AUTOMATIC GENERATION AND CORRECTION OF TECHNICAL EXERCISES

Ferran Prados[1], *Imma Boada*[2], *Josep Soler*[3], *Jordi Poch*[4]

Abstract – *The kernel of an e-learning environment developed to improve both teaching and learning at the technical/engineering degrees at Girona Polytechnic University is presented. Such a kernel consist of two modules. The exercise generation module, used to automatically generate different versions of a base exercise and the correction module, used to automatically correct the generated exercises by applying a solution code maintained in the base problem. The key of both modules is on the definition of the base exercise which varies according to the subject. Our environment supports mathematics, physics and computer science problems amongst others.*

Index Terms - Engineering and technology education. Automatic correction, exercise generation

INTRODUCTION

Over the last few years advances in technology and communications have lead to transformations in the education system. Not surprising e-learning has become a priority at the present time in higher education [1][2]. Currently, most higher education institutions have introduced web-based environments into their curricula.

In this paper we present the kernel of an e-learning environment developed to improve both teaching and learning at the technical/engineering degrees at Girona Polytechnic University. Concerned by the alarming failure rate of these courses we decided to develop an e-learning platform with the idea of reducing it. We identified two major reasons for the problem. The first is due to the fact that in most subjects concepts are built one upon the other and a lack of understanding in the first topics makes difficult the comprehension of the next. After the few first sessions an important percentage of students lose the thread of the course. The second reason is a direct consequence of the first. In general, engineering courses involve students having to solve problems by applying the theoretic concepts. If students have not acquired the theoretic level they do not feel motivated to do the problem solving component.

The evaluation of this situation pointed to continuous assessment as the solution to reduce the failure rate. A student should not be able to advance in the subject unless a specified level has been reached and the ability to solve related problems has been acquired. Since this ability is related with student skills, our interest is focused not only on the evaluation of knowledge but also on the skills to solve problems. In general, e-learning platforms are devoted to multiple choice, fill-in-the-blanks, etc. They are suitable for the assessment of theoretic concepts, but, for our purposes, we need an additional capability in order to support practical problems.

With this idea in mind we developed a new e-learning framework aimed at:

(i) *Supporting continuous assessment.* The main task involved in carrying out continuous assessment is teacher correction which in many cases is just a mechanical process. The idea is to develop a module able to perform this mechanical process.

(ii) *Supporting different kind of problems and not only test problems.* Our framework has to have a multidisciplinary domain that enables it to be adapted to different subjects. Although subjects might look quite different they can in fact share a lot of common features. The correction of physics and mathematical problems, for example, often involves the same steps.

(iii) *Providing teachers with feedback of student weaknesses.* We are not just interested in the correction of student work we also want to track their progress on the subject and the acquired skills.

(iv) *Providing students with a friendly scenario to solve practical problems.* Design an environment that makes student feel comfortable and supported. Each student must have personalized attention, this means continuous feedback from the platform and also exercises specifically designed for him. The environment should be seen as a private teacher who offers help when the student is doing practice.

Currently, our e-learning platform is used in several subjects of the technical/engineering degrees of the Girona Polytechnic University with very promising results. In [6] , [7] and [8] we described how it is applied in mathematic, programming and database courses, respectively. The purpose of this paper is to describe the kernel of the platform and how it has been designed in order to support different kind of problems.

THE KERNEL OF OUR PLATFORM

To define the platform the challenge was to develop a module able to automatically support different kind of exercises. Two main considerations had to be taken into account. First, to guarantee that each student has a set of

[1] University of Girona, Computer Science Department, Campus de Montilivi, 17071 Girona, Spain, ferran.prados@udg.es
[2] University of Girona, Computer Science Department, Campus de Montilivi, 17071 Girona, Spain, imma.boada@udg.es
[3] University of Girona, Computer Science Department, Campus de Montilivi, 17071 Girona, Spain, josep.soler@udg.es
[4] University of Girona, Computer Science Department, Campus de Montilivi, 17071, Girona, Spain, jordi.poch@udg.es

exercises specifically designed for him we need a method to automatically generate a great diversity of exercises. Second, to correct the exercises we need an strategy that avoids entering an specific solution for each exercise. To satisfy both requirements we introduce the concept of base exercises and we design the exercise generation and correction modules described below.

