Maths-CAA Series
September 2004

Pseudo-Randomised CAA by "Preprocessing" MathML

Contents

Pseudo-Randomised CAA by "Preprocessing" MathML

Peter James Rowlett
School of Computing & Informatics, The Nottingham Trent University
Email: peter.rowlett@ntu.ac.uk

Abstract: The development of the Mathematical Markup Language (MathML) has enabled mathematics to be expressed as open, machine-readable code on the Web. By utilising technologies for creating dynamic, database-driven websites, it is possible to "preprocess" mathematics, creating a Web-based test in which each question contains pseudo-randomised elements and where simple questions can combine to form more complicated ones. This means a small bank of relatively simple questions and distractors can be used to create large numbers of more complicated questions for students to answer. The generation of large numbers of questions in an accessible format can be seen to be an advantage when meeting the requirements of SENDA [1]. Questions on basic differentiation were given to undergraduate students of Engineering so that an evaluation of the test system could be performed. The procedure of accessing machine-readable mathematics code unlocks huge potential for the use of current Web technologies in mathematics CAA.

1. Introduction

Mathematics is a "hands-on" subject; the techniques must be practised in order to fully understand them. Consequently, students can benefit greatly from a wide variety of optional "self-test" materials. It has been recognised that computer-based assessment (CAA) can provide the opportunity for students to "try their hand" without the excessive drain on staff time taken up by marking hand written scripts. In addition, pseudo-randomisation can be used to vary the student's experience of the test, in order that it might be taken several times by the same student, if required.

Extra issues are raised by recent legislation. Under SENDA [1], institutions should be anticipating the needs of students with disabilities (even if no such students are known to be enrolled). A student with a learning difficulty might well need great quantities of practice material to understand a topic, particularly one involving the application of processes or algorithms. Students using assistive technologies, for example those with visual impairments, may not be able to interact with CAA systems that make use of non-accessible formats such as those using images for mathematics.

So, large amounts of new practice material might need to be created. With a lot of undergraduate mathematics, a module might only contain a limited number of questions that can be asked. If past examination papers are studied, the pattern of certain "recycled" questions can often be noticed. Large numbers of very similar questions could have to be written, and you might be left wondering if this process cannot be automated in some way.

This paper describes an attempt to create a system of mathematics assessment that allows pseudo-randomised constants to be introduced into expressions and allows the combination of mathematical expressions. This way only a small set of questions needs to be written to produce large numbers of questions for students.

2. Background

2.1 MathML

The Mathematical Markup Language (MathML) is described in some detail in Rowlett [2]. An eXtensible Markup Language (XML) implementation created by the World Wide Web Consortium (W3C) [3], this open format for describing mathematics has the potential to be very accessible and useful to the mathematics community. Indeed, there already exists a MathML plug-in for the Internet Explorer browser, MathPlayer [4], which allows mathematical expressions to be spoken aloud. Plans for further accessibility-related developments to MathPlayer exist [5].

MathML uses tags (like HTML) to describe mathematical items. To give the reader some conception of MathML code, an example is given in Figure 1.

$$\frac{\sin(x)}{x+2}$$

```
<math>
 <apply>
  <divide/>
  <apply>
   <sin/>
   <ci>x</ci>
  </apply>
  <apply>
   <plus/>
   <ci>x</ci>
   <cn>2</cn>
  </apply>
 </apply>
</math>
```

Figure 1: An example of MathML code (right) for the mathematical expression given (left).

For those familiar with MathML: Content MathML was used, the semantic richness having advantages over Presentation MathML when the code is to be processed automatically. Students were recommended to use Internet Explorer with the MathPlayer plug-in, though Mozilla would also render the test using XSLT.

2.2 PHP

PHP is a recursive algorithm for "PHP: Hypertext Preprocessor." It is a language used primarily to "preprocess" HTML code. When you type a web address, you are requesting that a web server send you a webpage. Often, this is just a static HTML file which has been hand written. However, the request could cause the server to execute a piece of code, written in PHP for example, which generates an HTML file dynamically. This is known as preprocessing HTML code.

The advantage of this is that a webpage can be customised for you personally, perhaps by interaction with a database. This is used, for example, to generate a "shopping trolley" page for an online shop or index pages for a database like an image library.

As well as outputting HTML, PHP can be used to "mimic" other file types, for example XML (and thus, MathML) files. This is done using the PHP header() function to output an appropriate Multipurpose Internet Mail Extensions (MIME) type.

