A COMPUTER BASED ASSESSMENT SYSTEM FOR UNDERGRADUATE ELECTRICAL ENGINEERING MODULES

P.A. van Vuuren *, **A. Alberts** † **and L. Hager** ‡

* *School of Electrical, Electronic and Computer Engineering, North-West University, Private Bag X6001, Potchefstroom, 2520, South-Africa. E-mail: Pieter.VanVuuren@nwu.ac.za*

† *E-mail: Andreas.Alberts@nwu.ac.za*

‡ *E-mail: Louw.Hager@nwu.ac.za*

Abstract: Good assessment is an essential component of good learning. Marking test and exam papers for large classes is however a time-consuming and repetitive task. This paper reports on the design and implementation of a computer based assessment (CBA) system suitable for undergraduate engineering modules. This prototype CBA system is capable of automatically grading relatively open-ended analysis and design problems and is also relatively user friendly. This CBA system consists of two separate subsystems, namely a test delivery subsystem and an automatic grading subsystem (which is the contribution of this paper). Automatic grading of the student answer papers is performed in Matlab®. The grading subsystem can automatically grade typed student answers containing symbolic mathematical expressions as well as numeric answers. Grading is performed accurately, consistently and very fast compared to a human lecturer. Students are given partial credit for partially correct answers and feedback is given in a very intuitive form quite close to the traditional manner in a pen-and-paper test. The system has been successfully used to take final exams in a second year engineering module on basic circuit analysis.

Key words: Computer based assessment, engineering education, automatic grading.

1. INTRODUCTION

Large classes are common in modern university level education. This is especially true of first and second year engineering modules. This situation places enormous pressure on lecturers to give timeous feedback to students after tests or exams have taken place. Despite being an onerous task grading of test papers is very important, since good assessment (including detailed feedback) forms one of the pillars of good learning.

Traits of a good test or exam are: accuracy, consistency and speed. The *accuracy* of a test refers amongst other things to the correctness of the grading of the student's answers [1]. Tests should also be marked *consistently* for all students in the class. Lastly, students should also receive speedy feedback on the correctness or quality of their efforts. The *speed* with which students receive their grades has a huge impact on the effectiveness of the assessment in the learning process [2]. This is obviously becomes a significant problem in large classes.

One solution to this problem is through the use of multiple-choice questions. Computer-based assessment (CBA) systems making use of multiple-choice questions can give near-instantaneous feedback, but students are often critical of the inability of multiple-choice type assessments to give partial credit for the method that they used [3]. Large problems requiring multiple stages of calculation furthermore aren't suited to multiple-choice questions. This limitation can be circumvented by dividing a large problem into smaller sub-problems which in turn can be assessed by means of multiple-choice or fill-in-the-blank type questions [4]. It is however our opinion that such an approach carries the risk of guiding students through the solution of larger problems.

Despite their limitations, tests consisting of so-called "teacher-supplied answers" (e.g. multiple-choice, fill-in-the-blank and true/false questions) still form the mainstay of most modern CBA systems e.g. OASIS [5], I-ASSESS [6] and QuestionMark's Perception [7]. Modern CBA systems are however capable of much more than mere multiple-choice questions as shown in [1]. In their paper Scalise and Gifford propose a taxonomy of e-learning assessment on the basis of the degree to which the problem is constrained and also the relative complexity of the problem [8]. From this taxonomy one can conclude that modern CBA systems can assess almost anything ranging from true/false questions to written essays [1]. It should come as no surprise that CBA systems have been used in a wide variety of disciplines ranging from hydraulic engineering [9] to pedagogical psychology [10].

One approach to extend the capabilities of CBA systems entails making use of existing technology to automate certain aspects of the grading process. Harnessing the power of symbolic mathematics engines, some CBA systems allow students to give answers in the form of mathematical formulae. Determining whether the student's answer is mathematically equivalent to the memorandum answer is performed by means of software routines. Examples of such CBA systems are AiM [11], CABLE [12] and STACK [13]. These CBA systems have been applied successfully to assess first year calculus and algebra modules. It should however be noted that the type of questions used in these systems are still relatively simple convergent problems. Another disadvantage of STACK is

that it assigns partial credit to student answers by means of so-called ”potential response trees” (directed graphs) that are cumbersome for a lecturer to create.

Another example of CBA systems capitalizing on existing software technology is the use of computer language compilers to automate the grading of computer programming tests. Since modern compilers can give quite detailed feedback on syntax errors and scripts can be written to automate the testing of working programs, it is no small wonder that even students' computer programs can be semi-automatically [14] or automatically [15] graded by modern CBA systems. In fact, it is even possible to automatically assess not only whether a student's program is syntactically and semantically correct, but also whether the student's program exhibits more advanced programming skills (such as meeting execution time and/or memory constraints) [16].

An alternative approach is to endow the computer with a measure of artificial intelligence to allow it to recognize numerous variants on the correct answer(s). One such example is a CBA system designed to assess practical IT skills of students [17]. This system makes use of a rule-based expert system to give feedback on the student's efforts. Rule-based grading has also been applied for the automatic assessment of freebody diagrams that are interactively drawn in a biomechanics module [18]. Bayesian networks have been used to automatically assess the performance of students performing a practical exercise in a virtual electronic laboratory environment [19]. The behaviour of the students is captured during their interaction with the virtual laboratory (by monitoring their mouse clicks and keyboard strokes). Each student's knowledge, skills and insight in the particular experiment is then deduced by the Bayesian network from the observed behaviour and converted into a grade for the particular student.

Machine learning techniques have furthermore been used to automatically assess the quality of children's argumentation in an elementary school level natural science module [20]. The children verbalize their arguments in free text, from which keywords are parsed and the elements in a Toulin argumentation model identified. Natural language processing has also been applied to automatically mark short answers in an object-oriented programming course [21].

Although only indirectly related to either formative or summative assessment the work of Hoeft *et al.* is noteworthy for their automatic assessment of computer drawn concept maps (in order to assess student's knowledge representations) [22]. Another graph-based approach for the assessment of students' structural knowledge can be found in [23] where quantitative features are extracted from student-drawn pathfinder graphs with the objective of predicting the students' performance in exams taken at a later date. Recently, an interactive computer-based assessment system was presented that has been used for formative assessment of computing and technology students' argumentation ability by means of argument maps [24]. In this system students are presented with a blank argument map which has to be populated from a list of options. This system is implemented by means of OpenMark [25] in Moodle [26].

A common denominator of a number of the above mentioned examples is that *advanced CBA systems* (CBA systems that can automatically grade questions that allow students more freedom to compose their own answers to the questions) are often limited in scope. Applying a CBA system that was designed with one particular subject in mind to another subject is sometimes difficult. Another limitation of advanced CBA systems is the ease of use for the lecturer. Providing a computer with the necessary alternative solutions to empower it to automatically grade highly variable answers often becomes a very time-consuming task.