I. Base Exercise

A base exercise is a single representation of a set of problems. It consists of three main components:

- *The problem descriptor* which includes the different descriptors of a problem. Each descriptor always contains one or more variable parameters which can be numbers, functions or words.
- *The problem parameters* which maintain for each parameter of the descriptor the list of possible values that could take.
- *The solution code* which maintains the strategy or the solving method that has to be applied to correct all the problems of the represented set.

The definition of this base exercise depends on the subject as it will be seen in next sections.

II. Exercise Generation Module

In an e-learning environment generating different exercises to guarantee that each student has a different workbook can be a daunting task for any teacher. Moreover, if we take into account that there is a large number of students per class. The purpose of the exercise generation module is to reduce this demanding task by a process that automatically comes up with different versions of a base exercise.

A problem can be seen as a description, i.e. the set of sentences describing the situation, and a set of parameters, i.e. the values that have to be taken into account to solve it. In order to obtain different versions of the same problem, we consider: the information of the problem descriptor, the problem parameters components of the base exercise and the different degrees of variability that can appear in these parts. Given a base exercise with this set of variable parameters the generation module applies a random process that generates as many different versions of the base exercise as the combination of the parameters allows. These exercises are recorded in the repository of the system and assigned to the students.

III. Exercise Correction Module

Since we have different problems generated from a base exercise, we require a method which automatically corrects all of them. To avoid entering individual solutions for each one of the generated exercises, the exercise correction module adjusts the solution code of the base exercise to the exercise to be corrected by taking into account its variable parameters. The adjustment varies with the kind of exercise.

Since the key of all our strategy is on how we define the base exercise, we are going to consider the typology of exercises that are currently supported by our e-learning platform and how the base exercise is constructed in each case. The main groups of problems are: problems solved by computer algebra systems, computer science problems and test problems.

PROBLEMS SOLVED BY COMPUTER ALGEBRA SYSTEMS

The first group of exercises we consider are problems which must be solvable with a computer algebra system. In this group a large variety of problems from different subjects such as mathematics, physics, chemistry, etc. can be included.

An example to give an idea of what a base problem looks like is given below. To improve the example comprehension a simplified notation has been used. In this example the descriptor item stores three different versions of the same problem. Observe that in all of these P1 and P2 appear as the variable parameters of the problem. The values that these parameters can take are represented in the problem parameters item.

```
Problem Descriptor:

- Find the volume of a solid whose sections along
planes perpendicular to the x-axis are P1 if the
base of each section lies between the curves P2
in the xy-plane

- The base of a solid is the region enclosed by
the curves P2 in the xy-plane and the vertical
sections parallel to the line x=0 are P1. Find
the volume of a solid

- A solid is considered whose base is the region
limited by the curves P2 in the xy-plane and
whose sections along vertical planes parallel to
the line x=0 are P1. Find the volume of the
solid.

Problem Parameters:

P1={equilateral triangles; half–circles; regular
hexagonal; rectangles; isosceles triangles}
P2={y=x^2 and 2y=x^2+1;y=3+3x-x^2 and y=3*x^2-
6*x-9;y=x^2-1 and y= 5+4x-x^2 }
```

Since these exercises must be solvable with a computer algebra system, we force the exercises generated by our system to be implementable in Mathematica code. Such a restriction rules out exercises in which a mathematical proof is asked for, but these are not the main objective of an intermediate-level mathematics course in engineering studies.

Taking this restriction into account, in the solution code of the base exercise we maintain the Mathematica code needed to solve all the exercises generated from it. Following with the previous example, below we describe the corresponding solution code. Note that P1 and P2 parameters will be replaced by the corresponding P1 and P2 values of the generated exercise. In this example `SO` represents the student's solution.

