor proposes some closed-loop specifications to be achieved with the control design, such as steady-state error, gain and phase margins, disturbance rejection, etc.
3. The student uses the methods explained in the theoretical lectures to design a set of controller parameters (such as K_p, T_d, T_i in the case of PID tuning) to satisfy these specifications. The student can verify the results by simulating the closed-loop system via Simulink, command-line commands such as `lsim`) or *Control System Toolbox* commands (e.g. `margin`).
4. The student submits the resulting set of controller parameters to the instructor.
5. The instructor evaluates whether the specifications are fulfilled using the aforementioned tools.

Q: Given the transfer function $G(s)$ corresponding to your ID, design a PI controller to guarantee that the resulting closed-loop system step response has an overshoot lower that 0.2 and a rise time lower than 10 s.

To define the solution the student must fill the following template:

```
Kp = ; % Introduce the proportional gain
Ti = ; % Introduce the integral time
```

Assuming that the transfer function is generated previously from the ID, the instructor's Matlab code could be:

```
c = tf(Kp*[Ti 1],[Ti 0]);
gbc = g*c/(1 + g*c);
[Y,T] = step(gbc);
yrp = Y(lenght(Y));
ymax = max(Y);
OS = (ymax-yrp)/yrp;
tr = T(min(find(Y > yrp)));
if (SO<0.2)
   grade = grade + 1;
end
if (ts<10)
   grade = grade + 1;
end
```

Note that how to generate the transfer function from the ID number is a key issue in the development of this exercise, and in general, it depends on the specifications of the closed-loop system. One option can be to define it from the digits of the ID, for example:

```
g = zpk([], [-ID(1) -ID(2) -ID(3)], 10);
```

Note that in this case, depending on the ID number, it could not be possible to satisfy the specifications. Another option, is to have a list of feasible parameter problems, which are assigned by the instructor to each student. The instruction then would have to add this assignments to the evaluation code, for example:

```
if ID = 10230405
   g = tf([2,[1 4 5 1]);
else if ID = 19471453
   g = tf([2,[1 2 4 2]);
else if...
```

This list would have to take into account all students. Note that there are many different ways to solve the parameter definition problem from the ID obtained from the database by the server. We have just presented two examples.

2.3. Algorithm design exercises

Finally, the proposed *black-box* evaluation procedure allows the instructor to grade algorithm design problems in which the students must design a program in the specified programming language. In some courses, it is not just some parameters of a structured controller what should be evaluated, but a whole algorithm implemented by a student. This case is particularly important for general programming courses and has been applied in the C Language and Matlab programming courses of the University of Seville. In addition, complex controller design procedures can be evaluated for higher complexity post-graduate courses. The idea for automatically evaluating a programming exercises has different approaches; the simplest, but probably most effective one is just executing the student code. If their algorithm is viewed as an information processing function, and in most cases it is so, the automatic evaluator must provide random input data to the function, execute the code, and compare the output with the one generated with an instructor-provided version. In order to illustrate this paradigm, the following C problem is proposed.

Q: Define a function that evaluates the norm of a real number. The prototype of the function is `float absolute(float x);`.

To define the solution the student must fill the following template:

```
float absolute(float x)
{ \\ BEGIN SOLUTION
} \\ END SOLUTION
```

In order to check that this result is correct, instructor code will sequentially execute the student and correct code, and check whether both solutions are equal. The server compiles the instructor code along with the solution of the student in a single program which writes the grade in the standard output, which is read directly by the server to retrieve the grade.

In our example, the following code would be sufficient.

```
#include <stdio.h>
#include <math.h>
float absolute(float x);
void main(void)
{
   int grade = 1; float x;
   for(x=-10; x<10; x++)
      if(absolute(x)!=fabs(x))
      grade = 0;
   printf("%d",grade);
   return;
}
```

It is important to remark, that in this paradigm, personalization of the student exercises is a much more complex issue than in the above mentioned paradigms in which numerical parameters may define an analysis or design problem.

3. Architecture of the proposed solution

At the beginning of our project, some instructors from the field of control engineering were consulted, and we observed that, to a certain extent, many of them used *ad hoc* Matlab scripts (`.m`) or C programs specifically designed to implement to some extent the evaluation paradigms presented in the previous sections. In addition, e-mail or general–purpose virtual education tools were used to collect the work from students. These approaches have been long used by some educators and they present advantages with respect to manual evaluation. However, they can be significantly improved by considering the following issues:

- The technique is commonly developed *ad hoc* by each instructor and it is difficult to share and export tools between different courses.
- There is no simplified and robust mechanism for student work delivery, filtering and tracking.
- There is no automated control of deadlines.
- There is no unique interface for students irrespective of the course and the instructor.

- There is no way of building a stable library of past exercises, including wordings and evaluating algorithms, so that they can be readily reused in future courses and by different instructors.
- Administrative tasks such as mark listings or task assignment to groups are not automated.

To implement the proposed interactive, model we have developed a virtual education platform based on mostly open source technologies. The infrastructure deployed in the Engineering School of the University of Seville consists of:

- Rack server HP Proliant DL-140, 2 GB RAM, 160 Gb hard drive.
- Operating system Windows 2003 Server R2.
- Web server Apache 2.2.6, PHP 5 and database server MySQL Server 5.0 all installed on the same machine.
- Matlab 7.3 installed on the same machine.
- Dev C++ GPL License C/C++ IDE and command-line compiler (for C/C++ programming courses).

The technologies deployed in the server are well known web server design tools: a) PHP scripting language is an open source powerful language and libraries for executing server code as a response of HTTP client requests. This language is the base of most dynamic XHTML applications with database support on the web today. b) Database server MySQL is a TCP/IP based open source SQL relational database server that interacts with PHP scripts. The server database is used to store data from exercise wordings, student lists, exercise lists, student solutions and marks, and customizable evaluation algorithms. c) Apache server is an open source web server that responds to HTTP requests issued at client browsers by generating dynamic XHTML pages after a PHP server preprocessing.

The application engine exploits ActiveX interprocess communication for remotely executing Matlab procedures, hence allowing the use of Matlab syntax to write down the solution of the practical problems proposed. The software has two main parts: the system for collecting the student work, which uses standard procedures similar to Moodle and other well known web education tools, and the evaluation engine, which comprises the sequential execution of three Matlab scripts: initialization (instructor-designed code), the actual exercise solution provided by students, properly formatted into Matlab syntax, and the instructor-defined evaluation script which observes the output of the student script and compares it with the correct solution, automatically providing a mark for the exercise. The main effort is probably given to the third script, the evaluator, and hence it is analyzed carefully here for a practical project in Control Engineering.