In view of the above mentioned limitations we decided to design and implement a CBA system suitable for second year engineering modules. This system had to be capable of automatically grading relatively open-ended analysis and design problems and also be quite user friendly (requiring minimal additional skills from the students). The design process for this CBA system was guided by the following criteria:

1. Accuracy. Answers should be graded correctly and reliably.
2. Consistency. All students should be treated the same.
3. Ease of use. The prerequisite skills for all users of the CBA system (both students and lecturers) should be minimal. Furthermore, the feedback given by the automatic grading software should also be intuitive.
4. Speed.
5. Partial credit. Students should be given credit for the process that they followed to arrive at an answer, even if that process itself is only partially correct.
6. It must be possible for the student to write a closed-book test on his/her own computer.

The CBA system which is the subject of this paper consists of two subsystems: a test delivery subsystem and an automatic grading subsystem. The test delivery subsystem is responsible for delivering the test to the students and collecting their answers. The automatic grading subsystem is responsible for grading each student's answers and presenting the results in a format that gives useful feedback to both the student and lecturer.

The main contribution made by the above mentioned CBA system is that students are allowed to answer relatively open-ended questions, which are then graded automatically.

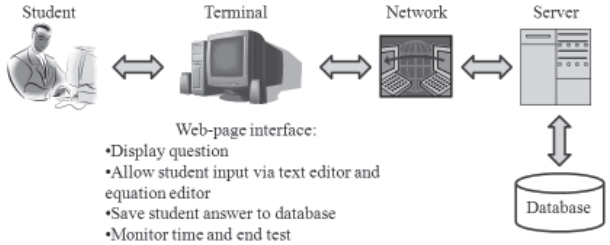

Figure 1: CBA test delivery subsystem

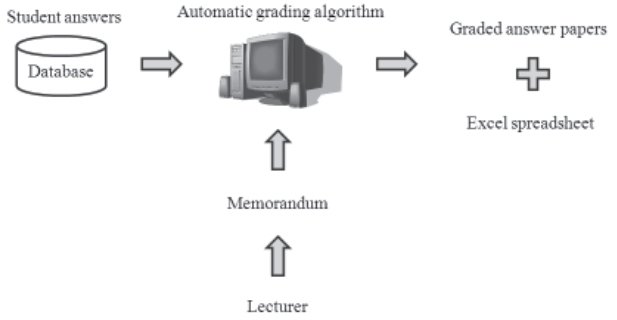

Figure 2: CBA automatic grading subsystem

In this paper the spotlight falls on the automatic grading subsystem. Section 2 briefly summarizes the structure of the entire system. More detailed discussions on the test-delivery subsystem and the grading subsystem are given in respectively sections 3 and 4. The entire CBA system was implemented in a second year engineering module during the first semester of 2012. Section 5 evaluates the performance of the automatic grading subsystem during its trial implementation. Recommendations for future work form the topic of the last section of this paper.

2. CBA SYSTEM OVERVIEW

The test delivery subsystem (figure 1) consists of a central server that communicates to remote computers via a computer network. A web-page interface is used to deliver the test to the students. Each question in the test consists of a field showing the question and a text box in which the student can type his/her own answer. When the student is done with a particular question or the entire test he/she clicks on a ”submit” button which causes the student’s answers to be sent back to the server where it is stored in a database.

Currently, the student answers are graded offline. After the test has been completed, the automatic grading subsystem (figure 2) queries the database for each student’s answers and then grades them by means of the memorandum supplied by the lecturer. After each student’s entire set of answers have been graded, the answers are saved in a LaTeX file, which can be compiled into a PDF file showing both the student’s answers as well as the traditional red tick-marks indicating correct responses. When the entire class has been processed, an Excel spreadsheet is generated showing how each student fared in each question and also giving the final grade of the student.

As we can all appreciate, grading exam papers is largely a repetitive task. Depending on the nature of the question, some knowledge and understanding of the subject matter might be required to make sense of the students’ answers. Examples of such questions are open-ended analysis or design problems. Therefore, the fundamental hypothesis upon which the automatic grading subsystem of this CBA prototype is based is as follows.

Provided that the question, answer and memorandum are sufficiently detailed and structured, automatic grading doesn’t require true knowledge or insight into the specific subject being assessed.

The above mentioned hypothesis can only be valid in a real-life examination scenario if the potential variability in student answers can be constrained to be within acceptable limits. The variability in the student answers to a question can be limited in three different ways:

(a) By constraining the *question*. This topic is briefly addressed in the discussion on the test-delivery subsystem in section 3.2.

(b) By including different options in the *memorandum*. This tactic is often used in traditional pen-and-paper exams, where the lecturer beforehand knows which different strategies will probably be followed by the students and then accordingly includes all of the options in the memorandum.

(c) By endowing the automatic *grading program* with a measure of simulated intelligence. By *simulated intelligence* is meant a program consisting of a number of conditional statements that has the appearance of intelligent behaviour on very small set of tasks.

3. THE TEST-DELIVERY SUBSYSTEM

The test delivery subsystem is built on web based technology, in order to ease its deployment in modern heterogenous computing environments. It consists of two major parts, namely the web site on which the test is administered and the environment in which the test is delivered.

The test delivery software is built using a traditional LAMP (Linux, Apache, MySQL, PHP) stack. Written in PHP, it dynamically renders web pages showing the test questions and accepts form data from the clients containing the student’s answers to the test questions.

Modern web environments are very open by default, posing significant risks when summative tests are delivered using it as a medium. We overcome this obstacle by delivering the test in a secure environment, using a live

boot Linux client, configured with only the necessities for the test and nothing else. This way, otherwise insecure systems can be made more secure by deploying a non-intrusive temporary environment on the client machine which is not under the end user's control.

More details on the various components of the test-delivery subsystem are discussed in the following few subsections.

3.1 Test-delivery software

The CBA test system is hosted on a Linux server running the CentOS distribution. CentOS is a freely available community driven rebuild of the sources of the commercial Red Hat Enterprise Linux distribution, widely renowned for its reliability and stability. We use the Apache web server, the PHP scripting language and the MySQL database server in our test system deployment.

Tests are encoded in text files using a very simple markup style, in order to code and deploy them easily. Students are registered in a database table containing their user IDs and passwords, so that only authorized users can log into the test. When a student successfully logs into the test program, the directory containing the encoded tests is parsed to determine which tests are available to the user. This information is written to a table that keeps track of session-specific parameters, such as the time the test has started, the amount of time left on the tests and the amount of times the test has been submitted. The available tests are then presented to the user.

When a user clicks on a test, the encoded text file containing the test is parsed and converted into a HTML page that is displayed in the browser. Currently, the system allows for auto-shuffled multiple choice, as well as free form text style questions.

Answers are saved when the user clicks on the submit button at the bottom of the test. An enhancement that can still be made is to submit answers in the background by means of AJAX queries as soon as they are entered, so that they are immediately persisted and to prevent the user from having to submit periodically [27].

3.2 The equation editor

The *lingua franca* of most engineering and scientific disciplines is mathematics. It is therefore also very important to allow students to express their thoughts in a test or exam by means of mathematical expressions. Whereas writing math by hand is quite easy, typing mathematical expressions is a different story all together.

