Implementation of a Computer Assisted Assessment System

N. T. DOUKAS and A. S. ANDREATOS
Div. of Computer Engineering & Information Science
Hellenic Air Force Academy
Dekeleia, Attica, TGA-1010
GREECE

Abstract: - A new automated examination system called e-Xaminer is being developed, in order to support the exam process at the Hellenic Air Force Academy. The goals of the e-Xaminer are to facilitate university-level instructors with limited computer training to administer electronically delivered tests, to minimise the student’s opportunities for cheating during examinations and, of course, to save the time spent in manual grading. The instructor composes a document with the exam questions, in a user-friendly environment which requires only basic computer literacy skills. The e-Xaminer uses this document as input to produce an html form containing the test and a marking agent. The exam is delivered to the students via the local intranet and the students submit their answers. The marking agent marks the submitted answers and produces a marking report which is reviewed and approved by the instructor. If necessary, the process can be re-iterated. This paper describes the design and implementation issues which were confronted during the realisation of some critical parts of this system.

Key-Words: e-Xaminer, Computer Assisted Assessment, CAA, e-Learning, Domain Specific Languages.

1 Introduction

A novel automated Computer Assisted Assessment (CAA) tool has recently been developed by the Computer Engineering Dept. of the Hellenic Air Force Academy (HAFA) [1] for the automation of the midterm exam giving process. The reasons for developing CAA tools are mainly the increasing population of classes [2], [3], the need for effective assessment of online courses [2], the increasing demand for distance learning [4], [5] and the everlasting human efforts for saving time.

The CAA system, called e-Xaminer, has been successfully tested during the current academic year. E-Xaminer takes into account concepts of electronic assessment that have been proposed in recent years [2],[3],[6]-[11], while introducing new solutions that allow the entire process to be automatic. This paper describes the design of some critical parts of the system and some crucial implementation issues: (a) the aims of the deployment of the system, (b) test question standardisation, (c) problems in implementing the different question types and (d) the overall design strategy followed, so as to achieve the system goals.

2 Overall aims

The development of the e-Xaminer began with two principal aims. The first aim was to use web technologies in order to reduce the students’ opportunities for exam time cheating. Additionally, it was intended to help university-level course instructors reduce the amount of tediousness involved in the process of examining students, thus making examinations more agreeable for both examiner and the examinee. A side effect of the use of this system is that, by standardizing the marking process, truly objective assessment criteria were introduced, that gave to both the students and the instructors an increased sense of fairness in grading.

2.1 Confronting cheating

A key issue in exam taking is cheating. As far as distance learning is concerned, cheating is easy when there is no biometric identification infrastructure, as is the case with most such systems. In this case, we cannot identify the person giving the exam. As far as local learning is concerned, cheating is easy when the students sit very close to one another, as is the case with most PC labs. Five countermeasures against cheating have been proposed in [6]:

1. Frequent delivery of tests so that another person helping the examinee may not be available too often.
2. Use of judgment questions rather than questions based on memorization
3. Delivery of different questions to different students
Countermeasure no.1 is a matter of policy, not a technical one. Let us examine how e-Xaminer incorporates the rest countermeasures. e-Xaminer supports 9 types of questions/problems, so that it can test many different skills of the examinee and not just memory. In types 5 and 7 the examinee must perform calculations, based on a plan he/she has first to carry out. Another technique we have recently employed is the use of textareas next to the answer boxes, where the examinee writes his/her reasoning or other comments. This technique has been used in the programming or assembly language exams, where the student is asked to write small pieces of code; then, in the text area he is asked to type the comments related to the commands. In type 5 and 7 problems, the student types there his reasoning. In this way, the instructor may check the comments later to make sure that the examinee knows how to solve the problem. And it is more difficult to copy the comments at a glance.

As far as the 3rd countermeasure is concerned, two solutions are provided: the first is parametric problems (question type no.7) and the second is the scrambling of the question order. Preliminary experimental results have shown that these countermeasures work.

As far as the 4th countermeasure is concerned, e-Xaminer will soon incorporate a count-down timer; when the timer hits zero, the form will automatically be submitted to the server.

2.2 Improving the nature of examinations

The second aim of the e-Xaminer was to automate the examination process. The main problem is that, in general, university level instructors may not be experts in web technologies. This may be true even for instructors of technical subjects. It was hence necessary to parameterise the process of automated examinations, so as to hide the technical details from all the parties involved in the process. In the following sections, the parameterisation used for setting the questions and for assessing the answers will be explained.

