Auto-Adaptive Questions in E-Learning System

Enrique Lazcorreta, Federico Botella
Operations Research Center, Universidad Miguel Hernández de Elche, Elche, Spain
{enrique, federico}@umh.es
Antonio Fernández-Caballero, José M. Gascueña
Department of Computer Science and Computer Science Research Institute of Albacete, Universidad de Castilla-La Mancha, Albacete, Spain
{caballer, jmanuel}@info-ab.uclm.es

Abstract

All books entitled "Learn … with 1000 exercises" have in common the same basic principle. They aim to supply enough material to students so that they may better understand the studied subject, starting from their own practice. If there is no instructor who helps students during the reading of the book, the students will not be able to understand the subject, as the excessive amount of information provided in this kind of books does not enable learners to pursue the learning goals.

There is a great boom in e-learning through the so-called Intelligent Tutoring Systems, excellent virtual instructors which guide their learners through the reading of such kinds of books and help their learners to classify all the exercises and recommend them which ones to solve first. Nowadays instructors and teachers are entrusted to produce these books and to classify all exercises, whatever implies an overload to teachers.

In this work we introduce a scalable system that only requires teachers to write the questions and their answers. The system will classify and manage all the questions. So the teacher will obtain, with the minimal effort, hundreds of exercises at the end of the course (and for future courses) which will reinforce individually his students.

1. Introduction

Today's technology enables e-learning systems to manage a great deal of didactical material and data on student courses. A good allocation of these data permits to retrieve the full path of any student of the course in real time, and so to give him/her the most suitable material to his/her real level and specific needs.

Much of the current research works are oriented to the development of Intelligent Tutoring Systems [1], which supervise the interaction of the learner with the contents and resources of the course with the intention of discovering his learning style and to guide him in his studies [2], or to suggest reinforcing badly acquired concepts [3]. These systems use to evaluate students by defining their knowledge level about the subject.

Butz et al [4] adapt the program of the course to the needs of a student who wants to review a unique topic (for making an immediate exam, for example), so the system reveals the minimal previous concepts to study.

Brusilovsky et al. [5] define an elementary programming course in 44 topics and two or three questionnaires for each topic, consisting in 5 or 6 questions like "*What is the final value of an expression?*" or "*What will be printed?*". Their proposal consists in using adaptive annotations to show the student which questionnaires he must solve depending on the authors knowledge about the prerequisites of the topic and its needs. Their work is complementary to a previous proposal [6] where the researchers show how to parameterize all the questions so that the student may solve them without repeating the same code in the question.

Student adaptivity in e-learning systems is of tremendous interest. Instructors are set free of guiding their students by holding their own rhythm. But instructors will have to prepare new material or to upgrade the past material in order to adapt to the new programs in the system.

The remainder of this work is structured as follows: in section 2 we show a view of the management and elaboration of adaptive didactical material. In section 3 we explain a system to set instructors free of this management and that helps them to elaborate material. Finally, in section 4 we present the conclusions and future work.

2. Adaptive didactical material

Courses in e-learning systems are composed of concepts that the student must learn. To determine if a student has learnt a concept, the system proposes several questions and writes down the successes and the failures in addition to showing the results to the student. Moreover, the instructor of the course can define prerequisites for each concept, so if a student wants to study a concept, the system may suggest him to first study some previous concepts.