Various standards exist for typesetting mathematical symbols e.g. MathML (mathematical markup language) and LaTeX. Of the available standards LaTeX is probably the most compact and human-readable representation of mathematical expressions. Unfortunately learning to type math in LaTeX represents quite a steep learning curve which would be unreasonable to expect from students enrolled in e.g. a module on electric circuits. This problem can however be addressed by means of an equation editor which allows the user to construct WYSIWYG mathematical expressions by means of a mouse and drop-down menus and also convert it to the corresponding LaTeX code. One such equation editor is EqualX [28]. EqualX is a freeware equation editor primarily designed for the Linux operating system. This allows it to be relatively easily incorporated in the Ubuntu live boot disk used for this CBA test delivery subsystem.

An example of EqualX is shown in figure 3. Numerous mathematical symbols and templates can be accessed via mouse from the dropdown menus at the top of the screen. When anyone of the buttons in the dropdown menus is pressed, the corresponding LaTeX code is shown in the bottom textbox. The keyboard can then be used to construct any conceivable mathematical expression. The purpose of the centre window is to show the final typeset version of the LaTex code (in human-readable form). The centre window therefore also serves the important purpose as quality control of the code that the student has "typed". It is actually quite simple: if the expression in the centre window looks correct (i.e. if it looks similar to what the student intended to write by hand), then the underlying LaTeX code is also correct.

At this point, the student must copy and paste the LaTeX code in the bottom window of the equation editor to the textbox in the test browser. The numerical answers and LaTeX code typed in by the student will then later be graded by the automatic grading subsystem.

In order to limit the potential variability in the students' answers, each question is accompanied with a list of permissible variables to be used in the answers. The LaTeX versions of these symbols are also given in the test browser. These symbols can then be copied and pasted into the equation editor to construct the expressions that the student desires. In this manner the process of typing math by means of an equation editor can be speeded up significantly. In fact, is is our experience that the entire process of typing in answers by means of the equation editor has only increased the time required for students to complete a test by a factor of at most $\frac{3}{2}$.

3.3 Live boot environment

In order to lock down the workstations the tests are being displayed on without installing any software or altering them in any way, we deployed a live boot image of the Ubuntu Linux distribution containing a minimal set of software packages. The live boot distribution loads itself from a DVD-ROM into a RAM disk on startup, and therefore does not interfere with the configuration of the computer in question in any way. Users writing tests in this stripped down environment only have access to a minimalistic browser and the equation editor software, and do not have root access to alter this configuration. This solution is similar to the zip disk approach proposed by Ko and Cheng [29] in that students are issued with

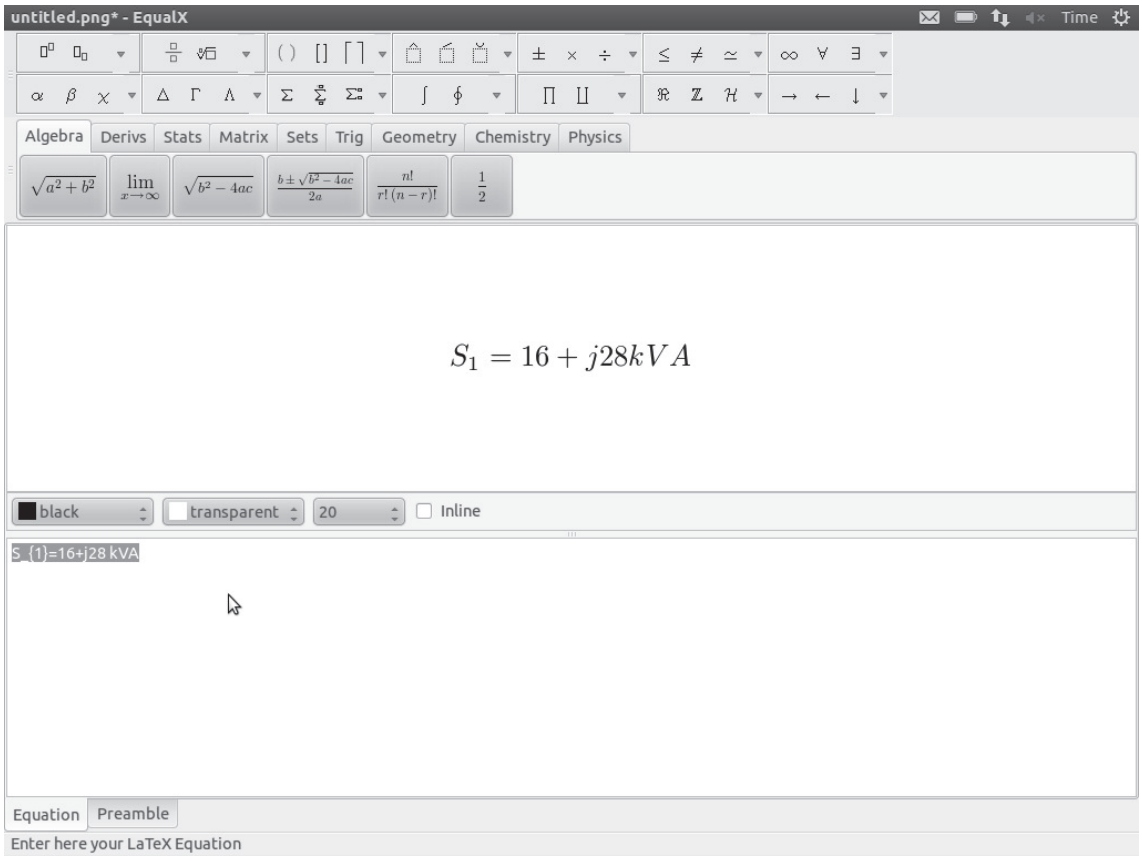

Figure 3: The EqualX equation editor

the live boot DVD prior to a test. Our solution however allows all student answers to be conveniently stored on a central database by means of a tightly controlled network connection.

3.4 Authoring a CBA test

At this point it might be instructive to take a look behind the scenes at how a test is authored by the lecturer in this CBA system. The test question and any necessary figures are produced by the word processing software of the lecturer's choice. The test delivery subsystem however requires the test in a text file format, which precludes the use of figures, tables and mathematical expressions as in e.g. a normal Word-file. This limitation can be circumvented by including each question in its entirety as a picture file. The result is a text file with numerous references to picture files, each containing a separate question.

4. THE AUTOMATIC GRADING SUBSYSTEM

4.1 Subsystem layout

Grading a test or exam paper by means of a memorandum is in essence a pattern recognition problem. It entails comparing a student's answers with the information given in the memorandum. If a match occurs, then suitable credit should be assigned to the student. To facillitate feedback to the student, tick marks are traditionally used to indicate both correct answers as well as the number of marks earned by each correct answer.

The process of reading a student's answer and comparing it to a memorandum can be easily automated. Figure 4 shows the main sequence of events in the automated grading subsystem.