3 Standardising testing

3.1 Standardising exam questions

In order to implement this system, it was necessary to standardise some types of questions that where to be used by instructors. The initial set of standardised question types was limited (just 9 types of questions that required just textual or numerical answers). These types are [1]:

1. True/false questions [type=1]
2. Multiple choice questions with one correct answer [type=2]
3. Multiple choice questions with multiple correct answers [type=3]
4. Questions with Short answers (textual or numerical); enumeration / quotation of specific terms [type=4]
5. Simple problems whose answers are arithmetic, algebraic or computer language expressions [type=5]
6. Simple theoretical questions from the textbook where the answer is a phrase [type=6]
7. Parametric problems whose answers are arithmetic, algebraic or computer language expressions [type=7]
8. Questions where items had to be matched with others from a limited set [type=8]
9. Composite Questions; a combination of one or more of the above types [type=9].

<table>
<thead>
<tr>
<th>Type</th>
<th>Question Type</th>
<th>HTML Form Type</th>
<th>Implications</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>True/false questions</td>
<td>Radio button</td>
<td>No real problems if the questions are clear, the student cannot be misinterpreted by the automated marking agent</td>
</tr>
<tr>
<td>2</td>
<td>Multiple choice questions [many options]</td>
<td>Radio button</td>
<td>None</td>
</tr>
<tr>
<td>3</td>
<td>Multiple choice questions with multiple correct answers</td>
<td>Checkbox</td>
<td>None</td>
</tr>
<tr>
<td>4</td>
<td>Short answers, enumeration / quotation of specific terms</td>
<td>Textbox</td>
<td>Language variety, character set</td>
</tr>
<tr>
<td>5</td>
<td>Simple problems whose answers are arithmetic, algebraic or computer language expressions</td>
<td>Textbox</td>
<td>Intermediate results, partial credit, expression decoding</td>
</tr>
<tr>
<td>6</td>
<td>Simple questions from the textbook: ANSWER=PHRASE</td>
<td>Textbox</td>
<td>Language variety, character set</td>
</tr>
<tr>
<td>7</td>
<td>Parametric problems whose answers are arithmetic, algebraic or computer language expressions</td>
<td>Textbox</td>
<td>Intermediate results, partial credit, expression decoding</td>
</tr>
<tr>
<td>8</td>
<td>Correspondence Questions</td>
<td>Textbox</td>
<td>None</td>
</tr>
<tr>
<td>9</td>
<td>A combination of the above</td>
<td>A combination of the above forms</td>
<td>Combinations of the above problems</td>
</tr>
</tbody>
</table>

Table 1 shows the standardised types of questions, along with the html form object used for
Despite the limited set of question types, the experience of the first semester of testing (approximately a dozen tests in classes of three different specialisations with population varying from 10 to 35 students) showed that, using the capabilities of html for presenting, test questions could be designed to be varying as well as challenging.

The automation of the marking process had to cope with several problems. The first set was due to linguistic issues. The Greek language is very rich in the number of different forms in which the same word may appear (four cases for nouns and three for verbs) as well as in the number of ways in which a word may appear in a sentence. Additionally, students used a significant number of different equivalent terms to answer their questions.

The terminals used by the students, operate with a dual character set (Greek and English). Answers may hence be typed in any combination of Greek and English characters and any combination of lower and upper case letters. The Greek language is also renowned for its complex spelling and its accents that students tend to mix up. Furthermore, some of the terminals used Unicode character sets, while others still used ASCII and its Greek variants.

For arithmetic answers, it was necessary to have the ability to give partial credit for correct intermediate results. For algebraic questions, it was also required that it be possible for the instructor to decide on ways to give partial credit for intermediate correct expressions. For these questions, it was also a challenge to decode complex algebraic expressions.

The above observation was also true when the answers to questions where computer language expressions. In this case further implications arise since answers may be implemented using a variable number of statements. The parameterisation of the marking process had to allow in this case for marks to be accredited depending on the instructor’s requirements: longer answers (i.e., containing a larger number of statements) may get the same or a smaller number of marks.