Using a server-side language like PHP allows access to the power of dynamically generated pages without the accessibility hang-ups of using a client-side technology such as Java or JavaScript.

3. The Idea: Prerendered Versus Preprocessed Mathematics

Mathematics on webpages is commonly in a prerendered format. This means that it has been written and saved in the format in which it will be displayed, as images, ASCII text, Java applets, PDF and occasionally even MathML. This prerendered mathematics is perfectly acceptable for many purposes, but advantages can be gained by using a different approach here. In the same way that HTML code can be preprocessed for dynamic websites, such as e-commerce sites or image libraries, MathML code can be preprocessed. This preprocessed mathematics can be dynamically altered for each user. Rather than having a database of explicitly defined questions, snippets of mathematical expressions can be stored and combined on the fly, thus, introducing pseudo-randomised elements for each student.

The chain rule, say, allows application of one function to another with a formula for calculating the result. Why then must solutions be written for differentiating, say, $\cos(x^2)$, $\exp(x^4)$, $\cos(x^3+2)$, $\exp(x^2+5)$, ...? All that seem to be needed are combinations of the solutions to differentiating the following: x^n, n, $\cos(t)$ and $\exp(t)$. Everything else is duplication.

MathML allows real access to a computer-useable code for mathematics. By using techniques developed for creating dynamic, database-driven websites, expressions can be created with pseudo-randomised constants and be combined together to avoid the duplication mentioned above. More specifically, using a server side scripting language such as PHP, pseudo-randomised constants can be introduced into MathML expressions. Interfacing this with a database, snippets of MathML code can be pulled together into more complicated expressions.

4. Implementation

4.1 Assessment tool

Learning styles differ; some people learn effectively by repetition. In mathematics in particular, it can be useful to apply a procedure or algorithm until confidence and understanding are achieved.

Assessment is an important component of a mathematics course. Whether contributing to a final mark or simply for individual practice, the chance to perform mathematics is regarded as essential to learning the subject. With large numbers of students requiring such practice, however, overheads such as staff time for writing and marking of problems create barriers to this happening. Pseudo-randomisation features enable a test to be taken by many students simultaneously, or by the same student several times, if required. Pseudo-randomisation has been used in CAA, and it is well recognised that it can increase the worth of a CAA package. The development of MathML has enabled mathematics to be manipulated by computer programs in a way that is simply not feasible with bitmapped mathematics or other previous formats for mathematics display.

4.2 Mathematics input/output

Strickland [6] discusses problems inherent with the input of mathematics to computers that can distract from the mathematics involved. A student who misunderstands a complicated input mechanism might be unable to express their answer correctly. In this case, the student's mathematical ability is not

being tested fairly.

Strickland also discusses the difficulties involved in having a computer program decide if an answer is correct. Since mathematics is such a flexible language, the answer to a problem might be expressed in a large number of equivalent forms. Such a computerised marking system would have to be able to perform the algebraic manipulation necessary to test this equivalence, which is far from trivial.

Lawson [7] states that reservations exist about the use of multiple choice testing, since a student with no knowledge or understanding can potentially guess the answer and thus get the question correct. However, Lawson outlines the use of negative marking to combat the possibility of students guessing the correct answers. When presented with four possible answers (or the option to abstain) the student will receive three marks for a correct answer and lose one mark for an incorrect answer (with no penalty for abstention). Of course, the idea is that by guessing a student ought to get approximately one quarter of the answers correct, and thus obtain a score close to zero.

Lawson observes that in a multiple choice test the author is forced into giving away the format of the answer, potentially giving a student with no idea of how to approach a question an indication of the method required. In this case it should be kept in mind that the multiple choice test will examine whether the student can apply a particular method or technique, and not necessarily test whether they are aware of when they should do so.

Despite the above problems, Lawson concludes that the advantages of giving a student immediate feedback and guidance as to any mistakes made far outweigh the difficulties inherent in computer-aided assessment.

4.3 Questions used

Multiple-choice questions were used; the technical issues surrounding computer marking of free entry mathematics were beyond the scope of this work.

Each page presented to the student contained one question with a correct answer and three distractors. The scoring used negative marking.

Importance was given to limiting the difficulty of the questions. With pseudo-randomisation it would be easy to generate excessively difficult questions; strict guidelines were enforced to prevent the questions becoming too difficult (or irrelevant).

In addition to selecting questions that are of a reasonable difficulty and relevance, when setting multiple-choice questions care must be taken over the distractors. Each distractor in the set must appear feasible. It should not be too obvious that a possible answer is incorrect, or the student will be untested by the distractor. Of course this depends on the student's expected ability.