At first, the computer reads the memorandum encoded in a text file and stores the correct answer(s) to each question separately. Secondly, Matlab querries the MySQL database containing the student answers to determine the class list and determine the number of students in the class. Then a main loop is engaged which cycles through all of the students in the class. For each student, the program retrieves his/her answer to each question separately. The answer is then converted into a format suitable for later LaTeX compilation. Finally, each answer is compared with the corresponding part of the memorandum. This process is denoted by the red rectangle in figure 4 and consists of further operations which will be discussed shortly. After all of the students in the class have been processed in this manner, a final summary Excel report is generated. This report indicates how each student fared overall as well as in each question individually.

Obviously, the heart of the entire grading subsystem is the

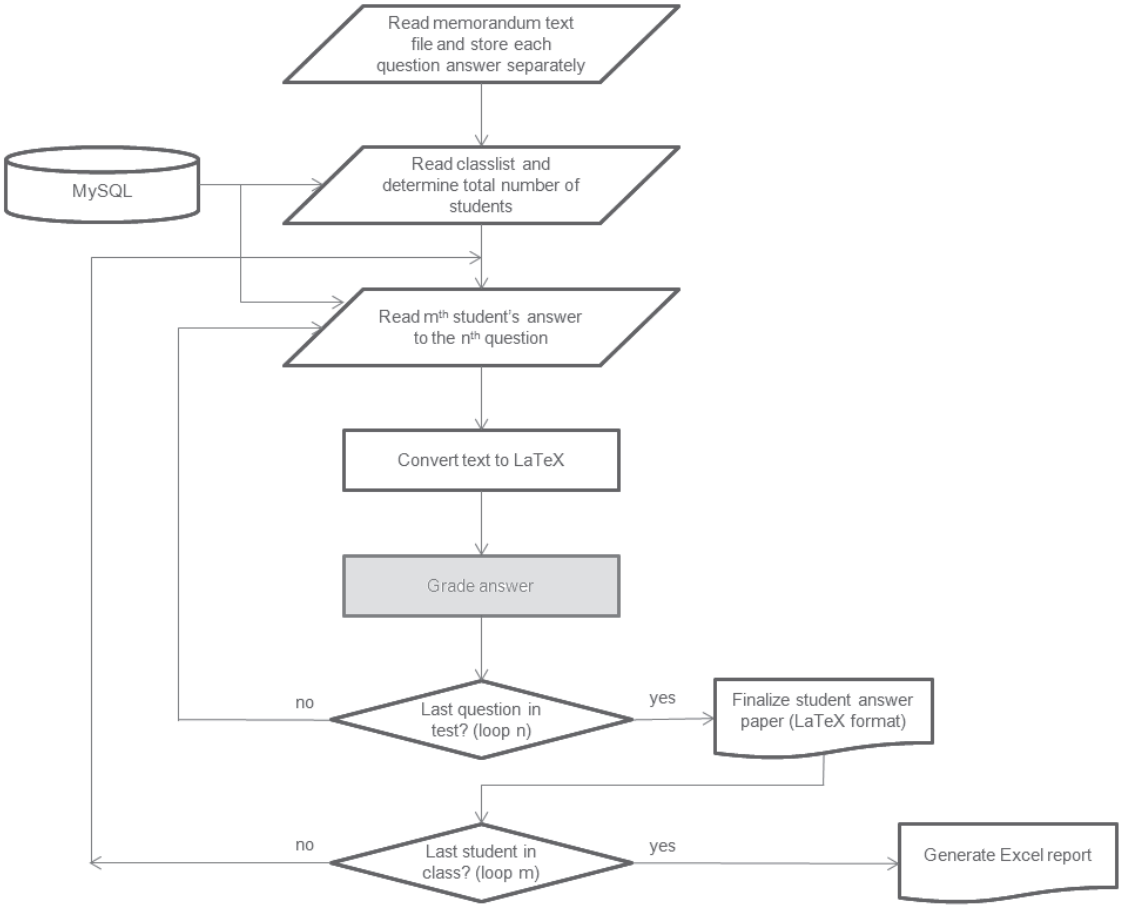

Figure 4: Main flowchart of the automatic grading process

component responsible for comparing a student's answer to an individual question to the corresponding section in the memorandum. Highlighted in red in figure 4, this component itself consists of a loop calling different subroutines. This loop entails that each line in the memorandum is read, interpreted and suitably used to grade the student's answer.

With the exception of the preamble, each line in the memorandum has the following structure: `GRADING COMMAND:(N) answer`. The meanings of the three different components in the line are as follows:

- `GRADING COMMAND` - This is a reserved word that indicates how the computer must treat the text given in `answer` and how the student is given credit. There are currently only six grading commands namely: `SYMBOLS external`, `SYMBOLS internal`, `TICK`, `TickNumEq`, `ANSWER` and `TickEq`. Each of these commands will be introduced shortly.
- `(N)` - An optional input is the number of marks allocated to a correct answer. The default value is one.
- `answer` - A free-form string, number or LaTeX mathematical expression which indicates a correct answer.

As an example, consider the following memorandum line:

```
ANSWER:(3) R_1 = 10 k\Omega.
```

The above line instructs Matlab to search through the student's answer to question x for a statement in which the student has calculated that $R_1 = 10$ kΩ. This correct answer is worth three marks.

For each line in the memorandum answer to a question, the following process is then followed:

1. Read the grading command, the value (N) of a correct answer as well as the correct answer.
2. If `GRADING COMMAND = SYMBOLS external` then a list is compiled of all allowable variable symbols that students might use in answering this question. This list contains numerous variations on spelling of the variable names given in the test paper as a safeguard against student typing errors.
3. If `GRADING COMMAND = SYMBOLS internal` then a list of aliasses for the above list of variable names is compiled. Two objectives are attained with this command. Firstly, all variable names are encoded into a format suitable for Matlab's Symbolic Toolbox.

Secondly, all the different variable symbols that can be used by the students are translated into a single list of variables for use by Matlab.

4. If `GRADING COMMAND = TICK` then Matlab searches through the student's answer for an exact match to the string given in `answer`. This grading command is of limited use, since it is extremely sensitive for typing variations. In future versions of the CBA system, this grading command should be improved to be robust for typing and spelling errors.

5. If `GRADING COMMAND = TickNumEq` then Matlab searches through the student's answer for a numerical value close enough to the given value in `answer`. If the correct answer is e.g. 15 μF, then any numeric value in the student's answer that lies within a prespecified tolerance from 15×10^{-6} is regarded as being correct. `TickNumEq` doesn't require that the student use the correct unit, it is only sensitive for the correct value scaled by the SI prefix used for the unit.

6. If `GRADING COMMAND = ANSWER` then Matlab searches for a numeric value close to the given answer. This command is similar to `TickNumEq` with the exception that a few additional requirements have to be met before the student receives credit for his/her correct answer. These additional requirements are specified in `answer`. As an example the following `answer`: $L_1 = 20$ mH implies that the student must use (a) the correct variable (namely L_1) as well as the correct unit (Henry). This grading command is useful in design and analysis questions where the student has to determine specific component values or specific quantities in a circuit.

7. If `GRADING COMMAND = TickEq` then Matlab searches through the student's answer for a mathematical expression that is equivalent to the one given in `answer`, e.g. $\frac{V_1 - 220\angle(23)}{12k}$.

