

MESS: the MathDox Exercise System

Hans Cuypers, Jan Willem Knopper, Hans Sterk

Abstract. Within MathDox [4, 5], an open source system for presenting highly interactive mathematical documents over the world wide web, we have developed an exercise system. This exercise system consists of an XML-format for interactive mathematical exercises, a player of these exercises and tools for using the exercises inside Learning Management Systems. We discuss the features of our system by way of an extended example.

Keywords. adaptivity, interactive mathematical documents, Math assistants, tutoring and assessment systems.

1. Introduction

In the area of Mathematics we see that quite recently various new tools have been created which offer students a rich environment for practicing mathematics over the world wide web. These systems offer parametrized exercises with open questions that are automatically graded. Indeed, Maple TA (www.maplesoft.com), Aleks (www.aleks.com), MyMathLab (www.mymathlab.com) and WebAssign (www.webassign.net) are commercial systems offering this type of facilities, while WIMS (wims.unice.fr) and WebWORK (webwork.math.rochester.edu) are some open source initiatives in this direction. The feedback that these systems provide, however, is usually very restricted. STACK (www.stack.bham.ac.uk), DWO (www.fi.uu.nl/dwo) and the LeActiveMath (www.leactivemath.org) system, are systems for mathematical exercises offering more intelligent feedback, see [6, 7] and [8]. Here we report on our own initiatives to create an interactive exercise system as part of our system MathDox (www.mathdox.org), which is capable of providing valuable feedback going beyond the correct/incorrect reply.

MathDox is a system serving dynamic and interactive webpages with high quality rendering of mathematics and easy access to mathematical services like computer algebra systems or dynamic geometry software, see [4, 5]. MathDox shows its potential when demonstrating the workings of an algorithm or explaining new concepts with dynamic on-screen calculations. One of the key features of MathDox, making the above possible, is the semantic encoding of mathematical expressions by OpenMath.

In this paper we focus on some features of the MathDox system and, in particular, we explain how these features have been used to create a rich environment, called MESS (MathDox Exercise SyStem) for parametrized mathematical exercises providing easy access to all kinds of mathematical (web)services, taking care of good presentation of mathematics, offering easy and natural input of mathematics, and of facilities to provide meaningful feedback to users. In particular, we will demonstrate how we make use of the semantic encoding of mathematical expressions with OpenMath.

Participating in the JEM network gave us a good opportunity to learn and discuss initiatives in educational mathematics, and partly shaped the MESS.

2. The MathDox exercise format

MathDox sources combine in a modular way various existing XML formats best suited for our purposes. Each format contributes a useful facet for interactive documents. The main XML formats used in MathDox exercises are: LEAMEL, the LeActiveMath Exercise Language [3]; OpenMath, for semantic encoding of mathematics; Jelly, a programming and scripting format. These formats, their use and purpose, will now be discussed. LEAMEL is the main format for a MathDox exercise. It captures the structure of an exercise.

A MathDox exercise can be considered to be an automaton. The various states of the automaton, called **interactions**, are presented by the MathDox player as webpages to the user. These interactions may contain the question of an exercise, the feedback on a correct or wrong answer and some hints. The transitions from one state into another, called **answer_maps**, are ruled by the actions of the user, and the results of various queries to mathematical services available within the exercise. We explain this by way of the following example: Determine $\frac{1}{4} + \frac{1}{6}$. This is the question of the exercise, it is the first state of the automaton that the user will encounter when starting up the exercise. Now the user may provide an answer, correct or incorrect. For each of these two possibilities the exercise contains an interaction, that the user will visit after supplying his answer. This exercise can be encoded in the XML-exercise format.

```
<exercise>
  <interaction id='question'>
    <para>Determine
      <para>
        <OMOBJ>
          <OMA><OMS name='plus' cd='arith1' />
            <OMA><OMS name='divide' cd='arith1' />
              <OMI>1</OMI><OMI>4</OMI>
            </OMA>
          <OMA><OMS name='divide' cd='arith1' />
            <OMI>1</OMI><OMI>6</OMI>
          </OMA>
        </OMOBJ>
      </para>
    </para>
    <para>
      <userinput type='blank'><set name='answer'></userinput>
    </para>
    <answer_map>
      <choose>
        <when target='correct'>
          <query>
            <OMOBJ>
              <OMA><OMS name='eq' cd='relation1' />