The network architecture of the system is depicted in Fig. 3. In that figure, the different actors are represented, comprising the web server for managing the database and the XHTML user interface; the evaluation server, which executes student code in an independent machine, and the user PCs. The use of a separate machine for executing the evaluation code is justified by the need of protecting sensitive data stored in the database from malicious student code. The user types are arranged into administrators, instructors and students. The latter can also be divided into “home users” for which there is no identity control and “school users” where the student ID and timing is controlled by a supervisor. In order to avoid fraud in the supervised work, we have also implemented an IP filter on the submission procedure that guarantees that the work submissions proceed only from the specified classroom.

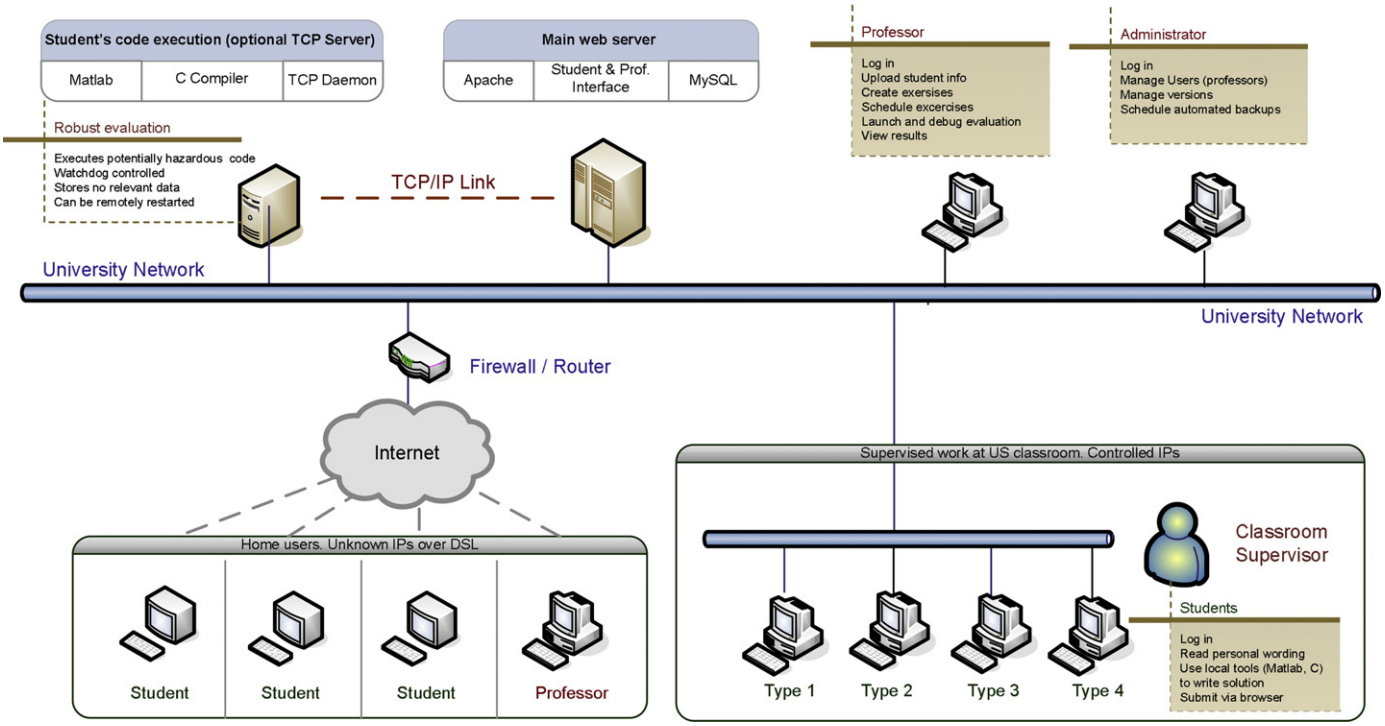

Fig. 3. System architecture over the University network.

3.1. Database structure

In order to understand the workflow for students and instructors, it is convenient to describe the structure of the application database, which is illustrated in the database diagram of Fig. 4. In it, two fundamental parts are distinguished, each of them coming from an independent feeding source: a) tables related to enroled students, list of courses, and organization of groups; and b) a table containing the data of the practical exercises (table `EXERCISES` in Fig. 4). The first set of tables stores lists of student names, passports and other enrollment information. This data is provided in general by the University administration. The student enrollment data is loaded into the database via an Excel file provided by the instructor with a simple column structure which, in the case of the University of Seville, is consistent with the course lists downloadable from the University Intranet.[1] To this basic information, student login data such as password and e-mail, is attached to the tables the first time a student logs into the system.

The entries of the second part of the database (table `EXERCISES`) are related to specific control exercises. They constitute a database in itself that can be copied or transferred from one server to another in order to favor code recycling. The data stored for each practical exercise is the following:

1. Brief description of the exercise in XHTML to be displayed in the student browser.
2. Name of a file uploaded to the server containing detailed exercise wording in PDF.
3. Matlab script defined by the instructor aimed at generating the input data necessary for the execution of the student code. This data can be a transfer function, a set of specifications, or any input argument to a calculation.
4. A Matlab optional text template to partially provide the structure of the solution. If this is used, the student is easily guided into a "filling blanks" task.
5. Matlab script to be executed after the student's code in order to check the validity of the provided solutions and compute the grade.
6. A set of flags that affect the general operation (interaction with the student and evaluation).

These two set of tables are fed independently, in the sense that new exercises can be introduced without creating a new group of students and vice–versa. However, the rationale of the system is based on linking these two databases by means of a so–called *assignment*. An association is an entry in that table that assigns a specific task to a group of students during a specific period of time. During that period, it stores the submissions in a new table (`SUBMISSIONS`) with consistent student-exercise data.

3.2. Student interface

The workflow for a student accessing the system is illustrated on the right part of Fig. 1. Initially, the student must login using their name and password. Then, the system identifies the groups to which the student belongs and displays the list of currently active tasks assigned to these groups. When the student selects one of these jobs, a new page is displayed containing:

1. The brief description of the exercise in XHTML.
2. A link to the full exercise wording in PDF.
3. A list of forbidden commands, words and operators that are not permitted in the exercise.
4. The name of the student in large capitals for impersonation control.
5. A multiline text field control for submitting the solution to the server. Optionally, this field may appear initially filled with the instructor-provided template.
6. A submit button.

The solution is provided by the student in plain text in the aforementioned field, by writing or editing the instructor template (if provided). The text submitted by the student must express the solution in Matlab syntax, though the student might be unaware of this. The syntactic rules to submit a correct solution are defined on a per–exercise basis; for instance, a PID tuning exercise could be responded using the following syntax: `Kp=10; Td=0.5; Ti=0.1;`. At evaluation time, the code provided by the student is executed in Matlab, leaving the three parameters in the workspace, so that the server can access them. Then, the required computations and simulations to check their validity are performed by the evaluation script.