The area of mathematics covered here is basic differentiation. This topic involves algorithmic solving of problems, and is an area in which learning is greatly facilitated by repetition. Regardless, the choice of topic is somewhat arbitrary and the techniques developed here could be applied to any mathematics that can be encoded in MathML.

During this development a cohort of around 30 undergraduate students of Engineering, at the Nottingham Trent University, covered basic differentiation topics, and was asked to take a test for evaluation purposes.

Rules were encoded to differentiate the functions listed in Figure 2. These could then be combined in a large number of ways, with pseudo-randomised constants used for added variety.

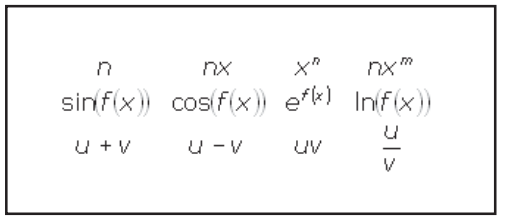

Figure 2: Mathematics used in the test system – the rules for differentiating these expressions were encoded

Having encoded the basic rules with markers, for when they might include pseudo-randomised constants or other functions, it became possible to write code to pseudo-randomise combinations of these simply. This created a wide variety of possible questions, however, not all of these were suitable for use. It became necessary to include guidelines for how the questions might combine, and this led to the outlines for eight satisfactory questions being written. These are listed in Figure 3. Obviously these are not the only reasonable questions that could be written using combinations of the rules in Figure 2; eight were chosen as the students were accustomed to taking paper-based self-assessment tests of around this length.

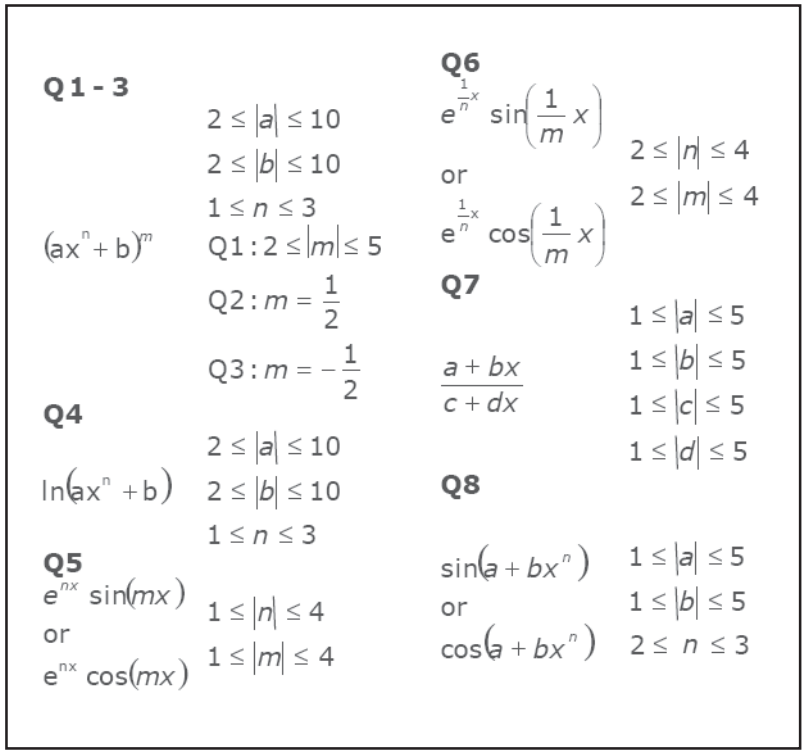

Figure 3: Specifications for the eight questions used in the test

As the functions in Figure 2 build up to form the questions in Figure 3, so the distractors associated with the former build distractors for the latter. Each of the function definitions shown in Figure 2 has encoded with it a number of distractors (often greater than 3). A distractor for one of the compound rules is formed by choosing a distractor from either the rule itself or one of its composite functions.

Indeed, a question on one of the rules will attempt to include one distractor involving an error in the rule, and one for each composite function. If this does not generate three distractors (e.g. if there is only one composite function, or two distractors simplify to the same value) then further distractors are chosen pseudo-randomly from the distractors of the component parts. So, if a question on the product rule combines the functions u and v, one distractor will involve an error in the application of the product rule, one an error in differentiating u and one, similarly, for v. If this fails to create three distinct distractors then

further distractors will be chosen from the rule, u or v. It was felt that producing more than one compound error per distractor would lead to unfeasible distractors.