The proposals of Brusilovsky et al. [5,6] lose their potential when grouping concepts into topics. The reason could be the awkwardness of elaborating and classifying the questionnaires if the teacher has to group them based on the hundreds of concepts which have to be learnt by an applicant to programmer, instead of grouping the questionnaires in 44 topics. This reasoning could lead to this situation:

- A student wants to study the topic “if_else” but he doesn’t know how to use the % operator in C language. However he knows well the topic integer_operator, because he knows well the remainder of operations of the topic. The first question that the system proposes to the student is:

  What is the final value of `i`*?*

  ```
  main() {
    int i = 0;
    if (6 % 2)
      i += 2;
    else
      i++;
  }
  ```

  Imagine that the student is not able to solve it. The system will write down this failure of the student with the topic if_else and will continue ignoring the failure of the student in the % operator.
- The system notifies the teacher about the progress of the student, and the teacher takes notice of this situation. Then he changes the question replacing % by >=.
- The next day the problem occurs again with another student who “knows” relational operators but who does not understand well the >= operator.
- If the teacher decides to recover the first version of the question and to maintain the second one, the system will maintain two near identical questions and will not be able to differentiate, because both questions belong to the if_else topic. If this situation is repeated with other concepts of the course, the questionnaires will grow and lose their adaptivity to students (a long questionnaire invites not to be solved and the student will lose the interest towards the system). And all that with a great effort of the teacher.
- If the teacher decides to eliminate the questions to avoid the abandon of the student, it will have the same consequence of the traditional education: even if the system is managed by a powerful computer system, the system is not able to reuse much material elaborated by the teacher.

If the system permits the student to select his own programme, and after studying the if_else topics the student would like to review the integer_operators subject, will it be appropriate to propose the student the exercise of our example? The system will not do it because it has classified as a question of the topic if_else, which is not the objective of the student.

Our proposal is not to discard the full power of this system and to facility the work of the teacher, by liberating him of the task to classify all the questions. For this reason we will work at the concept level and we will only lead the teacher to write down the headings of the questions, the answers and at last a subjective evaluation of its difficulty level. The system will classify all questions and determine what the most suitable question that better adapts to the student in each situation is.

3. Auto-adaptive Questions

When descending to the concept level, our example would be related to the concepts of C language: `main`, `int`, `=`, `if`, `%`, `+ =`, `else` and `++`. The teacher only has to write the question and a simple text parser can report these relations to the system.

When the question is introduced, it can be suggested by the system to any student that studies someone of its related concepts and knows the rest, not mattering if there exist other prerequisite foreseen by the teacher (another delicate and subjective task of which the teacher is liberated).

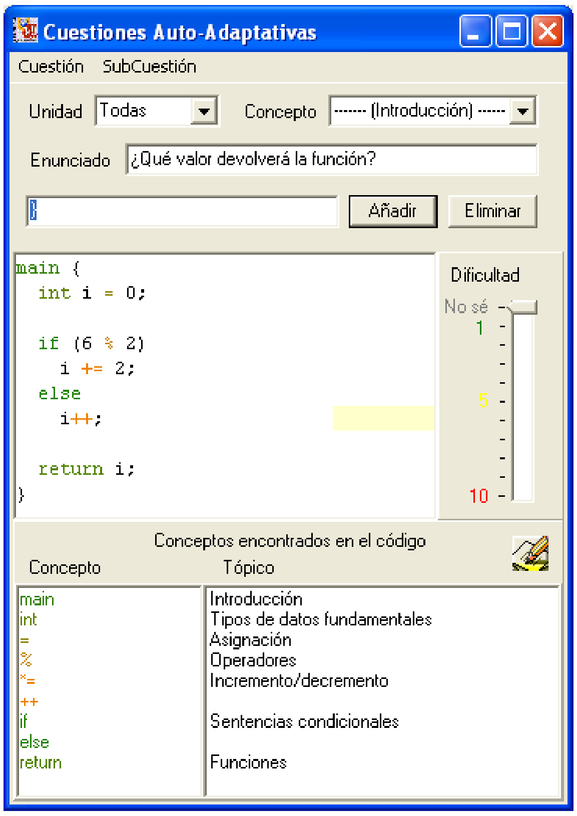
Obviously, a simple text parser does not constitute the whole adaptive education system; it is actually like an open door to prepare hundreds of questions in order that the system can adapt to the particular situation of any student. At any time, even when he is teaching lessons and a concrete example occurs to him, the teacher can invoke the Question Parser to add it as a question. And so, it is immediate to extract sub-questions (questions with slightly less difficulty and related to fewer concepts) from the question added.