3.3. Instructor interface

The workflow for an instructor accessing the system is illustrated on the left part of Fig. 1. After logging in, the interface consists in a menu with two types of options: a) Exercise definition and management; and b) Student group management. The items in the main menu are,

1. Create a new group. In order to introduce new students and courses into de database.
2. List of students in a course.
3. Create new exercises, defining all the parameters needed. The graphical interface for this step is partially shown in Fig. 5.
4. List and edit existing exercises.
5. Launch evaluation for a specific course or group. The evaluation procedure, once finely tuned, is executed off-line once all the submissions have been received. By this we make sure that any server failure due to malicious student code will not affect the collection of submissions.

[1] This format can be easily adapted to work in other education centres.

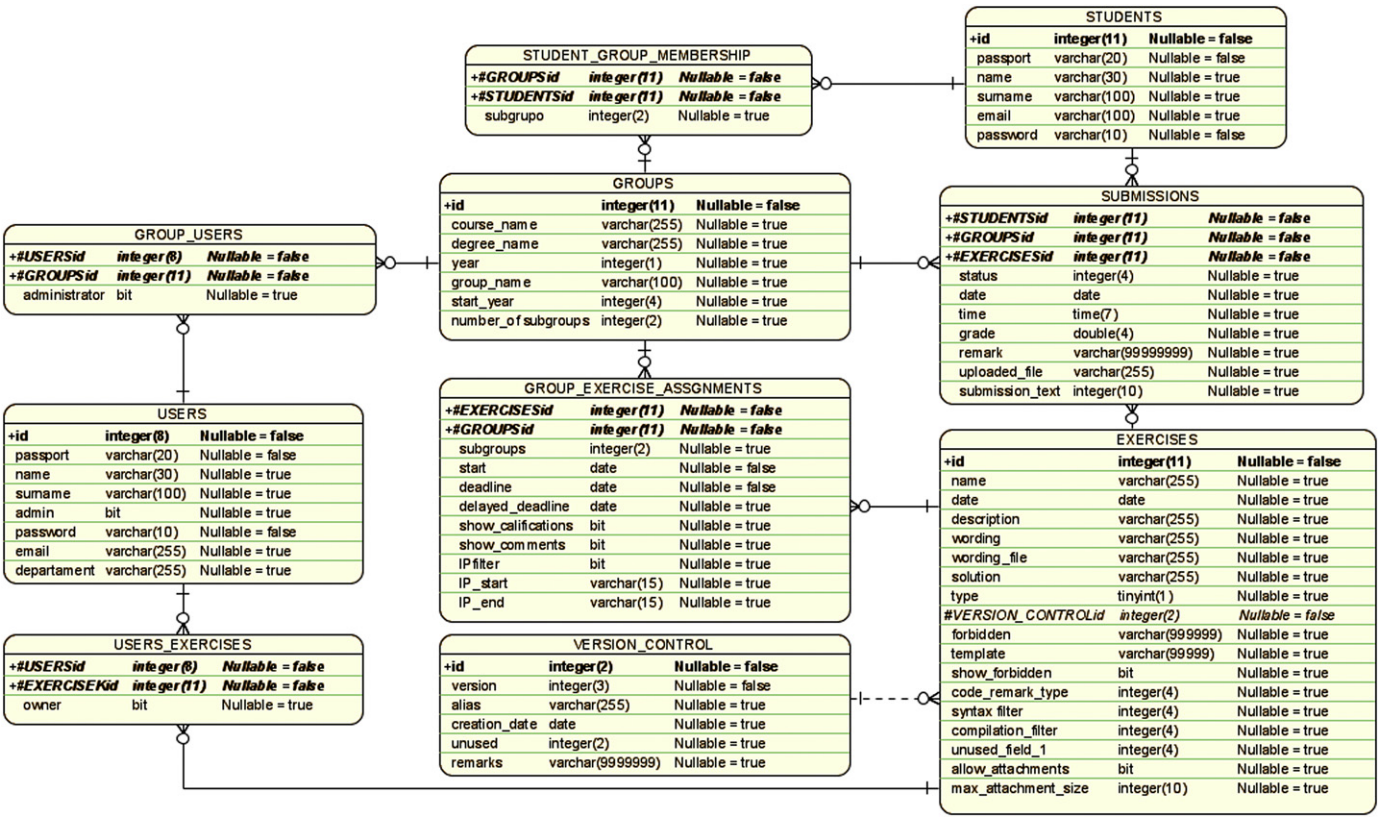

Fig. 4. Database structure.

6. View qualifications. The evaluation results may be viewed and downloaded in various formats such as XHTML and MS Excel. The evaluation results are stored in the database and has both numerical and alphanumerical data automatically generated by the evaluation algorithm.
7. Direct access to students submissions. This allows the instructor to study in depth the students solution, as well as manually correct minor syntax errors in order to debug the evaluation results.

It is important to remark that the evaluation procedure can be launched repeatedly for a specific group. This allows the instructor to iteratively fine tune the evaluation code by checking potential incoherences in the evaluation results. In practice, the evaluator is a complex

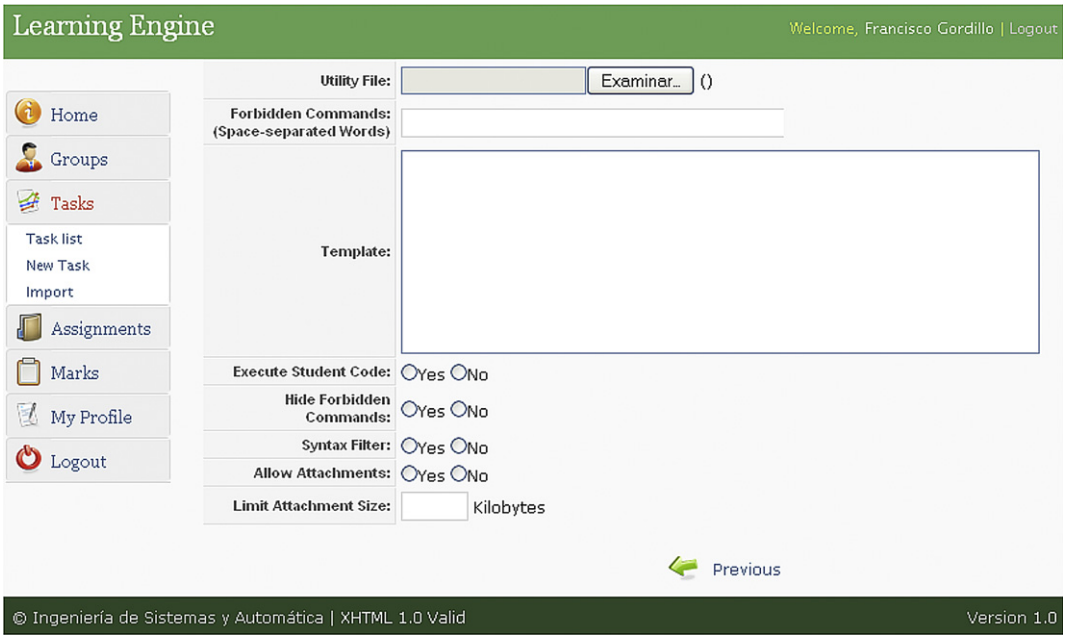

Fig. 5. Instructor interface for parameterization of exercises.

code that is debugged after a series of trial and error sequences. However, at this stage we have developed and debugged an important number of control exercises that can be directly recycled and put into work without any tuning at all.