4.4 Mathematical simplification

Once the differentiation rules have combined to form questions, there is no guarantee that they will provide presentable mathematics. Additionally, two or more distractors may combine to give the same value. A situation could even be imagined where a distractor can simplify to give the same value as the correct answer. Clearly, the system needs to be capable of some mathematical simplification. The procedure decided upon was to store snippets of MathML code containing mathematical expressions needing simplification, along with their simplified form. Armed with such information, the system could search an expression for the former and replace it with the latter.

In practice, this system is of limited usefulness. The time taken to parse the relevant files means that a test of much greater complexity than the one produced would take prohibitively long to load each question. Interaction with a server-side Computer Algebra System (CAS) might be useful in this case.

4.5 Test delivery

Care was taken to ensure that the students would not be confused by the test delivery method. Notes were placed at strategic points to encourage the students to use the test correctly, and an intuitive answer-selection method was an aim. In particular, a note was included to declare the use of negative marking; as this is included as a deterrent to reduce guessing, it was felt this should be clearly stated to the student.

In addition, it was decided that some students might benefit from being able to answer individual questions and gain immediate feedback. Indeed, some students might wish to limit themselves to a particular area on which to answer questions. To this end, an “individual questions” version of the test was included in addition to the full 8-question version of the test.

Both versions of the test have similar question display formats. Accessibility and usability were considered when developing the test interface. The format of the pages on which the questions appear was kept deliberately simple. It was believed an overcomplicated visual display would distract the student and possibly confuse the process. The test interface is shown in Figure 4.

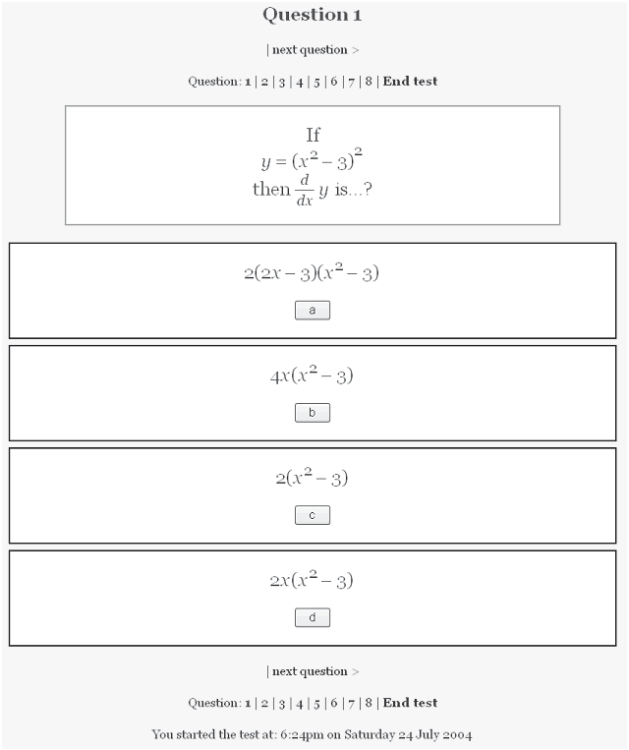

Figure 4: Full test interface

On completion of the full test, a "Feedback Report" is generated. This contains the number of correct and incorrect answers, the score obtained, the time taken to complete the test, and a question-by-question (limited) feedback. A sample feedback report is offered in Figure 5.

The question-by-question feedback simply states the question, the answer the student gave (if an answer was given) and the correct answer (if different). It is hoped that this, along with consultation with the student, ought to be sufficient to provide useful support. An advice line at the bottom of the page suggests that the student should print out the report and take it when seeking help, directing students to the University's Maths Support Centre for assistance. In addition, a note refers students who are having problems with just one type of question to the individual questions version of the test.