8. Repeat the loop until the end of the memorandum for this question is reached.

At present, the grading subsystem suffers from the limitation that no logical line of reasoning is enforced by the automatic grading software. This limitation can however be addressed relatively easily by means of pointers progressing through both the memorandum and a student's answer. In this manner a student will only be given credit for correct answers if they are in the same sequence as in the memorandum.

Of the six grading commands introduced in section 4.1 only three warrant additional explaination, namely: `TickNumEq`, `ANSWER` and `TickEq`. (The Matlab code of the entire automatic grading subsystem is freely available from the authors.)

4.2 The logic of 'TickNumEq'

The purpose of this grading command is to search through a student's answer for a number which is within a prescribed tolerance from a given number in the memorandum. The following sequence of events occurs when `TickNumEq` is called in the memorandum:

1. The required (complex-valued) number is extracted from the memorandum. This step is actually quite involved since both the memorandum and student answers are encoded in text files. Strings therefore have to be read and converted into numeric values where applicable. Regular expressions can be used to great advantage to perform this step.

2. As soon as the memorandum number is extracted, the program checks whether the number is accompanied with a unit and SI-prefix (e.g. kilo). All units are discarded. If an SI-prefix is detected, then the memorandum number is re-scaled accordingly.

3. Now the attention of the program shifts from the memorandum to the student's answer. All numbers are extracted from the string containing the student answer. For each of these numbers the following steps are performed:

   (a) Search for a unit and SI-prefix. If found, re-scale the number accordingly.

   (b) Subtract the found number from the memorandum number.

   (c) If the difference is within the prescribed tolerance, then the required number of tick marks are placed at the end of the relevant line in the student's answer. The current sub-routine is also summarily ended and the attention of the automatic assessor is diverted to the next line in the memorandum.

4.3 The logic of 'ANSWER'

This command is similar to the command `TickNumEq` with the exception that in addition to a correct numeric value the student also has to supply at least a correct variable and sometimes the correct unit as well. As one could expect, the logic of `ANSWER` is also quite similar to that of `TickNumEq`. Evaluating the command `ANSWER` entails the following procedure:

1. Translate all variable names in the student's answer to valid Matlab variable names. This is done by making use of the lookup table compiled during the evaluation of the commands: `SYMBOLS external` and `SYMBOLS internal` as discussed in section 4.1.

2. Find the specific variable mentioned in the memorandum answer associated with the current grading command.

3. Find the correct numeric value of this variable from the memorandum answer. This numeric value is also rescaled appropriately if an SI-prefix is detected.

4. Find the unit specified (if any) in the memorandum answer.
5. Search for the next numeric value in the student's answer string.
6. Search for any valid variable name immediately prior to the number extracted in the previous step.
7. Search for any valid unit and SI prefix immediately after the number extracted in step 5.
8. Test whether the student's number is sufficiently close to the memorandum number. Also test whether the correct variable name and unit (if any) are present in the student's answer. If both (or all three) of these conditions are met, then the student gets his/her due credit and the subroutine is discontinued. Otherwise, the search carries on until all numbers in the student's answer have been evaluated in this manner.

4.4 The logic of 'TickEq'

This command directs Matlab to search through the student's answer string for a mathematical expression which is equivalent to the one supplied in the memorandum answer.

The basic idea upon which the software of this grading command is based is to convert a student's LaTeX expression into a format acceptable for Matlab's Symbolic Toolbox. Testing for mathematical equivalence in the Symbolic Toolbox environment then becomes a mere formality. The entire procedure can be summarised as follows:

1. Extract the LaTeX equation in the memorandum answer.
2. Convert the LaTeX expression into Symbolic Toolbox format. Regular expressions once again will simplify this process considerably. At present however the conversion from LaTeX to Symbolic Toolbox format is done by means of string operations. As in the case of `ANSWER` (step 1), all variable names both in the memorandum as well as in the student's answer are translated to the correct internal format. The program responsible for converting LaTeX into Symbolic Toolbox expressions is obviously a work in progress. At present it is limited to fractions and algebraic equations.
3. Search through the student's answer for the next candidate equation. A candidate equation is any line containing at least one equals (=) sign.
4. Parse the candidate expression into its main terms.
5. Convert each term into Symbolic Toolbox format and test for mathematical equivalence. Two expressions are equivalent if their difference is zero. This test can be easily performed in the Symbolic Toolbox. If the student's answer is equivalent to the memorandum expression, then credit is allocated and the loop discontinued.

Although the actual program code of the automatic grading subsystem is quite complicated, the underlying principles are simple and intuitive. In its present form the grading software is generic and forms a useful foundation for further development. As the next section shows, the initial results that were obtained during the exams are encouraging.

5. RESULTS

This section gives an example of the feedback that the system can give to students. Thereafter the performance of the automatic grading subsystem is evaluated on the basis of the results obtained by students who wrote one of their mid-year exams in the CBA system, as well as on the basis of qualitative feedback obtained from the same students.

As mentioned earlier, the CBA system was implemented in a second year module on basic electrical circuits. This module is compulsory for all students in the Faculty of Engineering at the North-West University, but is presented separately to electrical and electronic engineering students on one hand (78 students in 2012) and chemical and mechanical engineering students on the other hand (217 students in 2012). The quantitative test results reported in this paper are those obtained by the latter group of students due to its greater size.

5.1 An example of a graded answer

Space limitations and ethical considerations prohibit the publication of an entire graded test paper. An example of a graded answer to a single question is however given in figure 5. This question required the students to calculate a specific voltage in a circuit by means of the well-known node-voltage method. Each red tickmark($\surd$) indicates one mark (as is tradition). All of the marks obtained for a particular line in the student's answer are placed either at the end of that line or immediately below it.

The design and development of the prototype CBA system was guided by the list of criteria given in section 1. How the CBA system measures up to each of these criteria will now be discussed in turn.

5.2 Accuracy

No errors should be made by the automatic grading subsystem. More specifically, correct answers shouldn't be overlooked and neither should incorrect answers receive any credit. Quality in this aspect of the CBA system is ensured by the following three step process: (i) spot checks performed by the lecturer; (ii) internal moderators; and (iii) the students themselves.

An indirect measure of the accuracy of the CBA grading subsystem is the distribution of the marks obtained by the

$$-(6+j13)mA+\frac{V_1}{50}+\frac{V_1-\frac{V_0}{10}}{j25}=0\surd\surd\surd\surd$$
$$I_0=\frac{V_1-\frac{V_0}{10}}{j25}\surd\surd$$
$$V_0=-20I_0(10-j20)$$
$$V_1=-0.2+j0.65V\surd$$
$$V_0=-2+j4V\surd\surd\surd\surd\surd\surd\surd\surd\surd\surd$$
$$I_0=0.01A$$

Figure 5: Example of a graded student answer

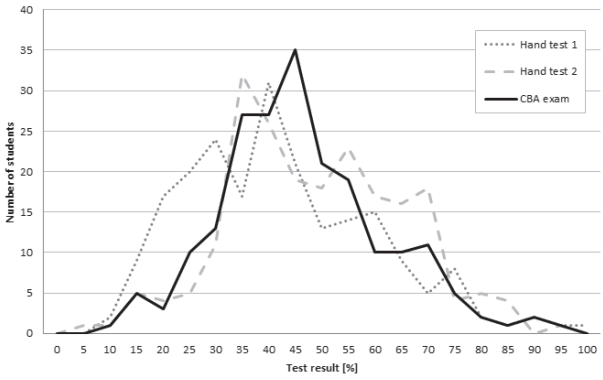

Figure 6: Distribution of results obtained in two handwritten tests and the CBA system

students for the various assessments during the course of the semester. Two semester tests were taken by means of the traditional pen-and-paper method. Experience gained in presenting the module over the past few years has shown that the performance in the semester tests is a good indicator of the performance in the exam. This holds for both the individual student as well as the class overall. If this trend also holds true after the introduction of the CBA system, then it is safe to conclude that the automatic grading system can accurately perform its task.