In sections 5,6 and 7 three case studies of the application of the proposed automated evaluation scheme to different courses are presented.

4. Guidelines for developing an automatic evaluation script

In this section we present the basic structure of the evaluation code that an instructor must design in order to use the proposed application. Although in general each exercise is unique, most exercises belong to one of the evaluation paradigms presented in Section 2. We review next these cases.

4.1. Exact result matching

Fig. 6 shows the flow diagram of the evaluation code for the exact result matching paradigm. The steps of the algorithm are the following:

- Initialization code. In this step the algorithm accesses the database to retrieve all the information needed to evaluate an exercise such as the student's ID, the exercise parameters or the date of submission.
- Execute the student solution. In this paradigm, the solution provided by the students consists mainly of a set of variable definitions using the appropriate syntax. Note that the application must be able execute or include this lines of code in an appropriate manner depending on the language used.
- Check solution consistency. Although the proposed application implements a syntax filter that rejects solutions that are not correct syntactically, the solutions may still have errors. Robustness of the evaluation code with respect to these errors in the student solutions is important in order to provide a precise grade of the students work. If an error occurs, the evaluation code cannot provide a grade and the instructor must evaluate personally that student. Many errors are corrected by implementing a consistency check of the name, dimension and bounds of the different variables that the student solution should create. If a variable does not exist, there is a type mismatch or the answer is out of bounds it is replaced by an incorrect answer of the appropriate type. Robustness at this stage is further enhanced if the whole script is embedded in a `try-catch` exception handling structure, available in most modern programming languages.

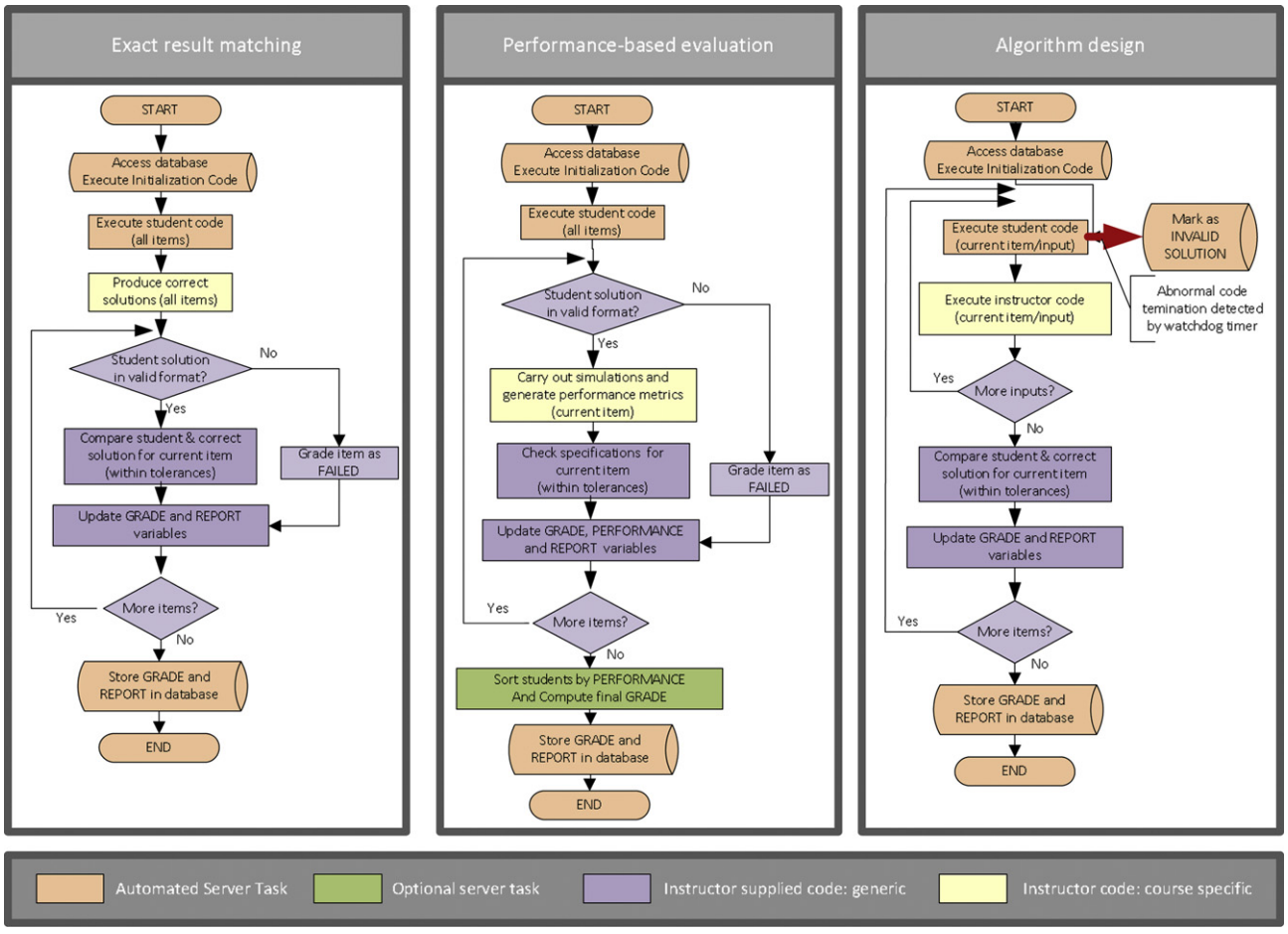

Fig. 6. Flowcharts of different types of evaluation paradigms.

- Generate correct solutions. Using the information retrieved from the database, the algorithm must generate the correct solutions to the exercise and the parameters corresponding to the particular student. To this end, the instructor leverages the possibility of carrying complex mathematical operations using Matlab, C or any other language implemented in the application. The solutions are stored in a different set of variables from the ones generated by the student.
- Evaluate. For each answer, the evaluation code compares the value of the variables created by the student and the correct value generated by the evaluation code. If the error is below a threshold, the grade is increased by a given value (different questions may have different weights). In some cases it is also important to generate a text comment to provide alphanumeric information for the student and the instructor.