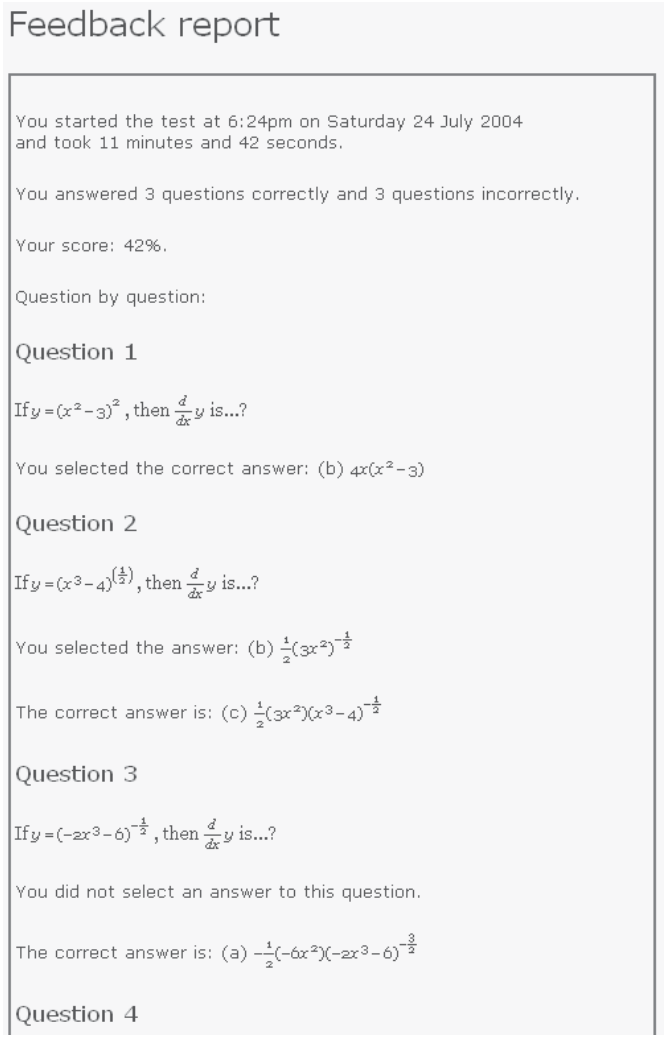

Figure 5: Sample feedback report

The individual questions version of the test provides instant feedback on each question. This is similar to the question-by-question feedback in the full version of the test, in that it contains the statement of the question, the given and correct answers (as needed).

5. Data for Evaluation

5.1 Student cohort

The decision was taken to limit testing for evaluation purposes to a relatively small student cohort (around 30 students, on an optional basis). A relatively small use of the test system would produce sufficient MathML code for an evaluation of the validity of the code, whilst also providing a limited amount of feedback from students as to the usefulness of the test system.

5.2 Data recorded

Students were not asked to login to the test system, and no usernames were recorded as this was not deemed to be a necessary feature. The advantages of being able to determine how many times particular students accessed the test system were believed to be out-weighed by the disadvantages of logging students in. These disadvantages include the technical requirements of validating usernames, or the problems with inaccuracies in the data if not validated. Of course this decision limits the evaluation slightly.

It was deemed necessary to the evaluation of the integrity of the system to keep a record of all MathML code generated. Thus, all questions and possible answers were stored in a database. In addition, the student responses to each question were recorded, to determine whether the questions were of an acceptable level for the student cohort. The date and time of each test was also recorded, to give an idea of usage.

5.3 Questionnaire

A feedback questionnaire was produced and presented to students by the teaching staff. A concern that any students who could not/did not access the test might have difficulty responding to a questionnaire delivered online with the test meant that a physical, written questionnaire was believed to be more suitable. A copy of the questionnaire is given in Figure 6. The aim of the questionnaire was to discern whether students found the test useful, and to provide them the opportunity to comment on it. The questionnaire was kept short, however, out of concern that a longer questionnaire would encourage fewer responses.

Diagnostic e-test.
By filling in this brief questionnaire you help to improve the process.
Only question 1 needs to be filled in if you did not try the test.

1. Did you try the test?
   a. **Yes**…
   b. **No**…

   If **No.** Any particular reason? (e.g. forgot, no time, couldn't log on, didn't think needed to etc.)……

2. **Yes**, how many times?
   a. **Once**…
   b. **Few**…
   c. **Many**…
3. Useful
   a. **Yes**…
   b. **No**…
4. Comments, e.g. Other topics which might be usefully covered in this kind of test
   ……

Figure 6: Student feedback questionnaire

5.4 Number of questions generated

Without pseudo-randomisation of numerical values in the expressions, the system would have been able to produce 11 distinct questions (see Figure 2). With the distractors created (selecting pseudo-randomly three for each question), these 11 questions would potentially be able to have associated with them 7,397 distractor sets (i.e. 11 questions are possible with 7,397 different sets of possible answers).

With the pseudo-randomisation enabled, these 11 written questions generate 21,292 distinct questions. Including distractors, the system is potentially capable of generating 9,479,180 question and distractor sets. These numbers are upper limits, as some questions and distractors may be equivalent.