Figure 1 shows the moment the question is raised for the first time, when studying the concept `return` during the study of the Function's topic. Figure 2 shows how the teacher only has to select some lines from the initial question and change the heading to get another question in the system.

In a few minutes he can incorporate all the sub-questions that make some sense (the previous one without `else`, …) without worrying about its classification. An important characteristic of the sub-questions is atomicity: it turns out to be elementary to create questions that only refer to a concept. This way, the system will have more resources to adapt to the students of the course. For example a sub-question that arises from the previous example is:

> *Knowing that variable* `i` *stores the value 3, what will be its value after the following sentence?*
> `i += 2;`

The adaptation of the system to the teacher enables that he does not lose any of his ideas. Moreover, other teachers or students could set up different question collections that serve to enrich the course contents.


Cuestiones Auto-Adaptativas
Cuestión
SubCuestión
Unidad
Todas
Concepto
------- (Introducción) -------
Enunciado
¿Qué valor devolverá la función?
Añadir
Eliminar
Dificultad
No sé
1
5
10
Conceptos encontrados en el código

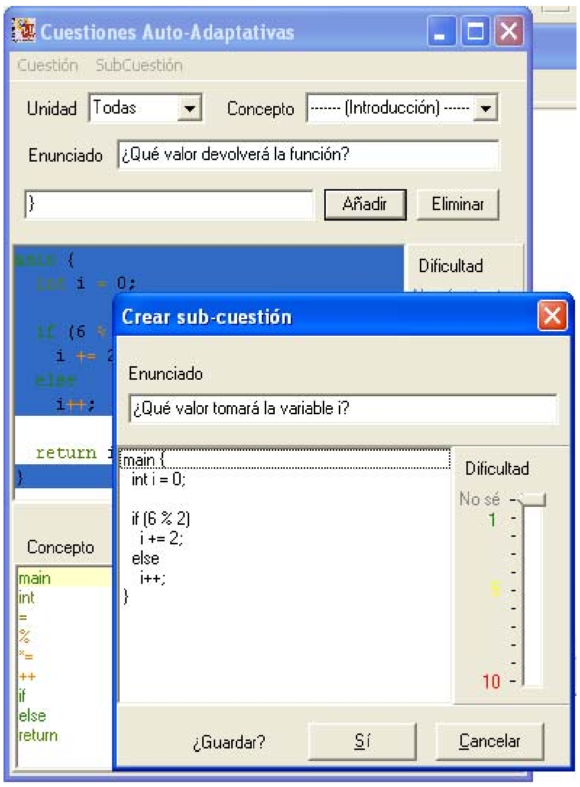
```
main {
  int i = 0;

  if (6 % 2)
    i += 2;
  else
    i++;

  return i;
}
```

Concepto	Tópico
main	Introducción
int	Tipos de datos fundamentales
=	Asignación
%	Operadores
*=	Incremento/decremento
++	
if	Sentencias condicionales
else	
return	Funciones

Figure 1: Question Parser



```
main {
  int i = 0;

  if (6 % 2)
    i += 2;
  else
    i++;
}
```

Figure 2: Adding sub-questions

3.1. System architecture

Besides the Question Parser, the system has a data base formed by four tables: TConcepts, TQuestions, TAnswers and TStudents.

Table TConcepts is common to the majority of tables for this purpose in current e-learning systems. To work with the auto-adaptive questions, the system only needs the course curriculum: a collection of concepts classified under thematic units (topics).