Note that this code can be automatically generated (via templates) to some extent for a wide class of problems, in particular, the steps of checking consistency and comparing the solutions are independent of the exercise. The part of the algorithm which has to be designed "ad hoc" is the code which generates the correct solutions depending on the exercise and the parameters.

4.2. Performance-based evaluation

Fig. 6 shows the flow diagram of the evaluation code for the performance-based evaluation paradigm. The steps of the algorithm are similar to the previous case, but in this case, instead of generating correct solutions, the evaluation code must carry out a set of simulations to check the student design.

- Initialization code.
- Execute the student solution.
- Check solution consistency.
- Analysis of the student solution. In this step a set of simulations based on the solution of the student are carried out. The output of these simulations is then analyzed and a set of variables with several performance measures are generated.
- Evaluate. For each performance measure, the evaluation checks if the design specifications are satisfied and updates the grade and the comments appropriately. Note that it is possible to provide a grade on behalf of a single performance measure.

Part of this algorithm can be generated automatically for a wide class of problems, for instance exercises in which the grade depends on the number of specifications (i.e., performance measures satisfy given bounds) met by the student solution.

In general, the main difference between exact result matching and performance evaluation lies on the analysis of the student solution. How to design the inputs of the simulations and how to measure the different performance indexes depend on each particular exercise.

4.3. Algorithm design

In this evaluation paradigm the student must design an algorithm to solve a problem using a given programming language. This may lead to very different classes of exercises. We devote this section to exercises in which the algorithm must implement a function with specific input/output specifications. The application presented is able to include the solution provided by the student so that the evaluation code may call the functions developed by the students several times for both Matlab and C. To this end, the student code must satisfy several design requirements depending on the exercise for example C functions must implement a particular prototype. Fig. 6 shows the flow diagram of the evaluation code for the algorithm design paradigm.

- Initialization code.
- Include/compile the student solution.
- Include/compile the instructor solution.
- Analysis of the student solution. For each function, several inputs are applied to both the student and the instructor solutions. The resulting outputs are stored in a set of variables.
- Evaluate. For each answer, the evaluation code compares the value of the variables created by the student and the correct value generated by the evaluation code. This step is similar to the evaluation step in the exact result matching paradigm.

The proposed scheme is based on the idea that the students provide a solution which uses the sintaxis of a given programming language which implies that the resulting program may fail. The proposed application has been designed in order to be robust with respect to errors of the students code. If the evaluation program fails (for instance, hang-ups detected through timeouts), the server must be able to detect these errors and simply mark them in the database so that the professor knows that there is a programming error in the solution of that particular student. In addition, is important that these failures do not stop the rest of the evaluations. This is very important when a large number of students are being evaluated.

5. Case study: 2008 Systems Theory

The automated evaluation web server was first used during the Spring quarter of 2008 in the course Systems Theory, a compulsory course in second year of the Industrial Engineer degree at the University of Seville. The course was followed in 2008 by 366 students divided into four different groups. In order to pass the course the students have to do four different exercises based on carrying out a series of simulations using Matlab and Simulink. The exercises are different for each student because they depend on a personal ID number. These exercises take place in the computer labs of the School of Engineering. The students have to carry out the simulations and submit the results in a limited time. The results of these exercises are numerical, for instance the settling time or the overshoot of the response of a given system to a step input. This implies that these exercises are particularly appropriate for applying the proposed evaluation scheme.

5.1. Objectives

The main objective of this case study was to test the automated evaluation web server with a significant amount of students. Out of four regular practical sessions, we modified two in order to evaluate them using the proposed web-based tool, while the other two exercises were evaluated by the instructors of the course using traditional evaluation methods. The exercises were aimed to show fundamental results of the study of linear continuous time systems and familiarize the student with Matlab and Simulink. The four exercises deal with the following topics:

1. Modeling of LTI linear SiSo systems in Simulink (traditional evaluation method).
2. Time response of first and second order LTI systems (traditional evaluation method).
3. Frequency domain response of first and second order LTI systems (automated evaluation).
4. Linearization of a non-linear system around a given operation point (automated evaluation).

In what follows we present an example of one of the exercises that the students solved and that was evaluated with the proposed web-based tool.

5.2. Example of exercise: Frequency domain response of a first order LTI system

Consider the following linear system:

$$Y(s) = \frac{k}{1+\tau s}U(s)$$

where k and τ are defined as

- $k = (D1{+}1) \times 10$
- $\tau = D2 + 1$

with $D1$ and $D2$ the first and second digits of the student's ID number. Study using Simulink the response of the system subject to a set of sinodal inputs of amplitude 1 and different frequencies and fill Table 1 with the results.

The results have to be stored in a set of Matlab variables. In this case 7 different variables have to be generated (variables **p1** to **p7**), one for each row of the table. Each variable has four elements:

```
identifier = [frec amp tim_lag pha_lag];
```

Example of a solution submitted by a student:

```
p1 = [0.15 10 2 0.3];
p2 = [1.5 10 0.2 0.3];
p3 = [4.5 5.7 0.010.3];
p4 = [15 −5 0.01 0.3];
p5 = [45 −10 0.01 0.3];
p6 = [150 −30 0.010.3];
p7 = [1500 −50 0.001 0.3];
```

For this type of strongly structured solutions, the application allows the instructor to upload a code template that will appear on the student editable area, in such a way that the student will only have to substitute the dummy values of the template for actual numerical results, thus avoiding mistakes related to vector sizes, variable names, and syntax. In our sample exercise such a template looks as follows,

```
p1 = [0 0 0 0];
p2 = [0 0 0 0];...
```

Table 1
Results. Input frequency ω (rad/s); output amplitude Y_0 (dB); time lag Δ (s); phase lag φ (rad).

ω (rad/s)	Y_0 (dB)	Δ (s)	φ (rad)
p1 $0.01/\tau=$			
p2 $0.1/\tau=$			
p3 $0.3/\tau=$			
p4 $1/\tau=$			
p5 $3/\tau=$			
p6 $10/\tau=$			
p7 $100/\tau=$			

5.3. Results

The students used the automated evaluation web server in eight lectures (four groups and two exercises). In each lecture they solved the exercise using Matlab and Simulink, generated the results script and logged in to the server to submit the results. In general, there were no problems with the register/login procedure and the students collaborated during the whole quarter. The main difficulty found was that many of the solution scripts submitted did not comply with the format asked in the exercise description, mainly because the students were not familiar with Matlab's syntax. Of the 269 scripts received of the first exercise, around 30% had some kind of syntax error. This ratio was improved to 10% in the second exercise once the students familiarized with the system. This experience lead us to program a second version of the automated evaluation web server able to reject online submissions with syntax error using Matlab's `mlint` syntax checker. This second version has been used in the second case studied presented in the next section.

5.4. Evaluation

The first two exercises were manually corrected by the instructors using standard procedures (based on running a Matlab script for each student). This is a time consuming procedure, and in general each student is evaluated with four different marks, A, B, C or D based on the instructor criteria.

The third and fourth exercises were graded over 1 using the automated evaluation web server. In order to evaluate each of the exercises a different Matlab program has to be designed to carry out the following tasks (see Section 4):

- Generate the particular parameters of the exercise depending on the student's ID.
- Generate a set of variables with the correct answers using Matlab and Simulink.
- Compare the student's answers with the correct one and provide the resulting evaluation for a fixed criterion (for example percentage of correct answers with a given margin error).

Table 2 shows the frequencies of the resulting grades. What is most important of these results, is that using the automated web server allows us to distinguish with a much better precision the work done by each of the students. Moreover, once the exercise is programmed, the evaluation is done instantaneously, with a significant decrease in the time needed to grade the students of a large course, and to give them feedback.

The evaluation was repeated with a progressively refined evaluation script, until a reliable version was obtained. All the instructors that participated agreed that even if the automated evaluation web server cannot replace completely standard grading procedures (for instance, it may penalize minor typing errors that obscure a correct solution), it is a very powerful tool that helps the instructor to grade more precisely in courses with a significant number of students. In particular, the automated evaluation web server has been used to correct all the four exercises of Systems Theory discussed above during the 2009 spring quarter.

In addition to the opinion of the instructors involved in the case study, a survey was done to take into account the student's opinion. Over 200 surveys have been collected. Table 3 shows the questions and the mean value of the survey results (numerical value between 0 (negative opinion) and 5 (positive opinion)). As shown in the results Fig. 7, students generally rate in a positive manner the possibility of being evaluated more often and with a higher precision in order to have an estimate of their evolution along the course.

6. Case study: 2008 Automatic Control

In this section we present the results of the application of the proposed automated evaluation scheme to the Autumn quarter of 2008 in the course Automatic Control, a third year compulsory course the Industrial Engineer career of the University of Seville. The course was

Table 2
Evaluation results.

Grade	Exercise 3	Exercise 4
1	5	9
0.9	24	2
0.8	21	2
0.7	24	4
0.6	49	3
0.5	49	11
0.4	24	43
0.3	31	118
0.2	15	60
0.1	7	7

Table 3
Student's survey questions.

Question
Q1 Use of a web server as a mean to collect the results of an exercise
Q2 Quality of the instructions provided of each exercise
Q3 Ease of use of the automated evaluation web server
Q4 Possibility to hand in results out of lecture hours
Q5 Possibility of using the automated web server to carry out exams
Q6 Possibility of increasing the amount of exercises of the course

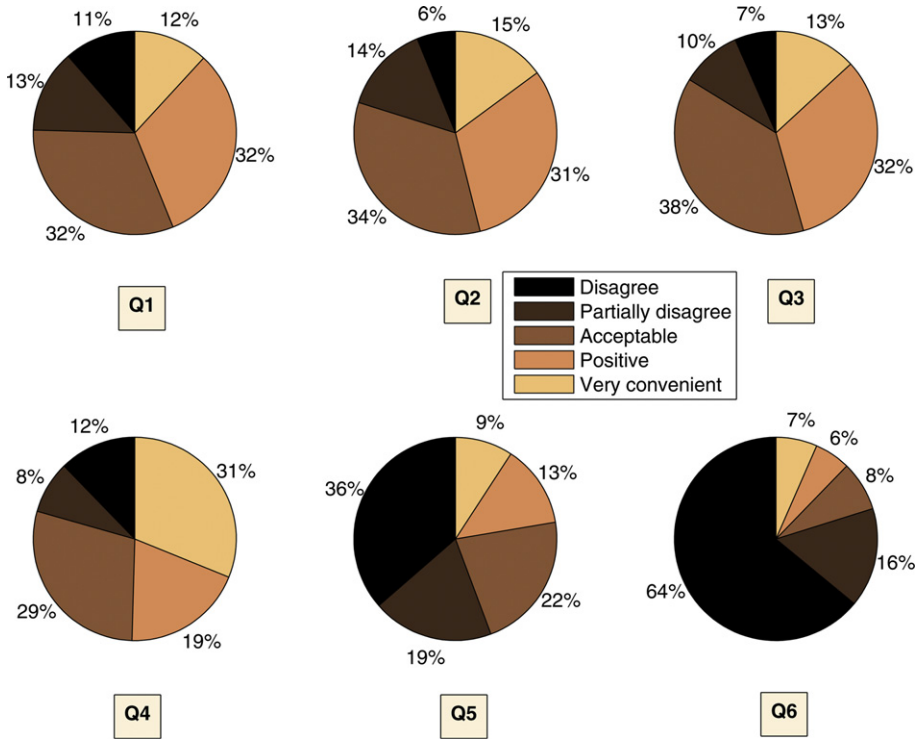

Fig. 7. Student's survey results.

followed in 2009 by 334 students divided into four different groups. In this course, the automated evaluation web server has been used to evaluate a final course project. The project consists in the design of several controllers for a non-linear system which parameters depend on a personal ID number of each student.

6.1. Project definition

The objective of the project is to put into practice the theory learned during the course. The students must obtain a Simulink model of a two conic tank system and design a set of controllers in order to satisfy a set of requirements. The model of the two tank system considered is the following:

$$
\begin{aligned}
q_v(t) &= K_{vs}\sqrt{P_v}R^{x(t)-1} \\
\tau\dot{q}(t) &= q_v(t) - q(t) \\
\dot{V}_1(t) &= q(t) - k_1\sqrt{H_1(t)} \\
\dot{V}_2(t) &= k_1\sqrt{H_1(t)} - k_2\sqrt{H_2(t)} \\
H_1(t) &= \sqrt[3]{V_1(t)/\alpha_1} \\
H_2(t) &= \sqrt[3]{V_2(t)/\alpha_2}
\end{aligned}
$$

where the parameters of the model depend on the ID number of each student. In particular, each student must convert its ID into a binary number. Table 4 shows the value of each parameter depending on the binary digits. It can be seen that 512 different systems are defined.

The objective is to regulate the level of the second tank around the operating point H_{2eq} (which also depends on the student's ID) by modifying the opening of the valve $x(t)$. The response of the closed-loop system to a step in the reference of amplitude 0.05 H_{2eq} from the operating point of must satisfy the following requirements:

Table 4
System's parameter definition.