```

```

<out name='answer' />
<OMA><OMS name='divide' cd='arith1' />
  <OMI>5</OMI><OMI>12</OMI>
</OMA>
</OMA>
</OMOBJ>
</query>
</when>
<otherwise target='incorrect' />
</choose>
</answer_map>
</interaction>
<interaction id='correct'>
  <para>Well done!</para>
</interaction>

<interaction id='incorrect'>
  <para>Sorry, that is wrong.</para>
<answer_map text='Try again'>
  <interaction xref='question' />
</answer_map>
</interaction>
</exercise>

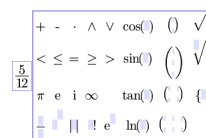
```

The question is posed in the interaction with id equal to `question`. In that interaction one also finds a `userinput-tag`. This tag represents an input field for the user. Inside the webpage presenting this interaction, the `userinput-tag` is replaced by the MathDox-formula editor, see Figure 1. The answer of the user is set to the variable with name `answer`. In the `answer_map` at the end of the interaction, a query (to for example a Computer Algebra System) is defined that checks whether the answer is equal to $\frac{5}{12}$. This `answer_map` redirects the user either to the interaction with id `correct` or the interaction with id `incorrect`. The latter interaction offers the possibility to try the exercise again.

The mathematics inside the exercise is encoded in OpenMath, a format for semantically rich encoding of mathematics. Within MathDox mathematics appears in various forms: Mathematics as used in computations by software packages or mathematics solely meant for the user to read or sometimes for both the computer and the user. For each type of usage one wants the mathematics to have specific properties. In general a user will be able to grasp the meaning behind a, possibly ambiguous, mathematical expression, often helped by the context in which the expression appears. Computer software, however, will need its mathematics to be completely unambiguous, since it cannot benefit from the context in the way a mathematically skilled reader does in order to solve gaps caused by incompleteness and ambiguity.

OpenMath has been chosen as the main format for mathematics within MathDox documents. It is well suited because it is semantically rich, unambiguous, XML-based and can easily be transformed into other formats, such as MathML and \LaTeX , which are better suited for presentation. These transformations allow MathDox documents to use OpenMath where semantics is important and switch to MathML or \LaTeX , when the mathematics needs to be presented to the user either on screen or on paper.

Determine $\frac{1}{4} + \frac{1}{6}$.



Submit

FIGURE 1. Exercise in MESS

To give the reader an idea of the OpenMath encoding of the semantics of a mathematical expression, we analyze the snippet of OpenMath representing $\frac{1}{4} + \frac{1}{6}$:

```
<OMOBJ>
  <OMA><OMS name='plus' cd='arith1' />
    <OMA><OMS name='divide' cd='arith1' />
      <OMI>1</OMI><OMI>4</OMI>
    </OMA>
    <OMA><OMS name='divide' cd='arith1' />
      <OMI>1</OMI><OMI>6</OMI>
    </OMA>
  </OMOBJ>
```

In this encoding we find an application (OMA) of a symbol (OMS) with name `plus` and defined in a so-called content dictionary (cd) with the name `arith1` being applied to the two fractions. Each fraction consists of an application of the symbol `divide` to two integers (OMI). The semantics of this expression is hidden in the attributes `name` and `cd`. Indeed, the content dictionaries contain the semantical information of the various symbols. They are provided with the document.

The semantic information available from the OpenMath expressions enables easy translations between OpenMath and application specific syntax. At the moment MathDox supports interaction with various mathematical (web)services such as the computer algebra systems Mathematica, Maple, GAP, Maxima, Wiris, Magma and Singular, and the dynamic geometry systems Geogebra and Cinderella.

Interoperability between these systems is achieved by making use of so-called OpenMath phrasebooks. For each of the forementioned systems there is a separate OpenMath phrasebook (Caprotti, Cohen & Riem [1]) that handles the translation from OpenMath to the computer algebra system language and back.

The types of possible computations are only limited by what the computer algebra systems are capable of performing and what the OpenMath phrasebooks will be able to translate. However, the phrasebooks also allow for communication in the application's specific syntax.

To specify and fine-tune reactions of a MathDox document to user input, an author needs programming constructs. For this purpose Jelly¹ has been included in the MathDox format. Jelly is a JSP-like XML-language². Jelly can be used for conditional statements, loops, variables, calls to Java objects and calls to web services. The specific use of Jelly in exercises will be explained in the examples in the following sections.

3. MathDox software

So far we have only discussed the MathDox Exercise format and just briefly mentioned the server-side software responsible for processing of this format. In this section we discuss the MathDox software.

The MathDox Player is responsible for making MathDox documents accessible over the web. Its task is similar to that of a web server in the sense that both a web-server and the MathDox Player offer stored documents from the server to the outside world.

¹<http://commons.apache.org/jelly>

²<http://java.sun.com/products/jsp>

The main difference between a normal web-server and the MathDox Player is that a web-server offers ready-made HTML files, whereas the MathDox Player dynamically creates these HTML files. On request of the user the MathDox server collects data from the source document, the user and from mathematical backengines providing services, and creates, by applying a number of XSL transformations on this input, a new view on the document.

Collecting data and converting the MathDox source together with the input of the user and backengines into presentable HTML pages is the main task of the MathDox Player. To enable the users of MathDox documents to interact with the system, HTML offers various options, like buttons, text areas and links. However, there is no standard way to communicate mathematics in a meaningful way. Especially, since we want to enable the interaction with various mathematical services offered by the system, we require the user to provide semantically rich mathematical expressions. To this end we have developed a formula editor featuring:

- a two-dimensional WYSIWYM (what you see is what you mean) interface.
- semantic representation of the formulae in OpenMath.

The editor is written in Javascript and can easily be integrated in HTML web pages.

4. How to handle student answers

Mathdox exercises consist of various **interactions**. A crucial part in these exercises is the transition between these various **interactions**. As explained above these transitions are ruled by the **answer_maps**, and depend on various parameters, including student answers to the questions posed in the exercises.

In the example of Section 2, we have encountered the exercise ‘Determine $\frac{1}{4} + \frac{1}{6}$ ’. The query

```
<query>
  <OMOBJ>
    <OMA><OMS name='eq' cd='relation1' />
      <out name='answer' />
      <OMA><OMS name='divide' cd='arith1' />
        <OMI>5</OMI><OMI>12</OMI>
      </OMA>
    </OMA>
  </OMOBJ>
</query>
```

where **answer** is the name of the variable in which the answer to this question is posed, is used to decide whether the answer is correct or not.

This query is shipped to a mathematical service, and evaluated. If the response is **True** the student has passed the exercise, if not he has failed it. The default service used by MESS is a Computer Algebra System, e.g. Mathematica or Maxima.

Although at first sight this seems to be a good way to proceed, there is a lot of ambiguity in this approach.

We mention some problems. If the student supplies the answer $\frac{5}{12}$, then the query returns **True**; but this will also be the case when the student answers $\frac{10}{24}$, and also when he answer $\frac{1}{4} + \frac{1}{6}$. Maybe, the second answer, $\frac{10}{24}$, is acceptable, but the third, $\frac{1}{4} + \frac{1}{6}$,

certainly not. In general, a CAS will reply **True** when testing equality of these possible answers with $\frac{5}{12}$.

A similar problem occurs when we ask a student to simplify the expression $\frac{x^4-1}{x^2-1}$ as far as possible. Student answers could be $\frac{x^4-1}{x^2-1}$, $\frac{x^3-x^2+x-1}{x-1}$, $1+x^2$ or x^2+1 . Here, the last two answers are probably the ones we would like to approve.

Of course, we can program the CAS in such a way that it only returns **True** in the desired situations. Although MESS allows for this approach, it is not the preferred way of handling this kind of problem. Indeed, it would involve too much of CAS-specific programming in an exercise, while our goal is to be as much independent of the choice of CAS as possible.

The systems STACK and LeActiveMath have attacked these problems by designing various types of equivalence checks, see [6, 8]. For example, in STACK one can distinguish the following equality checks for two objects (e.g. students answer and answer provided by the system):

CASEqual	Are the parse trees of the two expressions equal
Equal_com_ass	Are they equal up to commutativity and associativity of elementary arithmetic operations? E.g. $a + b = b + a$, but $x + x \neq 2x$.
AlgEquiv	Are they algebraically equivalent, i.e. does the difference simplify to zero within the CAS used?
SubstEquiv	Can we find a substitution of the variables of ex2 into ex1 which renders ex1 algebraically equivalent to ex2?
SameType	Are the two expressions of the same type? Note that this test works recursively over the entire expression.

LeActiveMath provides similar checks. These equality checks cover a lot of cases, but certainly not all.

What do we do when a student provides a complete computation instead of just the final answer?

Or suppose one asks the student to write the gcd of 111 and 93 as a linear combination of 111 and 93. In this case you expect an answer in the form $a \cdot 111 + b \cdot 93 = \text{gcd}(111, 93)$, where a and b are integers. But how do we check correctness?

As a (partial) solution to these problems MathDox use of XPath functionalities in Jelly. These functionalities enable us to inspect the student's answer in similar ways as is done in STACK, but offers more flexibility.

For example, suppose the student provides an answer to the question of determining $\frac{1}{4} + \frac{1}{6}$, and this answer is an OpenMath expression set in the variable `answer`. Then we want the answer only to involve integers and division. The XPath-test

```
<if select="$answer//OMS[not(@name='divide' and @cd='arith')]"/>
```

tells us whether there is any other OpenMath symbol, different from

```
<OMS name='divide' cd='arith'/>
```

in the student's answer. If we encounter another symbol, then we can report this to the student and ask him to provide an answer only involving integers and division.

In answering the second question on simplifying $\frac{x^4-1}{x^2-1}$, we might test the student's answer to be equal to $x^2 + 1$ by not only using a query that checks equality up to algebraic equivalence (using a CAS) but also performing a check on the absence of the divide symbol:

```
<if select="not($answer//OMS[@name='divide' and @cd='arith1'])"/>
```

These XPath tools also give us the possibility to select various parts out of an OpenMath expression. For example, if the student answer equals the OpenMath expression

```
<OMOBJ>
  <OMA><OMS name='divide' cd='arith1' />
    <OMI>a</OMI><OMI>b</OMI>
  </OMA>
</OMOBJ>
```

where **a** and **b** are two integers, then with

```
<set name='enumerator'>
  <copyOf select='$answer//OMS[@name='divide']/following-sibling::*[position()=1]' />
</set>

<set name='denominator'>
  <copyOf select='$answer//OMS[@name='divide']/following-sibling::*[position()=2]' />
</set>
```

we can select the enumerator and denominator of **answer**. They can now be checked individually to be equal to 5 and 12, respectively.

Moreover, using XPath makes it possible to have a student provide a computation like

$$\frac{1}{4} + \frac{1}{6} = \frac{3}{12} + \frac{2}{12} = \frac{5}{12}.$$

With XPath we can chop this computation into

$$\frac{1}{4} + \frac{1}{6} = \frac{3}{12} + \frac{2}{12} \text{ and } \frac{3}{12} + \frac{2}{12} = \frac{5}{12}$$

and check each of these equations separately.

As a final example we mention that with XPath we can also analyse an answer like $-5 \cdot 111 + 6 \cdot 93 = 3$ to write the gcd of 111 and 93 as a linear combination of 111 and 93. Indeed, we can select both left and right hand side of the equality and check that they are equal to $\text{gcd}(111, 93)$. Moreover, we can check that one of the sides is an integer, and the other side is a linear combination of 111 and 93.

These kinds of XPath operations provide powerful tools to analyse student's answers, do type checking, test for forbidden symbols, and select subexpressions for further computation.

5. MESS at work in an example

In this section we will illustrate the possibilities of MESS by an extended example. We continue with the exercise as described above: add two fractions. However, now we use some more of the facilities offered by MathDox to create a more interactive exercise, which provides the student with useful feedback. First generalize the question by parametrizing it. For this purpose we add the new randomized parameters **a**, **b**, **c** and **d** to the first interaction and introduce the variables with names **sum** and **answer**.

```

<interaction id='question'>
<random var='a' minimum='1' maximum='9'/> <random var='b' minimum='3' maximum='9'/>
<random var='c' minimum='1' maximum='9'/> <random var='d' minimum='3' maximum='9'/>
<set name='sum'>
<OMOBJ><OMA><OMS cd='arith1' name='plus'/>
<OMA><OMS name='divide' cd='arith1'/>
<OMI>${a}</OMI><OMI>${b}</OMI>
</OMA>
<OMA><OMS name='divide' cd='arith1'/>
<OMI>${c}</OMI><OMI>${d}</OMI>
</OMA>
</OMOBJ>
</set>

<set name='answer'>
<OMOBJ> <OMA><OMS name='divide' cd='arith1'/>
<OMV name='a'/><OMV name='b'/>
</OMA>
</OMOBJ>
</set>

<para>Determine <out name='sum'/>.</para>
<para>Provide your answer in the form <math>\frac{a}{b}</math>.</para>
<para>
<out name='sum'/>=<userinput type='blank'>
<set name='answer'/><OMOBJ><out name='answer'/></OMOBJ></set>
</userinput>
</para>

<answer_map text='check your answer'> ... </answer_map>
</interaction>

```

This results in a randomized question. Next we want to extend the responses on a student's answer. Besides the correct and incorrect feedback, we can also provide the following types of feedback: If the answer is not fully simplified (i.e., the gcd of the numerator and denominator is not equal to 1) we provide the following feedback:

```

<interaction id='almost'>
<para> Indeed, <out name='answer'/> equals <out name='sum'/>, but maybe you can simplify your answer a bit. </para>
<answer_map text='Try again'>
  <interaction xref='question'/>
</answer_map>
</interaction>

```

Determine $\frac{2}{8} + \frac{8}{5}$.

Provide your answer in the form $\frac{a}{b}$.

$$\frac{2}{8} + \frac{8}{5} = \frac{a}{b}$$

check your answer

Indeed, $\frac{74}{40}$ equals $\frac{2}{8} + \frac{8}{5}$, but maybe you can simplify your answer a bit.

Try again

FIGURE 2. A randomized exercise and feedback

Another useful type of feedback is feedback on common mistakes, referred to as 'buggy rules'. A common buggy rule for adding fractions is the addition both the numerators and denominators. If a student provides such an answer, then the exercise replies with the following feedback.

```

<interaction id='buggy'>
  <para>Oops, did you add the denominators and enumerators? This is not allowed.</para>
  <para>You first have to make the denominators equal and then add the two fractions by adding the enumerators.</para>
<answer_map text='Try again'>
  <interaction xref='question' />
</answer_map>
</interaction>

```

Having added these interactions, it remains to specify the `answer_map` in the `question`-interaction. This `answer_map` will have the following structure:

```

<choose> <when target='correct'> ...</when>
  <when target='buggy'> ...</when>
  <otherwise target='incorrect' />
</choose>

```

Each `<when/>`-tag contains a query involving the student's answer to a CAS resulting in either a `True` or `False`. The user is redirected to the first target for which this query yields a `True`. For the `correct`-target we have to analyze the student's answer. Indeed, we check that the answer is a fraction not only equal to the expected answer, but that the gcd of the enumerator and denominator is equal to 1. Here we make use of the XML-structure of the answer:

```

<OMOBJ><OMA><OMS name='divide' cd='arith1' />
  <OMV name='a' />
  <OMV name='b' />
</OMA>
</OMOBJ>

```

With the Jelly-command

```
<copyOf select='$answer//OMS[@name='divide']/following-sibling::*[position()=1]' />
```

we can select the first sibling of the divide symbol, i.e., we get the enumerator of the fraction. Similarly

```
<copyOf select='$answer//OMS[@name='divide']/following-sibling::*[position()=2]' />
```

provides us with the denominator. We use this in the following query that checks correctness of the answer, and whether it is in simplified form.

```

<query>
<OMOBJ><OMA><OMS name='and' cd='logic1' />
  <OMA><OMS name='eq' cd='relation1' />
    <out name='answer' /><out name='sum' />
  </OMA>
  <OMA><OMS name='eq' cd='relation1' />
    <OMA><OMS name='gcd' cd='arith1' />
      <copyOf select='$answer//OMS[@name='divide']/following-sibling::*[position()=1]' />
      <copyOf select='$answer//OMS[@name='divide']/following-sibling::*[position()=2]' />
    </OMA>
    <OMI>1</OMI>
  </OMA>
</OMOBJ>
</query>

```

Now the second `<when/>`-tag can be filled with

```

<query>
<OMOBJ><OMA><OMS name='eq' cd='relation1' />
  <out name='answer' /><out name='sum' />
</OMA>
</OMOBJ>
</query>

```

while the target `buggy` will be chosen if the following query yields a `True`.

```

<query>
<OMOBJ><OMA><OMS name='eq' cd='relation1' />
  <out name='answer' />
  <OMA><OMS name='divide' cd='arith1' />
    <OMI>${a+c}</OMI><OMI>${b+d}</OMI>
  </OMA>
</OMOBJ>

```

```
</query>
```

This results in the following XML-encoding of the complete exercise.

```
<exercise>
<interaction id='question'>
<if test='${ft ne 'false'}>
<random var='a' minimum='1' maximum='9'/> <random var='b' minimum='3' maximum='9'/>
<random var='c' minimum='1' maximum='9'/> <random var='d' minimum='3' maximum='9'/>
<set name='sum'>
<DMOBJ><OMA><OMS cd='arith1' name='plus'/>
  <OMA><OMS name='divide' cd='arith1'/>
    <OMI>${a}</OMI><OMI>${b}</OMI>
  </OMA>
  <OMA><OMS name='divide' cd='arith1'/>
    <OMI>${c}</OMI><OMI>${d}</OMI>
  </OMA>
</DMOBJ>
</set>
<set name='answer'>
<DMOBJ><OMA><OMS name='divide' cd='arith1'/>
  <OMV name='a'/><OMV name='b'/>
</DMOBJ>
</set>
</if>
<set var='ft'>false</c:set>

<para>Determine <out name='sum'/.</para>
<para>Provide your answer in the form <math>\frac{a}{b}</math>.</para>
<para><out name='sum'/.><userinput type='blank'>
  <set name='answer'><DMOBJ><out name='answer'/.></DMOBJ></set>
  </userinput></para>

<answer_map text='check your answer'>
<choose>
<when target='correct'>
<query>
<DMOBJ><OMA><OMS name='and' cd='logic1'/>
  <OMA><OMS name='eq' cd='relation1'/>
    <out name='answer'/.>
    <out name='sum'/.>
  </OMA>
  <OMA><OMS name='eq' cd='relation1'/>
    <OMA><OMS name='gcd' cd='arith1'/>
      <copyOf select='${answer//om:OMS[@name='divide']/following-sibling::*[position()=1]}'/>
      <copyOf select='${answer//om:OMS[@name='divide']/following-sibling::*[position()=2]}'/>
    </OMA>
    <OMI>1</OMI>
  </OMA>
</DMOBJ>
</query>
</when>

<when target='almost'>
<query>
<DMOBJ><OMA><OMS name='eq' cd='relation1'/>
  <out name='answer'/.><out name='sum'/.>
</DMOBJ>
</query>
</when>

<when target='buggy'>
<query>
<DMOBJ><OMA><OMS name='eq' cd='relation1'/>
  <out name='answer'/.>
  <OMA><OMS name='divide' cd='arith1'/>
    <OMI>${a+c}</OMI><OMI>${b+d}</OMI>
  </OMA>
</DMOBJ>
</query>
```

```

</when>
<otherwise target='incorrect'/>
</choose>
</answer_map>
</interaction>

<interaction id='correct'>
<para>Well done!</para>
<answer_map text='Try a new exercise'>
  <set var='ft'>true</c:set>
  <interaction xref='question'/>
</answer_map>
</interaction>

<interaction id='almost'>
<para>Indeed, <out name='answer'/> equals <out name='sum'/>, but maybe you can simplify your answer a bit.</para>
<answer_map text='Try again'>
  <interaction xref='question'/>
</answer_map>
</interaction>

<interaction id='buggy'>
<para>Oops, did you add the denominators and enumerators? This is not allowed.</para>
<para>You first have to make the denominators equal and then add the two fractions by adding the enumerators.</para>
<answer_map text='Try again'>
  <interaction xref='question'/>
</answer_map>
</interaction>

<interaction id='incorrect'>
<para>Sorry, that is wrong!</para>
<answer_map text='Try again'>
  <interaction xref='question'/>
</answer_map>
</interaction>
</exercise>

```

6. Organising courses with MESS

Despite the fact that MathDox and MESS can be used to create and play exercises, they are not intended or designed to be a Learning Management System (LMS) like Blackboard, Moodle, or Sakai. To enable the use of MathDox exercises in an LMS, we developed a scoring system and made it possible to include MathDox documents in SCORM (Sharable Content Object Reference Model)³ packages. In this way it is possible to use the facilities of these LMS to organise courses with MESS.

To facilitate the management of these SCORM packages, and the administration of the results of the students, it is possible to use the functionalities of the used LMS. But we do not rely on the LMS. Indeed, we have built a management system for the content and student administration, which enables users to create scorm content out of our database of exercises, attach them to various courses and get extended reports on the results of students on the exercises. This has been implemented in the project Wortel TU/e⁴, the mathematical web environment of the Technical University of Eindhoven, built upon MESS and Moodle. Here one finds also a collection of about 1000 parametrized exercises in basic highschool math, undergraduate calculus, linear algebra, algebra and discrete mathematics.

³www.adlnet.gov/scorm

⁴wortel.tue.nl

7. Conclusions

The MathDox Exercise System MESS provides a very flexible and rich environment for mathematical problem solving. Besides offering easy access to various computer algebra systems, high quality rendering of mathematics in browsers, and a convenient mathematical input editor, it also provides many ways of giving immediate, meaningful and intelligent feedback on user's actions.

References

- [1] O. Caprotti, A.M. Cohen, A.M., Riem, M. , Java phrasebooks for Computer Algebra and Automated Deduction, Sigsam Bulletin (2000).
- [2] A.M. Cohen, H. Cuypers, E.R. Barreiro, H. Sterk, Interactive Mathematical Documents on the Web, in Algebra, Geometry and Software Systems (eds M. Joswig and N. Takayama), Springer Verlag, 289-306 (2003).
- [3] A. Cohen, H. Cuypers, D. Jibeteau, M. Spanbroek, LeActiveMath Exercise Language, Deliverable 7, LeActivemath, <http://www.leactivemath.org/publications1.html> (2004).
- [4] A.M. Cohen, H. Cuypers, E.R. Barreiro, MathDox : mathematical documents on the web. In M. Kohlhase (Ed.), OMDoc : An Open Markup Format for Mathematical Documents (pp. 262-265) Berlin: Springer-Verlag (2006).
- [5] A.M. Cohen, H. Cuypers, J.W. Knopper, M. Spanbroek, R. Verrijzer, MathDox - A System for Interactive Mathematics, in Proceedings of Ed-media, 5177-5182 (2008).
- [6] G. Gogvadze, A. G. Palomo, E. Melis, Interactivity of Exercises in ActiveMath Towards Sustainable and Scalable Educational Innovations Informed by the Learning Sciences Sharing. Good Practices of Research Experimentation and Innovation, Volume 133, Edited by Chee-Kit Looi, David Jonassen, Mitsuru Ikeda (2005).
- [7] E. Melis, J. Siekmann, ActiveMath: An Intelligent Tutoring System for Mathematics, in the proceedings of the Seventh International Conference 'Artificial Intelligence and Soft Computing, LNAI 3070, 91-101, ed. L. Rutkowski, J. Siekmann, R. Tadeusiewicz, L.A. Zadeh, Springer Verlag (2004).
- [8] C. Sangwin, STACK: making many fine judgements rapidly (2007).

Hans Cuypers, Jan Willem Knopper, Hans Sterk
Department of Mathematics and Computer Science
Eindhoven University of Technology
info@mathdox.org