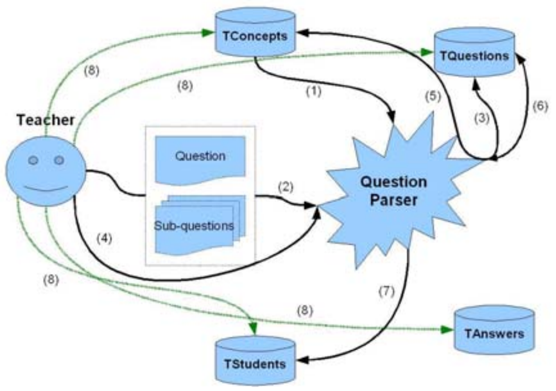


Figure 3: Teacher-System interaction

Figure 3 shows how the system starts its work reading this information to be able to use it with the Questions Parser (1). When the teacher inserts a new question (sub-question) (2), the system analyzes and stores in TQuestions this information (3):

- idQuestion
- Heading
- Source code
- Solution
- Number of different topics
- Number of different concepts
- Father Question (0 if not sub-question)
- Difficulty level (0 unclassified)
- concept1_frequency
- ...
- concept*C*_frequency

where *C* is the total number of concepts in the course.

The stored frequencies allow weighing in real time the affinity level of the question with the request done by the student. The number of different topics and concepts are useful in order to determine the simplicity (few concepts) or complexity of the question. The difficulty level assigned by the teacher can also be used by the system, but as it is just a subjective measure it will be considered on a second plane.

If the teacher needs to incorporate a new concept he can indicate it to the system by using the Question Parser (4). The system will update first the table TConcepts (5) and later it will look for the new concept in all the questions of the course to update the table TQuestions if necessary (6). Also it will add to table TStudents the field corresponding to the new concept and its right/wrong frequency will be updated as right/wrong if the concept has been found in some already existing question in the course (7).

Besides, the system allows the teacher to see the data stored in the different tables under different perspectives (8).

3.2. Student adaptivity

When a student wants to perform a session of practices the system displays its interface, allowing deciding among:

- To revise a topic or a concept
  1. as the only aim
  2. and practice also the topics/concepts with a lower punctuation
- To do a general revision
  3. of the topics/concepts with a lower punctuation
  4. of the topics/concepts with better punctuations (to learn without straining in this instant, fun learning)
  5. randomly, only with known concepts
  6. purely random
- To visualize his knowledge level in the different topics and concepts to decide what to do at this instant

Students also can decide their preferred level of difficulty of the questions as well as to use or not to use questions that already he tried to solve.

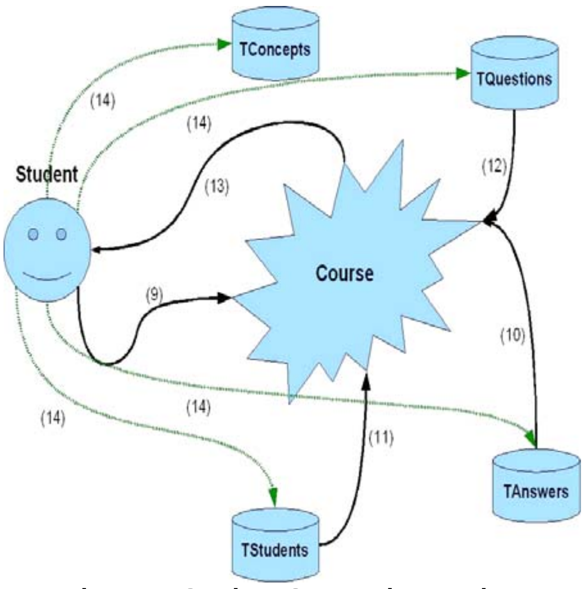


Figure 4: Student-System interaction

Figure 4 shows the underlying mechanism of the system after having received the student request for a practical session (9). The system obtains from TAnswers (10) the questions that must not be suggested to him, either because he already solved them correctly or because he does not want any question to be repeated, even he answered in a wrong way in the past. This table stores data about each answer received by the system. Its fields are