Parameter	Units	0	1	⇐ Bit value
K_{vs}	$m^3\ min^{-1}\ bar^{-1/2}$	3	4	LSB
P_v	bar	1	3	
R		25	50	
τ	min	0.3	1	
α_1		0.5	0.7	
k_1	$m^3\ min^{-1}\ m^{-1/2}$	0.6	0.8	
α_2		0.5	0.7	
k_2	$m^3\ min^{-1}\ m^{-1/2}$	0.6	0.8	
H_{2eq}	m	1.5	3	MSB

1. Maximum steady-state error of 1 cm.
2. Maximum overshoot of 20%.
3. Maximum rising time of $(1/2)\tau$, where τ is the lower time constant of the linearized model of each tank.

To this end, the students must design a P, a PD, a PI and a PID controller using root-locus based techniques and a lead-compensator, a lag-compensator and a lead-lag compensator using loop-shaping techniques based on the linearized model of the non-linear system around the operating point.

6.2. Project evaluation

This exercise belongs to the performance-based evaluation paradigm. The solution of each student consisted of the eight tuned controller parameters. The evaluation code retrieved these parameters, carried out simulations with the non-linear system and provided a grade based on the number of specifications satisfied.

It is important to remark that the design of the evaluation code for a project of this class requires in general a considerable amount of work. In this particular project, each evaluation involved several simulations which were solved using a fixed-step Euler ODE solver. The evaluation code was made of five Matlab scripts (over 500 lines). Although the solutions provided by the students were syntactically correct (thanks to the `mlint`-based syntax filter), several errors could occur (for instance, when the students did not provide all the answers). Several iterations were needed to obtain a robust evaluation code.

However, once the code is designed, precise grading of the work of all the students of the course is possible. In this particular case, because of the number of students, a project of this class was never proposed before because of the time needed to grade it. Note that because of the personalized nature of the system, each project would have to be graded and simulated independently. All the instructors of the course agreed however on the relevance of offering the students the possibility to apply in a case study using Matlab and Simulink all the theoretical issues reviewed in the course. Using the web-based tool allows the instructors to implement this class of project-based learning methodologies.

In order to assess whether the used of the new tool has improved the overall performance of our students, we have checked historical data from final exams (which are still done in the traditional written form). The results are summarized in Table 5, and shows a significant improvement the last year (when the application has been used) in terms of successful students with respect to past editions of the course.

7. Case study: 2010 Computer Science and Programming

In this section we present the results of the application of the proposed automated evaluation scheme Computer Science and Programming, a first year compulsory course the Industrial Engineer career of the University of Seville. The course was followed in 2010 by 412 students divided into four different groups. In this course, the automated evaluation web server has been used to evaluate eight different assignments. Each assignment consisted on the definition of three functions for which the prototype was given. The assignments were solved by the students in the computer classroom with limited time. The grade of all the assignments accounted for 15% of the course grade.

7.1. Example of exercise: Leibniz formula

The value of π has been approximated dy different methods. One method is to use Leibniz's formula:

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = +\frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \ldots = \frac{\pi}{4}$$

Design a function that provides an approximation of π obtained by adding a fixed number of elements of Leibniz's formula. The prototype is:

```
double leibniz(int N);.
```

Example of the evaluation code:

```
#include <stdio.h>
#include <math.h>
double leibniz(int N);
double Pleibniz(int N);
```

Table 5
Academic results in Automatic Control Course (Industrial Engineering) over period 2006–2009 (% of students in each grade interval).

Grade	2006–07	2007–08	2008–09
>95%	0.8%	0.0%	4.0%
85%–95%	1.7%	0.0%	2.0%
70%–85%	10.0%	21.1%	40.5%
50%–70%	45.8%	56.6%	45.9%
<50%	41.7%	22.4%	7.4%

Table 6
Summary of past experiences over period 2007–2011.

Course	Degree	Period	Students
Comp. Science and Prog.	Industrial Eng.	2008–2011	1860
Comp. Science and Prog.	Aeroespatial Eng.	2008–2011	547
Comp. Science and Prog.	Civil Eng.	2010–2011	77
Comp. Science and Prog.	Chemical Eng.	2008–2011	80
Machine Theory	Industrial Eng.	2010–2011	611
System Theory	Industrial Eng.	2007–2011	1432
Automatic Control	Industrial Eng.	2008–2011	1091
Automatic Control	Telecomm. Eng.	2009–2011	387
Automatic Control	Tech. Industrial Eng.	2008–2011	400
Optimal Control	Electronic Eng.	2008–2011	52
Distributed Control	Computer Science	2009–2011	47

```
void main(void)
{
   float grade=1;
   int N;
   for(N=1;N<100;N++)
      if((fabs(leibniz(N)−Pleibniz(N))>0.05*Pleibniz(N)))
      grade = 0;
   printf("%.1f",grade);
}
double Pleibniz(int N)
{
   double res,aux;
   int n;
   res = 0;
   aux = −1;
   for(n=0;n<N;n++)
   {
      aux = −aux;
      res += aux/(2*n+1);
   }
   return 4*res;
}
```

The function `Pleibniz` is a function designed by the instructor which solves the proposed problem. The evaluation code must write in the standard output the value of the grade, which is one if the problem is correct and zero if not.

In 2010 each student had to face 27 different C programming short problems. There were over 400 students. If the 5 professors had to manually check whether those over 10,000 programs compiled and provided the correct solution, each would have to compile around 2000 programs. The development of the evaluation codes took also a lot of time, but of course just a fraction of the manual evaluation. Moreover, the exercises will be used for other courses/years.

8. Summary of past experiences

In order to support the claims of this paper, besides the two case studies, a summary of past experiences carried out since 2007 with our tool is presented in Table 6. In this table, a list of all the courses offered by the University of Seville which used the tool presented is shown. The entry “Degree” shows in which degree the course is taught while the entries “Period” and “Students” show the period of time in which the tool has been used and the total number of students which have followed each course during this time.

By observing the empirical data, it is easy to conclude that the tool has reached maturity and that it has gained a significant place in our education institution (technical college with about 7000 students in most fields of engineering), while it is spreading elsewhere. Nowadays, the database holds over 4377 students organized in over 33 different courses and more than 20,157 submitted (and evaluated) results.

9. Conclusions

In this paper we have presented a novel web based application which can be used to collect and evaluate student’s submissions of control course projects and exercises. The application was first applied with students of the Engineering School of the University of Seville, where it is being used on a weekly basis in numerous courses. The automatic evaluation web server allows the instructor to collect and automatically correct students submissions using a Matlab script, which provides a general framework in which different classes of exercises can be easily defined. The tool has been tuned and refined for over four years in order meet numerous challenges that have appeared in the different control and programming courses where it has been applied and is particularly appropriate for courses with a high number of students.