Figure 6 shows the results of these two handwritten semester tests as well as the first CBA exam paper. These results reflect the performance of the 207 *active* students in the class of 217 Chemical and Mechanical Engineering students. These distributions show that there isn't much difference between the results of the two handwritten semester tests and the CBA exam. Consequently, one can conclude that the introduction of the CBA system didn't have a significant impact on the students' performance. Furthermore, the general shape of the distributions seem to be roughly the same for both the handwitten tests and the CBA tests and is no cause for alarm.

5.3 Consistency

From a computer's perspective each student reduces to nothing more than a file that has to be processed. Consistency is therefore an inherent trait of any CBA system, since it is impossible for an inanimate piece of hardware to exhibit a preference for one student over another.

The truth of the matter is one thing, but it is also important to take the perceptions of the students into account when evaluating the CBA system. With this in mind, an attitudinal survey was conducted amongst the students three months after the first semester exams. Lickert scale questions were used to measure their opinions and experiences on a wide variety of aspects of the CBA system. Only four response categories were used in the Lickert scale, namely: strongly disagree, disagree, agree and strongly agree. Unfortunately only 44 of the grand total of 294 second year students participated in the survey. This does cast some doubts on how representative the results of this survey are of the true general feeling amongst the students.

Two questions were posed in the survey to specifically measure the students' perception on the fairness of the automatic grading subsystem. These questions were:

Q1 Do you feel that the computer is too rigid? In other words, would you prefer it if a person can read your answer and judge its merit?

Q2 Do you feel that the computer was truly consistent in terms of treating all students exactly the same?

Figure 7 summarizes the results obtained by these two questions. It is quite clear that all of the respondents felt that the grading system is too rigid. Unfortunately their preference for a human assessor that can judge their answers on merit unavoidably introduces subjectivity into the grading process. Whereas it is undoubtedly true that a human will often have mercy on a student due to his/her situation (which constitute external factors to the test), this does have a detrimental effect on the quality of the assessment.

With respect to the second question (namely whether the CBA system treated all students equally), the results in figure 7 indicate that the majority of respondents felt that the system did indeed treat all students exactly the same. Why 43 % of the students didn't agree with this statement is puzzling and should be investigated more deeply. All of the above mentioned results should however be interpreted by taking into consideration that the CBA system was introduced during the last two weeks of the semester. The students therefore only had three weeks to get accustomed to the CBA system before writing the exam.

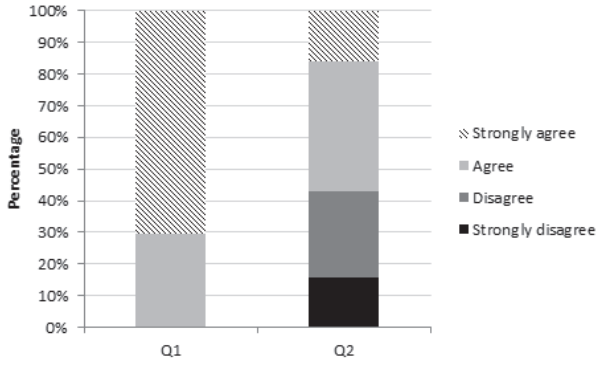

Figure 7: Student survey results concerning the fairness of the grading subsystem

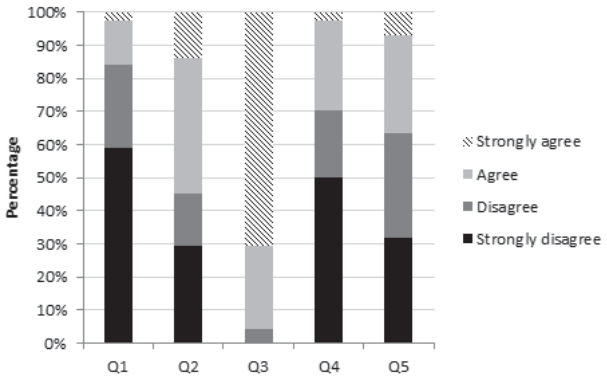

Figure 8: Student survey results concerning the usability of the CBA system

5.4 User friendliness

Any CBA system has two primary users, namely the lecturer and the students. The benefits of a CBA system for the lecturer in terms of grading test papers are obvious. Although test papers have to be converted in the correct format to be displayed in the test-delivery subsystem, this is indeed a small price to pay in return for having the grading performed accurately, consistently, quickly and above all automatically.

To a large extent the students' experience on the user friendliness of the entire CBA system is determined by the test-delivery subsystem.

In terms of the user friendliness of the test-delivery subsystem, the following five questions were posed in the attitudinal survey:

Q1 Although the CBA test-delivery system was introduced only three weeks prior to the date on which I wrote the final exam, I was relatively confident with the CBA test delivery by the time that I had to write the exam.

Q2 Students in general will be able to use the test delivery system with confidence given enough time to get to know the system.

Q3 I prefer to write traditional pen-and-paper tests.

Q4 I will not mind writing tests or exams on CBA systems.

Q5 I found the CBA test delivery system user friendly.

The results obtained from these five questions are summarized in figure 8. The response to question 1 clearly indicate that the students felt that they had too little exposure to the system before they had to write their exam. The fundamental user friendliness of the system is however revealed in the response to question 2. A small majority of the class were of the opinion that they would be quite confident in the usage of the system, given sufficient exposure.

The typical resistance to paradigm shifts is quite evident in the response to question 3 (namely that the vast majority of the class prefers pen-and-paper tests). A significant minority of the class (29.5 % to be exact) however wouldn't mind to write tests or exams on the CBA system in the future. This is an important result, since it shows that the system isn't fundamentally flawed, only introduced unwisely. Although the fifth question clearly shows that the majority of the respondents felt that the test-delivery subsystem was unfriendly, this opinion should be balanced by slightly more than a third of the respondents who experienced the system as user friendly.

In balance, the students felt that the CBA system is not user friendly; a feeling that can only partly be attributed to the late introduction of the system. Any prototype has significant room for improvement. Specific aspects of the test-delivery subsystem that need attention in future were identified from three open-ended type questions in the survey. These questions asked the students respectively what they regarded as the disadvantages and advantages of the system, as well as any suggestions for future improvement.