- date
- idStudent
- idQuestion
- right/wrong

The student's level of knowledge is stored in table TStudents. To be able to find the questions that better adapt to the current profile of the student, the system can look into the following fields:

- Registration date
- Final date of the course
- Grades (it might include partial grades)
- idStudent
- concept1_right
- concept1_wrong
- ...
- concept*C*_right
- concept*C*_wrong

With the information obtained in the request of the student and the information that the tables report (10, 11) the system only has to link the table TQuestions with the most appropriate criteria (12). For instance, if the student wants to revise exclusively a concept, he will obtain only the questions from TQuestions, weighted by the following features: (a) it contains the requested concept, (b) it does not contain other concepts, and (c) it contains only a few additional concepts that the student knows well. Moreover, the questions are filtered by the preferences of the student when solving only new questions and by their difficulty.

The system shows the student a list of questions arranged according to its affinity to the request (13). The student can freely choose the question to solve, answers it and the system annotates in TAnswers a new record. Next, the system updates the rest of tables and the query, and invites the student to continue practising with the matter.

Lastly, a student may also consult some views from the system tables (14).

3.3. Informing the teacher

The system will regularly inform the teacher about the usage of his questions, and it sends an e-mail when it can not help the student above a certain threshold (as a number of questions found or as an affinity to the request of the student).

4. Conclusions and future work

We have defined auto-adaptive questions and have shown a methodology that enables the teacher to simply write the questions without classifying them. It is rather the system which proposes the student the appropriate questions depending on the knowledge of the student and the objectives of the study. Moreover, the system permits to create sub-questions, and is able to inform the teacher which concepts are the most (or lest) used in his questions, which pair of concepts has been missing in their questions when recommending, or even to show questions containing these concepts that are not in the system yet.

Our system will not lead to "Learning C language by practice" by itself because the system needs quality theoretic material to achieve the success of the course.

As future work we will export the system to subjects, not so structured as programming languages, but it will allow to be evaluated by test exercises: the teacher will classify the concepts involved in each option, and random questions with multiple replies will be generated for which the teacher has prepared many options.

Acknowledgements

This work has been partially funded by the R.R.1256/04 grant supported by University Miguel Hernández of Elche – Bancaja and by the Spanish PBC-03-003 grant supported by Junta de Comunidades de Castilla-La Mancha.

References

[1] F.A. Dorça, C.R. Lopes, and M.A. Fernandes. "A Multiagent Architecture for Distance Education Systems", Proceedings of the 3rd IEEE International Conference on Advanced Learning Technologies (ICALT 2003), 2003, pp. 368-369.

[2] C.I. Peña, J.L. Marzo, J.L. de la Rosa, "Intelligent agents in a teaching and learning environment on the Web", Proceedings of the 2nd IEEE International Conference on Advanced Learning Technologies (ICALT 2002), 2002, pp. 21-27.

[3] A. Fernández-Caballero, J.M. Gascueña, F. Botella, and E. Lazcorreta, "Distance Learning by Intelligent Tutoring System.", Proceedings of the 7th International Conference on Enterprise Information Systems (ICEIS 2005), 2005, vol. 5, pp. 75-82.

[4] C.J. Butz, S. Hua, R.B. Maguire, "Bits: a Bayesian Intelligent Tutoring System for Computer Programming", Proceedings of the Western Canadian Conference on Computing Education (WCCCE04), 2004, pp. 179-186.

[5] P. Brusilovsky, S. Sosnovsky, and O. Shcherbinina, "QuizGuide: Increasing the Educational Value of Individualized Self-Assessment Quizzes with Adaptive Navigation Support", Proceedings of the World Conference on E-Learning (E-Learn 2004), 2004, pp. 1806-1813.

[6] S. Sosnovsky, O. Shcherbinina, and P. Brusilovsky, "Web-based parameterized questions as a tool for learning". Proceedings of the World Conference on E-Learning (E-Learn 2003), 2003, pp. 309-316.