10. How to test the application

Should the reader be interested in testing and using the application by freely creating auto-evaluated exercises and assigning them to their students, it is possible to do so on a fully functional application running on a public server. Please contact the authors by e-mail to gain password access.

Acknowledgment

Financial support from the University of Seville, under the programme *Plan de Renovación de las Metodologías Docentes*, 2007–2010, is gratefully acknowledged.

References

Barla, M., Bieliková, M., Ezzeddinne, A. B., Kramár, T., Simko, M., & Vozár, O. (2010). On the impact of adaptive test question selection for learning efficiency. *Computers and Education, 55*, 846–857.

Casini, M., Prattichizzo, D., & Vicino, A. (2003). The automatic control telelab: a user-friendly interface for distance learning. *IEEE Transactions on Education, 46*, 252–257.

Casini, M., Prattichizzo, D., & Vicino, A. (2004). The automatic control telelab–a web-based technology for distance learning. *IEEE Control Systems Magazine, 24*, 36–44.

Cedazo, R., Lopez, D., Sanchez, F. M., & Sebastian, J. M. (2007). Ciclope: foss for developing and managing educational web laboratories. *IEEE Transactions on Education, 50*(4), 352–359, URL. http://dx.doi.org/10.1109/TE.2007.907268.

Chiou, C., Hwang, G., & Tseng, J. C. R. (2009). An auto-scoring mechanism for evaluating problem-solving ability in a web-based learning environment. *Computers and Education, 53*, 261–272.

Douce, C., Livingstone, D., & Orwell, J. (2005). Automatic test-based assessment of programming: a review. *Journal on Educational Resources in Computing, 5*, 1–13.

Garcia, M. I., Rodriguez, S., Perez, A., & Garcia, A. (2009). p88110: a graphical simulator for computer architecture and organization courses. *IEEE Transactions on Education, 52*(2), 248–256, URL. http://dx.doi.org/10.1109/TE.2008.927690.

Gentil, S., & Exel, M. (2004). Is it feasible to teach control with MCQs?. In *Proceedings of 2nd IFAC Conference on IBCE*.

Gerosa, M., & Narayanan, S. (2008). Investigating automatic assessment of reading comprehension in young children. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2008* (pp. 5057–5060).

Gutiérrez, E., Trenas, M. A., Ramos, J., Corbera, F., & Romero, S. (2010). A new moodle module supporting automatic verification of vhdl-based assignments. *Computers and Education, 54*, 562–577.

Heck, B. S., Poindexter, S. E., & García, R. (2000). Integrating the web into traditional teaching methods. In. *Proceedings of the American Control Conference, Vol. 5* (pp. 3444–3448).

Jaaskelainen, A., Katara, M., Kervinen, A., Maunumaa, M., Paakkonen, T., Takala, T., et al. (2009). Automatic gui test generation for smartphone applications–an evaluation. In. *1st International Conference on Software Engineering - Companion Volume, 2009*, (pp. 112–122). ICSE-Companion.

Jiang, H., Zhang, L., & Ye, W. (2009). The automatic evaluation strategies and methods of multimedia work assignments. In *International Conference on Computational Intelligence and Software Engineering, CiSE 2009* (pp. 1–5).

Kapur, S., & Stillman, G. (1997). Teaching and learning using the world wide web: a case study. *Innovations in Education and Training International, 34*, 316–322.

Klai, S., Kolokolnikov, T., & van den Bergh, N. (2000). Using maple and the web to grade mathematics tests. In *International Workshop on Advanced Learning Technologies (IWALT 2000)* (pp. 89–92).

Maussion, P., & Tricot, A. (2008). A computer aided learning module for digital control: description, analysis, test and improvement. In *34th IEEE Annual Conference on Industrial Electronics, IECON 2008* (pp. 3467–3472).

Mavrikis, M., & Maciocia, A. (2003). WaLLiS: a web-based ILE for science and engineering students studying mathematics. In *Workshop of Advanced Technologies for Mathematics Education in 11th International Conference on Artificial Intelligence in Education*, Sydney Australia. URL. http://www.lkl.ac.uk/manolis/pubs/pdfs/aied-wallis-03.pdf.

Petridis, V., Kazarlis, S., & Kaburlasos, V. G. (2003). Aces: an interactive software platform for self-instruction and self-evaluation in automatic control systems. *IEEE Transactions on Education, 46*(1), 102–110.

Poindexter, S. E., & Heck, B. S. (1999). Using the web in your courses: what can you do? What should you do? *IEEE Control Systems Magazine, 19*, 83–92.

Rodriguez, S., Pedraza, J. L., Dopico, A. G., Rosales, F., & Mendez, R. (2007). Computer-based management environment for an assembly language programming laboratory. *Computer Applications in Engineering Education, 15*(1), 41–54, URL. http://dx.doi.org/10.1002/cae.20094.

Rodriguez, S., Pedraza, J. L., Garcia, A., Rosales, F., & Mendez, R. (2007). Computer-assisted assembly language programming laboratory. *International Journal of Electrical Engineering Education, 44*(3), 216–229.

Rodriguez, S., Zamorano, J., Rosales, F., Dopico, A. G., & Pedraza, J. L. (2007). A framework for lab work management in mass courses. application to low level input/output without hardware. *Computers and Education, 48*(2), 153–170, URL. http://dx.doi.org/10.1016/j.compedu.2004.12.003.

Sanchez, J., Dormido, S., Pastor, R., & Morilla, F. (2004). A java/matlab-based environment for remote control system laboratories: illustrated with an inverted pendulum. *IEEE Transactions on Education, 47*, 321–329.

Sanchez, J., Morilla, F., Dormido, S., Aranda, J., & Ruiperez, P. (2006). Virtual and remote control labs using java: a qualitative approach. *IEEE Control Systems Magazine, 22*, 8–20.

Sangwin, C. J. (2004). Assessing mathematics automatically using computer algebra and the internet. *Teaching Mathematics and Its Applications, 23*, 1–14.

Tartaglia, A., & Tresso, E. (2002). An automatic evaluation system for technical education at the university level. *IEEE Transactions on Education, 44*(3), 268–275.

Torres, F., Candelas, F. A., Puente, S. T., Pomares, J., Gil, P., & Ortiz, F. G. (2006). Experiences with virtual environment and remote laboratory for teaching and learning robotics at the University of Alicante. *International Journal of Engineering Education, 22*, 766–776.

Wang, T. (2010). Web-based dynamic assessment: taking assessment as teaching and learning strategy for improving students e-learning effectiveness. *Computers and Education, 54*, 1157–1166.