A large contingent of the respondents remarked that they had to perform draft calculations on paper before typing in the important parts of their answers. This two-stage process inconvenienced them, wasted a lot of time and was a cause of uncertainty (since they had to choose which parts of their calculations to type in and which not to). The last drawback mentioned by the students is very interesting, since it highlights the fact that the CBA system forces students to put extra thought into how they communicate their answers. One could therefore postulate that the CBA system has the added benefit of fostering logical and systematic thought in students. Support for the latter hypothesis can be found in the following interesting statement from one of the students in the survey: "*The CBA system did not make it any more difficult to pass the module in the sense that the format is exactly the same as when the equations were written on paper. Instead it helped me to avoid writing down unnecessary mathematics and calculations.*"

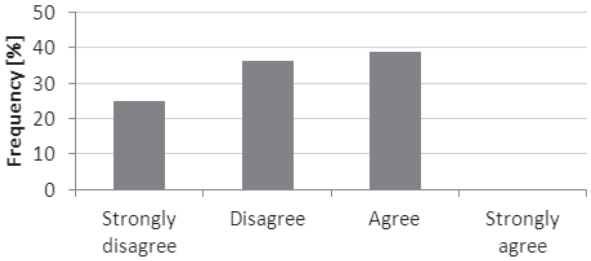

Figure 9: Student survey results concerning the quality of feedback in the graded papers

Surprisingly, a number of students also mentioned that typing *per se* was a problem for them. Being competent with a keyboard is however essential for professional engineers. It is therefore important for students to get even more exposure to electronic communication technology (even during tests and exams).

Lastly, a number of students implicitly recommended that the equation editor be embedded in the web-browser. This will undoubtedly improve the user friendliness of the system and ultimately increase the levels of user acceptance.

To measure the quality of feedback given by the automatic grading system, the following question was posed in the attitudinal survey: ”Are you of the opinion that the graded answer papers delivered by the computer (the pdf report with red tick marks in it) gives informative feedback so that you can learn from your mistakes?”.

The feelings of the students on this matter are summarized in figure 9. It is disappointing that only 39 % of the students regarded the graded answer papers as being useful feedback. The viewpoint of the majority of the students can probably be attributed to the fact that tickmarks are only placed at the end of a line, rather than in between the text as in a traditional pen-and-paper test.

5.5 Speed

The time required by an average student to complete a CBA test obviously depends on the student's proficiency in the system. Seasoned LaTeX users can produce mathematical expressions with minimal delay, while novices will take a long while to type in an equation by means of a mouse.

Our practical experience with the system has been that it takes an average student approximately two minutes for each mark in a test. A typical three hour exam can therefore only have a maximum of 90 to 100 marks. This compares favourably to the three hours and 120 marks allocated for previous handwritten exams in the module. One can therefore conclude that the CBA system only has a small impact on the time required by students to answer tests.

This conclusion is supported by the results of the student

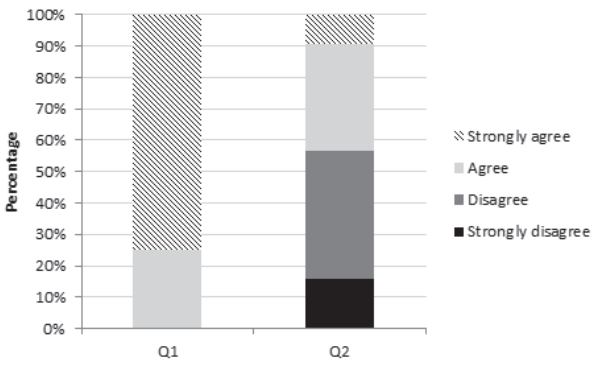

Figure 10: Student survey results concerning the speed of the CBA system

survey. Figure 10 reflect the students' experience of how long it took to complete a CBA test. The results of two questions are combined in figure 10, namely:

Q1 I found that I took longer to answer questions in the CBA system than in handwritten tests and exams.

Q2 I could answer all of the questions that I knew the answers of in the time provided to do the exam on the CBA system.

Clearly, all of the students felt that it took longer to complete a CBA test, than a traditional pen-and-paper test. This result should however be seen in the perspective of the results of the second question, from which one can conclude that only a small majority of the class felt that they didn't have enough time to communicate the answers that they knew in the CBA system.

The issue of the time required to answer a CBA test can therefore be easily addressed by the following measures. Firstly, students should be given exposure to the system from as early in a semester as possible. Furthermore the contents and length of a CBA test should be adjusted to fit within the constraint of three hours.

The savings incurred during the automatic grading of answer papers is obviously enormous. The mere fact that this part of the work is performed automatically already implies that the lecturer can continue with research during office hours and leave the grading of tests over to the computer while he/she is otherwise occupied. Still, the computer does perform the task of grading answer papers significantly faster than humans can. More precisely, it takes on average 60.5 seconds to automatically grade an entire exam paper of a student. To place these numbers in perspective, it took approximately 23 minutes in previous years to mark a single exam answer paper by hand. The relatively small amount of additional effort required from the lecturer to compile a machine-readable memorandum, is therefore handsomely compensated for by the subsequent automatic grading process.

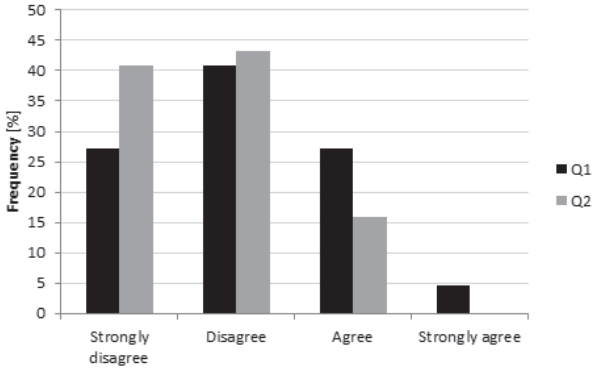

Figure 11: Student survey results concerning the ability of the system to assign partial credit

5.6 Partial credit

As we've seen in the previous section, the automatic grading subsystem can grade a full length exam paper in approximately one minute. Whereas this speed is impressive compared to humans, it is still far from the near-instantaneous feedback offered by multiple-choice tests. The difference lies in the detailed feedback and partial credit offered by the prototype CBA system.

It should be clear from the example given in this paper (figure 5) that the system does indeed give students partial credit for what they did correctly in a question. The grading commands comprising the core of the automatic grading subsystem (see sections 4.1 to 4.4) allow lecturers numerous creative ways to allocate marks in a test. This makes it possible to test more than one aspect of a module in a single question and yet give the student credit for each part that he/she could answer correctly.

It is interesting to observe that once again the students didn't share the above mentioned viewpoint in the attitudinal survey. Two questions on the theme of obtaining partial credit were included in the survey, namely:

Q1 Do you think that the computer system can give credit for more than one possible approach to answer a question?

Q2 Do you feel that the computer gave you credit for following the correct process in certain questions even though your final answers were sometimes incorrect?

The histograms in figure 11 summarize the student's feelings on the topic of the ability of the grading system to assign partial credit. Clearly, the majority of the respondents are of the opinion that the system can't assign partial credit to their answers. This majority does however correlate quite well with the general negative disposition of the class towards the CBA system (due to its late and forced introduction in a module that is highly unpopular amongst the majority of the students in the faculty).

6. CONCLUSIONS AND RECOMMENDATIONS

The prototype CBA system was primarily developed with an introductory course on electrical circuits in mind. Even though it is more suitable for senior undergraduate engineering subjects than other CBA systems currently available, it is still quite limited in its capabilities. These limitations should however be seen as fertile soil for further research and development, rather than critical flaws.

- Grading is performed offline once the test has been completed by all of the students. User acceptance of the system will obviously be boosted if grading could be implemented online as well.
- No automatic correction of spelling and typing errors is performed. The grading software is therefore very sensitive for spelling errors by the students. This however didn't pose much of a problem in the modules in which this system has been implemented, due to the mathematical nature of both modules.
- Graphical input can't be graded at present. Only numerical answers and mathematical expressions can at present be reliably graded.

The current version of the system is limited to modules with a significant mathematical content. Work is however underway to extend its abilities to automatically grade textual answers (e.g. paragraph-style questions) in other Engineering modules.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the help and advice given by Gustav Otto, Celeste du Preez, Steyn Geldenhuys as well as the anonymous reviewers who improved the quality of our paper.

REFERENCES

[1] G. Crisp, *The e-Assessment handbook*. Continuum, 2007.

[2] S. Mehta and N. Schlecht, "Computerized assessment technique for large classes," *Journal of Engineering Education*, vol. 87, pp. 167–172, April 1998.

[3] R. Henderson, D. Larimore, D. Lowhorn, V. Mayfield, and L. Hayes, "Testing and improving educational software," *Computers in Education Journal*, vol. 16, pp. 2–15, 2006.

[4] S. Hussmann, G. Covic, and N. Patel, "Effective teaching and learning in engineering education using a novel web-based tutorial and assessment tool for advanced electronics," *International Journal of Engineering Education*, vol. 20, pp. 161–169, 2004.

[5] C. Smaill, "The implemention and evaluation of oasis: A web-based learning and assessment tool for large classes," *IEEE Transactions on Education*, vol. 48, pp. 658–663, 2005.

[6] L. Fawcett, B. Foster, and A. Youd, "Using computer based assessments in a large statistics service course," *MSOR Connections*, vol. 8, no. 3, August - October 2008.

[7] Questionmark, "Questionmark perception: measure knowledge, skills and attitudes securely for certification, regulatory compliance and successful learning outcomes," 2012. [Online]. Available: https://www.questionmark.com/us/perception/Pages/default.aspx

[8] K. Scalise and B. Gifford, "Computer-based assessment in e-learning: A framework for constructing intermediate constraint questions and tasks for technology platforms," *The Journal of Technology, Learning, and Assessment*, vol. 6, no. 6, pp. 4–44, June 2006.

[9] N. Nirmalakhandan, "Use of computerized dynamic assessment to improve student achievement: Case study," *Journal of Professional Issues in Engineering Education and Practice*, vol. 135, pp. 75–80, 2009.

[10] V. Crisp and C. Ward, "The development of a formative scenario-based computer assisted assessment tool in psychology for teachers: The pepcaa project," *Computers and Education*, vol. 50, pp. 1509 – 1526, 2008.

[11] "AiM assessment in mathematics," 2012. [Online]. Available: http://maths.york.ac.uk/yorkmoodle/course/view.php?id=67

[12] "CABLE: computer algebra based learning and evaluation," 2012. [Online]. Available: http://www.cable.bham.ac.uk

[13] C. Sangwin and T. Hunt, "Philosophy of STACK," November 2012. [Online]. Available: https://github.com/maths/moodle-qtype_stack/blob/master/doc/en/About/The_philosophy_of_STACK.md

[14] N. Kalogeropoulos, I. Tzigounakis, E. Pavlatou, and A. Boudouvis, "Computer-based assessment of student performance in programing courses," *Computer Applications in Engineering Education*, 2011.

[15] E. Gutierrez, M. Trenas, J. Ramos, F. Corbera, and S. Romero, "A new moodle module supporting automatic verification of vhdl-based assignments," *Computers & Education*, vol. 54, pp. 562–577, 2010.

[16] S. Almajali, "Computer-based tool for assessing advanced computer programming skills," in *2012 International Conference on E-Learning and E-Technologies in Education, ICEEE 2012*, Lodz, 2012, pp. 114–118.

[17] S. Long, R. Dowsing, and P. Craven, "Knowledge-based systems for marking professional it skills examinations," *Knowledge-based Systems*, vol. 16, pp. 287–294, 2003.

[18] R. Roselli, L. Howard, and S. Brophy, "A computer-based free body diagram assistant," *Computer Applications in Engineering Education*, vol. 14, pp. 281–290, 2006.

[19] I. Achumba, D. Azzi, V. Dunn, and G. Chukwudebe, "Intelligent performance assessment of students' laboratory work in a virtual electronic laboratory environment," *IEEE Transactions on Learning Technologies*, vol. 6, no. 2, pp. 103–116, 2013.

[20] C. Huang, Y. Wang, T. Huang, Y. Chen, H. Chen, and S. Chang, "Performance evaluation of an online argumentation learning assistance agent," *Computers & Education*, vol. 57, pp. 1270–1280, 2011.

[21] R. Siddiqi, C. Harrison, and R. Siddiqi, "Improving teaching and learning through automated short-answer marking," *IEEE Transactions on Learning Technologies*, vol. 3, pp. 237–249, 2010.

[22] R. Hoeft, F. Jentsch, M. Harper, A. Evans III, C. Bowers, and E. Salas, "TPL-KATS - concept map: A computerized knowledge assessment tool," *Computers in Human Behavior*, vol. 19, pp. 653–657, 2003.

[23] D. Burkolter, B. Meyer, A. Kluge, and J. Sauer, "Assessment of structural knowledge as a training outcome in process control environments," *Human Factors*, vol. 52, pp. 119–138, 2010.

[24] P. Piwek, "Supporting computing and technology distance learning students with developing argumentation skills," in *2013 IEEE Global Engineering Education Conference (EDUCON)*, IEEE. Berlin: IEEE, March 13-15 2013, pp. 258–267.

[25] "Openmark examples," 2013. [Online]. Available: http://www.open.ac.uk/openmarkexamples/

[26] "Moodle," 2013. [Online]. Available: https://moodle.org/

[27] H.-C. Yang and T. K. Shih, "Using ajax to build an online assessment management system based on qti and web 2.0," *WSEAS Transactions on Information Science and Applications*, vol. 4, no. 5, pp. 939 – 945, 2007.

[28] M. Niculescu, "ExualX LaTeX equation editor," 2010. [Online]. Available: http://equalx.sourceforge.net/

[29] C. Ko and C. Cheng, "Flexible and secure computer-based assessment using a single zip disk," *Computers & Education*, vol. 50, pp. 915–926, 2008.