Of course, this analysis does not take note of the fact that the 11 questions and their distractors were not written but in fact were combined from simpler questions. This means they required yet less work in terms of writing mathematics, and that many more than 11 initial questions could be combined by the system if required.

6. Evaluation

The test was loaded 16 times. Of these attempts, the test was actually completed (i.e. the "End Test" button clicked) only five times. The remaining 11 times the test was loaded, only the first question was viewed.

Of the five times the test was completed, there were two occasions when a student sat down and worked through the whole of the full version of the test. In the first instance around 5 minutes was taken to answer all questions, in the second nearly an hour was taken. Both times a score of around 70% was achieved. The other three times, only two or three questions were loaded before the test was submitted for marking.

A number of other times students attempted individual questions. The timings indicate perhaps 8 separate instances of individual questions being loaded by students. There was a much lower non-completion rate here than with the full test. Perhaps the instant feedback, the length of the test and choice of question type are factors here.

This limited test usage produced nearly 9,000 lines of MathML code. Despite the verbose nature of MathML, this comprised over 250 mathematical expressions. Questions appear to have been reasonable, apart from one case where a student identified an error with the mathematics of one question. This error was not an inherent problem with the system. Rather, a simplification had been encoded incorrectly. It is believed that all code generated was valid MathML and that, apart from the case above, the mathematics generated was valid and sensible.

Five responses were received from the feedback questionnaire. Three of the respondents did not use the test. Two of these cited "Couldn't log on" as the reason for this. This could be a difficulty accessing the webpage, or possibly a problem with the MathPlayer plug-in. Perhaps the question should have been worded differently.

Two respondents to the questionnaire claimed to have used the test. Both respondents reported that they had found the test useful. One of these, having used the test a "few" times, found it "a good way to practice." Both suggested the development of similar tests on integration topics.

The relatively large number of students who loaded the first question and then left the test is of some concern. It might indicate students being "scared away" by a question perceived to be difficult. Perhaps a "simple" introductory question, or varying difficulty levels might have solved this. It should be noted that a student who loaded the test without the MathPlayer plug-in would get what might be perceived as an error. They might then close the test. In this case, the test system would record that they loaded a single question and gave up. Two students in the questionnaire reported difficulties "logging on" to the test. Perhaps the lack of widespread MathPlayer installation is an issue.

7. Conclusion

The aim of this work was to investigate the dynamic production of MathML code and its application to CAA in mathematics. A system was produced that would take a small number of relatively simple mathematics questions and produce a variety of more complex questions. This system was capable of automatically generating vast quantities of useable, valid mathematics. This should reduce concerns that this method of preprocessing mathematics might 'run rampant' and produce invalid mathematics.

In time, encouraged by this validation, a more sophisticated test system (with variable difficulty, for example) might be produced, more students could be engaged and more testing conducted. Nevertheless, the initial positive feedback is encouraging.

A small number of written questions, adapted with the methods created here, would yield a comparatively large number of valid questions for students to answer. It seems reasonable to suggest that, using the test system created here, students will have mastered the required techniques before they suffer from having to answer the same questions repeatedly.

Therefore, this report concludes that utilising the power of preprocessed mathematics for CAA has some merit. It is hoped this will demonstrate the huge potential of current Web technologies when access is given to machine-readable mathematics code. The author hopes to conduct more work in this area in the future, including improvements to and further evaluation of this test system.

Acknowledgements

The work described in this article was completed as part of an MSc dissertation at the Nottingham Trent University, supervised by Tony Sackfield.

References

[1] Special Educational Needs and Disability Act (SENDA). 2001. Stationery Office Ltd.
[2] Rowlett, P. 2003. MathML: the current state of play. Available online at: http://mathstore.ac.uk/mathml/rowlett.html.
[3] World Wide Web Consortium (W3C): http://www.w3.org/.
[4] MathPlayer: http://www.dessci.com/en/products/mathplayer/.
[5] Rowlett, P. 2003. Have you seen this..? / DDA Update: MathPlayer and the Design Science Mathematics Accessibility Project. *MSOR Connections*, **4**(2), 5, LTSN Maths, Stats and OR Network, Birmingham.
[6] Strickland, P. 2001. How should a Perfect Computer Aided Assessment Package in Mathematics Behave? Available online at: http://mathstore.ac.uk/articles/maths-caa-series/aug2001/index.shtml.
[7] Lawson, D. 2001. Computer Assisted Assessment (CAA) in relation to Learning Outcomes. Available online at: http://mathstore.ac.uk/articles/maths-caa-series/oct2001/index.shtml.