3.2 Standardising the implementation

The implementation of the e-Xaminer uses the concept of Domain Specific Languages (DSL) [12] to produce the code necessary for running the tests. This approach implies that neither the instructor (the author of the test) nor the examinees need to have any experience in Software Engineering. The examiner writes the test using a simple interface (e.g. a limited subset of the capabilities of an html editor such as MS Front Page or a simple text editor such as Notepad). Perl scripts generate the code necessary for the operation of two entities: the testing and the examination agent.

The testing agent is a set of perl, perlscript and HTML files that are used in the delivery of the tests. These are the files that are published the exam web page in the local intranet and contain all the code necessary for the students to submit the test form and create a humanly readable answer file that is required both for the student to be able to review their answer as well as for the school archive. The testing agent is also responsible for generating the parameters required in parametric questions and scrambling the question order where necessary. A test is implemented as an active page that uses standard html forms and perlscript in order to validate and collect the student’s answers. The page may contain specific html form objects, namely text boxes, text areas, check boxes and radio buttons. The pages are required to use only standard html features, so as to ensure portability between browsers. The exams included in this study where tested in both Microsoft Internet Explorer 6 and Mozilla 1.7.12 had worked without problems. Perl need only be installed in the web server which is used.

The marking agent contains the code necessary for checking the answers supplied by the students against those provided by the instructor. It generates an extended marking report for each student, a summary report for the class, as well as class statistics. The instructor reviews the reports and if they are not approved, the marking process is iterated. Note that the marking agent, where it is imperative that the correct answers are contained, is kept physically separate from any other system entity to which the students may gain access. This is done as an extra precaution to avoid exam-time cheating. The concept of the e-Xaminer is illustrated in Figure 1.

Figure 1: Structure of the e-Xaminer
Currently the tests are not allowed to include interaction with non-textual objects (e.g., clicking on or selecting items from an image). This feature will be added in future versions of the e-Xaminer. However, interaction with such objects has been indirectly achieved via combinations of static images and text boxes.

4 Pilot application of the e-Xaminer

4.1 Deployment of the system

The e-Xaminer has been used to deliver examinations on various classes of the Div. of Computer Engineering & Information Science of the Hellenic Air-Force Academy. These included Microprocessors, Programming Languages, Computer Networks and Digital Electronics. The students which were tested came from three different course disciplines of the Academy, namely Flying Officers, Telecommunications and Electronics Engineers and Air-defense Controllers, with class sizes from 10 to 35 persons. Testing was carried out during the current academic year.

Overall it was found that, despite the experimental nature of the system, the use of the e-Xaminer produced time – saving for the instructor. Furthermore the novel concept of the automated exam was very well accepted by the students, who seemed to be less stressed by the prospect compared to their reaction to a paper exam. The fact that tests had differences between them overwhelmed the students, discouraged cheating and earned their respect for the new system. There were no complaints like “my question was harder than his”.

4.2 Example of a parametric problem

It is worth giving an example of type 7 questions. The principle is to use the same problem with different sets of parameters. Parameterisation is based on the last digit of the examinee’s registration or ID number. Example problem: “A company owns a set of class B IP addresses. The company has x LANS with an increase rate of 40% during the next 10 years. Each LAN connects y computers, with an increase rate of 85% during the next 10 years. Which is the best way of sharing the class B host bits between subnets and computers?”

Table 2 shows 10 (x,y) pairs, each associated with a digit from 0 to 9, representing the last digit of the examinee’s ID number. There are strong chances that two neighbouring students will have different last digits, thus, different sets of parameters. Yet another improvement is possible for small classes: to provide a different set for each student.

<table>
<thead>
<tr>
<th>Examinee’s ID last digit</th>
<th>x (LANs)</th>
<th>y (hosts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>4000</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>1200</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>900</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>650</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>590</td>
</tr>
<tr>
<td>5</td>
<td>23</td>
<td>480</td>
</tr>
<tr>
<td>6</td>
<td>30</td>
<td>310</td>
</tr>
<tr>
<td>7</td>
<td>50</td>
<td>220</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>120</td>
</tr>
<tr>
<td>9</td>
<td>130</td>
<td>80</td>
</tr>
</tbody>
</table>

4.3 Problems encountered and solutions proposed

The problems encountered due to the nature of the questions have been already analysed. These were:

1. Linguistic:
   a. There exist many equivalent terms
   b. There exist many different (often uncorrelated) forms in which a word (term) may appear.
   c. There exist multiple ways in which the words may be re-arranged in a sentence (the Greek language allows virtually any permutation of the same words, however only sometimes does the meaning change)

2. Character - set issues. The answers contained many combinations of:
   a. Greek and Latin characters
   b. Accents and spelling mistakes
   c. ASCII variants and Unicode
   d. Uppercase and lowercase versions of the above.