Solution Code:

```
Clear{so1,p1,p2,f,g}
Sol=SO;
Accuracy = 6;
p1=P1;
p2 ={P2};
f[x_]=p2[[1]];
g[x_]=p2[[2]];
s=Solve[{y==f[x],y==g[x]},{x,y}];
s=x/.s;
x1=Min[s];
x2=Max[s];
Integrate[p1*{f[x]-g[x]}^2,{x,x1,x2}];
v=N[%];
If[SetAccuracy[IntegerPart[sol]-
                IntegerPart[v],accuracy]!=0,
Print["NotCorrect"]; Quit[]];
If[SetAccuracy[FractionalPart[sol]-
              FractionalPart[v],accuracy]!=0,
Print["Not Correct],
Print["Correct"],
Print["There is probably a syntax error"]];
```

When the student has the solution of the problem that has had assigned, he enters it into the system by using an specific interface designed for it [6]. The correction module adjusts the Mathematica code of the base exercise with the values of the problem. Then, it executes on-line the Mathematica code and answers back whether the solution is correct or not. In case of error the student can send a new solution.

COMPUTER SCIENCE EXERCISES

Currently, the system also supports computer science exercises of introductory programming and database courses. In these exercises the degree of variability that can be entered is minimal compared to that previously presented. In fact the main point of interest of these exercises is in its automatic correction. To present them we describe first programming exercises and then database exercises

I. Programming Exercises

Programming exercises have been designed for introductory programming courses. These courses consist of lectures, where key concepts of programming are explained, and laboratory sessions where students practice. To make programming less difficult for students we use a pseudo-code in the lectures and a programming language in the laboratory. The programming language is the variable parameter of a programming base exercise.

Next, we present a simple example of a programming base problem just to see how it looks like.

Problem Descriptor:

```
Design a program using P1 to compute the area of
a rectangle. Consider integer data.
```

Problem Parameters:

```
P1={Pseudocode, Java, C++, Pascal}
```

To solve a programming exercise the student must write the solution code using the language fixed by the exercise. Such a solution is written in a text file and sent to the correction module. This module calls the appropriate compiler and afterwards shows the compiling errors, if there are any. In the case that there are errors, they can be corrected and a new solution can be sent. This process can be repeated until a solution with no compiling errors is obtained.

To support pseudo-code solutions we have developed a pseudo-code compiler which generates JVM (Java Virtual Machine) code. This compiler provides all the facilities to identify syntax errors easily. When no compiling errors appear in the solution, the correction module validates the student code by using the set of testing values of the base exercise. These values simulate the results obtained with the correct solution of the problem for a given input, i.e. a set of inputs with the corresponding correct output. During this process the system shows the correct solutions and the solutions obtained with the student code. In the case that there are errors a new solution can be sent. The test data used to correct the student code is selected in a random process. Therefore if the student sends different solutions the testing sets used in each correction have not to be the same.

Following the example the test data entered for the problem are the input, two integers representing the basis and the height of the rectangle, and the area as the output.

Test Data:

```
input= { 0,0 } output = {0}
input= { 2,4 } output = {8}
input= { 4,5 } output = {20}
input= { 3,3 } output = {9}
input= { 5,9 } output = {45}
```

II. Database Exercises

In undergraduate database courses students are taken through all stages of database development: analysis, conceptual, logical and physical design. At the end of the course the student, amongst others things, has to be able to design relational database schemas. In the case of database problems our interest is not on the automatic generation of different versions of the problem but in the automatic correction. We consider that the degree of variability is inherent to the problem descriptor and hence the automatic generation of new versions is not necessary.

To describe these exercises we consider an example of a well-known database exercise from [5]. In this case *The problem descriptor* presents a set of requirements for a real life application from which the student has to design a relational database schema, i.e. a set of tables with the corresponding attributes including primary and foreign keys. In the descriptor, we fix the names of all attributes, so that the correction will be easier. Observe in the example that the name of the attributes which have to be used to solve the problem are written in bold..

```
Problem Descriptor:

Design the relational database schema for a
company taking into account that:

- The company is organized into departments. Each
department has a unique name (dname), a unique
number (dnumber), and a particular employee who
manages the department (mgrssn)
- A department controls a number of projects,
each of which has a unique name (pname), a unique
number(pnumber)and a single location (plocation).
- We store each employee's name (fname)(lname),
social security number (ssn), address( address),
salary (salary), sex (sex) and birth date(bdate).
An employee is assigned to one department but may
work on several projects, which are not
necessarily controlled by the same department. We
keep track of the number of hours (hours) per
week that an employee works on each project.
```

The *solution code* is the set of tables with the corresponding attributes, including primary and foreign keys, that satisfies all the requirements of the problem. The solution of a database exercise is not unique. For this reason in the base exercise we maintain some of these solutions. In the example it can be seen that each one of them consists of a set of tables with the corresponding attributes, primary and foreign keys.

```
Solution 1:

EMPLOYEE
Primary key: ssn
Foreign key: dnumber
Others attributes: fname, lname, bdate, address,
salary, sex

DEPARTMENT
Primary key: dnumber
Foreign keys: mgrssn
Others attributes: dname

WORKS_ON
Primary key: ssn, pnumber
Foreign keys: ssn, pnumber
Others attributes: hours

PROJECT:
Primary key: pnumber
Foreign keys: dnumber
Others attributes: pname, plocation
```

To solve the database exercise the student enters by using an interface specifically designed for it the set of tables with the corresponding attributes with the primary and foreign keys. He must use the attributes given in the problem descriptor. The correction module compares the solutions maintained in the base problem with the student solution. It checks if the student has organized attributes in the same way and if the same primary and foreign keys have been defined. The result of the correction is given on-line

TEST EXERCISES

The last group of exercises we are going to present are test exercises. We consider three different test modalities which differ in the type of questions that they include. These are: (i) *True-false questions*. There is a single question and the correct answer is yes, no or unknown. (ii) *Multiple choice questions with only one correct answer*. There is a single question with a set of possible answers and only one of these answers is the correct one. (iii) *Multiple choice with more than one correct solution*. There is a single question with a set of possible answers and more than one of these answers are correct.

In general, an e-learning platform that supports test exercises has a pool where all test questions have been entered [3][4]. To generate a test exercise a random process selects questions from this pool. The random process can be applied to select the questions and, in the case of multiple choice with more than one solution, it can also be applied to select the set of possible answers that appear in the question descriptor. Moreover, if questions are organized by topic the random process should also randomize the number of questions per topic. However, independent of the degree of sophistication of the random process and the number of parameters that can be randomized, in the end the variety of tests that can be generated in these platforms depends solely on the number of questions stored in the pool.

Our proposal introduces a variability factor in the question descriptor. A test base exercise is composed of four items:

1. *The question descriptor*. Here we include the different descriptors of the question with the corresponding variable parameters. We can consider whole questions or questions composed by different parts. In this last case, we decompose the question into different parts: introduction, body and final question. For each one of these parts we introduce different descriptors. For example, an introductory part should be *Consider that … / Suppose that … /Let us consider that…* i.e. different sentences to say the same thing. The generation module applies the random process to each one of these parts and the final version is obtained by connecting them.
2. *The problem parameters* which maintains the possible values that could be taken by the parameter.
3. *The solution code* maintains the different answers, which can also have variable parameters. Each item has assigned the solution code (i.e. if it is true or not, and in case of depending of a function how it has to be computed). If the answer depends of a parameter the solution code is given as a function of this parameter.
4. *Number of visible answers*. According to the type of test we have selected (true-false or multiple choice) we can select the number of answers of each question of the test.

Next, we give an example of a base test problem. In this case there is only one descriptor of the question and the question has one variable parameter P1. As it can be seen in next item P1 represents two matrices A, B which can take three possible values. In the answer descriptor we write a possible answer and its corresponding solution code. The first one for example is `A·B is a 3x4 matrix`, in this case the answer will be correct if A and B are substituted by the second set of values that can be taken by the parameter P1. Finally the number of visible answers has been set to 4, this means that each generated test in which this question appears will have four possible answers.

```
Problem Descriptor:

Given the following matrices P1 which of the
following sentences are true

Problem Parameters:

P1={
A=((1 2 3)(4 7 6)) B=((2 0)(-1 4)(2 7)(1 5));
A=((7 6)(2 4)(1 -1)) B=((5 6 2 1)(0 -1 4 3));
A=((4 7 -1 0)(2 6 3 8)(0 1 0 1))
B=((2 1 0)(1 -3 -1)(0 -1 0))
}

Answer descriptor and Solution Code:

- A·B is a 3x4 matrix
Correct when P1 is the second
- A·A is an squared matrix
Incorrect
- B^t·B =B·B^t
Correct when P1 is the third
- A^t·A is a 3x3 squared matrix
Correct when P1 is the first
- A·B =B·A
Incorrect
- A·B is an squared matrix
Incorrect

Number of visible answers: 4
```

An important feature of our test generation module is that it supports question with images. This is of special interest in exercises that require the evaluation of a graph. We can enter the variable parameters into the graph function in such a way that each plot will be different.

The correction of test exercises is straightforward. The correction module compares the solution entered by the student with the correct solution maintained in the base problem. As in the previous case there is an interface designed to enter the solutions of test exercises. The test correction module integrates a functionality, supervised by the teacher, that allows students to view their response and the correct response to each question on the test. This is a good way for students to learn from their mistakes. Even though students would give the correct test answer to another student, we can be confident that the student's test question and answers are unique from other students.

OUR E-LEARNING PLATFORM

To end the paper we give a brief overview of our e-learning platform to see how these modules are integrated in it. Besides the generation and correction modules our e-learning platform has: (i) A repository of problems which maintains the base problems introduced in the system. (ii) An assessment module which collects from the data base quantitative data about student work such as number of errors, types of errors, time taken to complete the problem, etc. (iii) A help module which supports help exercises. The repository of problems stores exercises and exercises with help levels that guide the student to the solution. The help module according to the mistakes made by student activates the different levels of help. (iv) A Student-Teacher Communication Channel which establishes a virtual communication channel between the teacher of a group and all the students that compose this group. This channel can be used by the teacher as a virtual tutoring system.

A demonstration of our e-learning platform is available at *http://acme.udg.es/demoicece2005*

CONCLUSIONS

We have presented the kernel of an e-learning platform developed at the Girona Polytechnic University and used in the technical/engineering degrees. Different of other platforms it supports different types of problems by using a similar strategy based on a base exercise and a automatic correction and an exercise generation module. The capability of our platform to support a great variety of exercises makes it suitable not only for knowledge assessment but also for skills. Currently, the platform is used in several courses with very promising results.

REFERENCES

[1] Barajas M., “Restructuring Higher Education institutions in Europe. The case of virtual learning environments”, *Interactive Educational Multimedia*, No.5, October 2002, 1-28.

[2] Colace F., De Santo M, Vento M., “Evaluating On-Line Learning Platforms: A case Study” *36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 5* January 06 - 09, 2003

[3] Hwang G.J., "A Test-Sheet-Generating Algorithm for Multiple Assessment Requirements", *IEEE Transactions on Education,* Vol. 46, No.3, August 2003, 329-337.

[4] Tartaglia A., Tresso E.., "An Automatic Evaluation System for Technical Education at the University Level", *IEEE Transactions on Education,* Vol. 45, No.3, August 2002, 268-275 .

[5] Elmasri R., Navathe B. “Fundamentals of DataBase Systems” *3rd edition. Addison-Wesley, Reading, 2000.*

[6] Soler J., Poch J., Barrabés E., Juher D., Ripoll J., " A tool for the continuous assessment and improvement of the student's skills in a mathematics course", TICE 2002, 105-110.

[7] Boada I., Soler J., Prados F, Poch J., " A teaching/learning support tool for introductory programming courses", IEEE Proceedings of ITHET 2004, 5th Annual International Conference, 604-609.

[8] Prados F., Boada I., Soler J., Poch J., " An automatic correction tool for relational database schemas", IEEE Proceedings of ITHET 2005, 6th Annual International Conference, S3C 9-14.