3. Decoding of (and giving partial credit for partially correct parts of) algebraic expressions or computer language commands.

4. Locating and giving partial credit for intermediate results (algebraic or numeric).

The solution to problems (1a) and (1c) above is currently based on the iterative marking process. The instructor adds alternative terms so as to ensure that all student answers are covered. Instructors were shown to rapidly acquire experience on setting model answers for questions they set that successfully cover the entire range of the students’ answers. Future extensions may
include an automated system for retrieving alternative keywords. More specifically, for problem (1c), linguistic criteria need to be employed that will prevent the student from getting credit for answers that just contain the right words without giving the correct meaning (e.g. keyword near keyword).

For problem (1b), word correlation was exploited. The root of a word was used as the keyword to be searched for.

The initial approach to problem (2) was to predict erroneous or scrambled spelling and insert as keywords misspelled words, while ignoring all case (convert everything to lowercase). As this is a tedious process, a new approach is currently being adopted that converts all answers to lowercase Latin characters according to the standard used for converting names in passports. This is a many-to-one correspondence which eliminates spelling and hence resolves the issue. The efficiency of this approach will be evaluated in future tests.

The solution to problem (3) adopted so far is to set questions which dictate one specific way of answering (e.g. suggest variable names so as to avoid variety). A more robust approach would be to include some form of lexical or syntactic analysis, of the type that a compiler would include. Such an approach has been reviewed in [13]. The same approach could be used for problem (4) for the case of algebraic questions. For numerical questions, the instructor may set intermediate answers that give partial credit.

5 Future extensions of the e-Xaminer

The e-Xaminer has been used for midterm exams of students of all four years of the Hellenic Air-Force Academy during the current academic year. The system will be further developed so as to:

1. Allow the students to answer questions by pointing on images. This extension will represent a novelty even over paper examinations.
2. Include the capability of enforcing the allowed time for the exam.
3. Allow the system to automatically retrieve alternative terms.
4. Include linguistic criteria so as to search for the meaning of a phrase rather than keywords.
5. Include lexical and syntactic analysis for algebraic and computer language questions.
6. Strengthen security and fight cheating attempts, we shall add a registration module; the user will first pass through a login and a logout process when they submit their answer form.
7. Introduce strict enforcement of time limits.

The e-Xaminer will continue to be used for progress examinations during the current and the subsequent semesters. It is intended that the system be used for final exams as soon as possible. Results will hence be obtained for the new approach to the misspelling problem.

Future extensions of the system could have some form of interactive testing included (e.g. creating a file in another program and submitting that file for assessment). This is a suggestion that has come, surprisingly enough, from students and shows that, if nothing else, the new system is well received.

Finally, another ambition is to incorporate e-Xaminer into our pilot distance learning course on Computer Networks [14] for e-assessment and self-assessment.

6 Conclusion

The implementation of a Computer Assisted Assessment system, called e-Xaminer, conceived to support exam giving at the Hellenic Air Force Academy was presented. The primary goals and advantages of this system are: to facilitate instructors with limited or no computer training to electronically assess their students; to stiffen cheating; to minimise human errors and increase fairness and --of course- to save time and effort to the instructor.

The process is as follows: the instructor composes the exam questions and the corresponding answers and grading policy (weight of each question, alternative solutions, partial credits etc.) as documents, in a user-friendly environment that requires only basic computer literacy skills. The e-Xaminer interprets this Domain Specific Language and produces all the code necessary in order to perform the test. The exam is delivered to the students via a web interface; automatically generated scripts mark the students’ answers. Another important feature of the e-Xaminer is the possibility of reviewing the grading policy a posteriori, if it turns out that a problem was too difficult or if not foreseen alternative solutions emerge.

The problems encountered during the implementation of this system have been analysed and possible solutions have been proposed. Guidelines for future work have been given. It was observed that the use of the system made the overall process of assessing students fairer, less tedious and more appealing to both students and instructors.
References:


