

On Automatic Generation of Quizzes using MATLAB and XML in Control Engineering Education*

Technical Report fik07xml

M. Fikar

March 23, 2013

Abstract

The main aim of this report is to show how to employ MATLAB as a scripting and templating engine to generate automatically a large number of identical quizzes with different numerical values.

Two sources: computational part and templating part are used to construct an output neutral XML file. Then, MATLAB is employed in the next step using XSLT to generate a desired output format - either PDF or import format suitable for LMS Moodle.

History:

2013/3/23 – Version 1.2. Added Moodle MA export able to handle subproblems with numerical answers. Changed win1250 to utf8.

2007/8/26 – Version 1.1. Added Moodle MA export able to handle problems with several subproblems.

2007/4/7 – Version 1.

1 Introduction

In the last two decades computer technology revolutionised the world of computing and teaching. Tools are continually being developed to explore various ways that improve teaching and solve challenges that were unsolvable before.

One of the challenges is to deal with a large number of students. If they have the same quiz or assignment they tend to cheat. However, it is reasonable to give them quizzes of the same complexity so that some of them are not in advantage.

In this paper, we discuss creation and maintenance of large sets of questions for e-learning course on Automatic Control Fundamentals. This includes generation of on-line and off-line quizzes for the course with approximately 250 students. The approach taken includes MATLAB as a scripting language combined with XML templates to produce input files. These can be further processed using XSLT to obtain HTML representation, input to LMS system Moodle, or various versions of PDF files produced by \LaTeX .

2 Evaluation Module

Examination of the course is divided into written and oral parts.

The written part consists of a series of computational exercises with multiple choice questions with five choices each and students mark one correct choice. Students can use any printed references (books, tables, etc). Preparation of the written examination has been automated in a

*Department of Information Engineering and Process Control, Faculty of Chemical and Food Technology, STU, Radlinského 9, 812 37 Bratislava, fax : +421 2 52496469 and e-mail : miroslav.fikar@stuba.sk, web : www.kirp.chtf.stuba.sk

similar way as it is done with assignments. It is using MATLAB as the computational engine, XML/XSLT as the transformation engine, and L^AT_EX, HTML, and Moodle as formatting tools.

Currently, there is about 50 different problem choices with different levels of difficulty that can be given to students and that cover the whole range of the course. In the first run, MATLAB generates a random combination of them so that the required total number of points is achieved. Also problems with similar topics are grouped together and always only one of them can be in the quiz. Next, MATLAB generates random values for each problem and outputs them as a XML file. This is repeated for a given number of groups so that students cannot just copy the results of their neighbour. Thus each group has the same type of problems but with different values.

Finally, XML file is transformed into a final form. This can include printed version, html version, or a Moodle quiz.

The whole procedure will now be explained on a simple example.

3 Tutorial

Let us consider this simple example problem together with its solution.

3.1 Problem Definition

The closed loop system consists of a controlled system with transfer function of the form $G(s) = \frac{b_0}{s^2 + a_1s + a_0}$ and a PID controller of the form $G_c(s) = P + I/s + Ds$. If the setpoint value is changed at $t = 0$ from 0 to w , the permanent tracking error is given as:

$$e(\infty) = \begin{cases} w \left(1 - \frac{b_0 P}{a_0 + b_0 P}\right) & \text{if } I = 0 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

3.2 Template XML File

Each problem consists of a template file where items that are to be changed are marked with **##ID##** where ID identifies name of the particular variable part. The template file `tro.tpl` for our problem is as follows

```
<problem name="tro">
<subproblem score="3" shortans="yes" numerical="yes">
<problemtext>
  The closed loop system consists of a controlled system
  with transfer function of the form
  <span CLASS="math">G_p(s) = \frac{##b0##}{s^2 + ##a1## s + ##a0##}</span>
  and a controller of the form
  <span class="math">##gr##</span>.
  If the setpoint value is changed at <span CLASS="smath">t=0</span>
  from 0 to ##w##, the permanent tracking error is given as
</problemtext>
<answers>
  <choice order="##1##" ans="gooditem">##tross##</choice>
  <choice order="##2##" ans="baditem">##VAL2##</choice>
  <choice order="##3##" ans="baditem">##VAL3##</choice>
  <choice order="##4##" ans="baditem">##VAL4##</choice>
  <choice order="##5##" ans="baditem">no other choice is correct</choice>
</answers>
</subproblem>
</problem>
```

3.3 XML File Creation

The first step is to generate some admissible and random data. MATLAB is used for this task. Each problem has a function associated with it that prepares data that are to be changed each time. A typical function consists of three parts.

1. Generation of data. This is demonstrated below and closely follows mathematic problem definition.

```
function [str,template, valstr] = tro

w=fix(rand(1,1)*10)+1;

num=fix(rand(1,1)*10)+1; b0=num(1,1);
den=fix(rand(1,2)*10)+1;
a1 = den(1,2); a0 = den(1,1);
xx=fix(rand(1,3)*10)+1; pp=xx(1,1);ii=xx(1,2);dd=xx(1,3);
if (randn(1,1)<0)
    ii=0;
    tros = w*(1 - (b0 * pp)/ (a0+b0*pp) );
else
    tros = 0;
end
```

2. Generation of false data, preparation of parameters for the template file. In our example, we need to be sure that any of other three false answers is not the same as the true one. In addition, if the permanent tracking error is not zero, we give one of the false answers equal to zero. Finally, the transfer function of the controller is prepared taking care between PID and PD cases.

```
tross = sprintf('%.2f',tros);
choic = randn(1,4);
choi = sprintf('%.2f',choic);
while sum(findstr(choi,tross))>0
    choic = randn(1,4);
    choi = sprintf('%.2f',choic);
end
ss2 = sprintf('%.2f',choic(2));
ss3 = sprintf('%.2f',choic(3));
ss4 = sprintf('%.2f',choic(4));

if tros ~=0
    ss2 = '0.00';
end

if ii~=0
    [gr,e]=sprintf('G_R(s) = %.0f %+.0f s %+.0f/s ',pp,dd,ii);
else
    [gr,e]=sprintf('G_R(s) = %.0f %+.0f s ',pp,dd);
end
```

3. Finally, values for the template file have to be specified. These are then substituted to the template.

```
valstr= {'b0', int2str(b0)}
```

```

    'a1', int2str(a1)
    'a0', int2str(a0)
    'gr', gr
    'w', int2str(w)
    'tross', tross
    'VAL2', ss2
    'VAL3', ss3
    'VAL4', ss4};

template = 'tro.tpl';
str = writetpl(template, valstr);

```

The result is returned in string `str` to the calling routine. In our case the result can be as follows.

```

<problem name="tro">
<subproblem score="3" shortans="yes" numerical="yes">
<problemtext>
  The closed loop system consists of a controlled system
  with transfer function of the form
  <span CLASS="math">G_p(s) = \frac{5}{s^2 + 3 s + 5}</span>
  and a controller of the form
  <span class="math">G_R(s) = 7 +10 s </span>.
  If the setpoint value is changed at <span CLASS="smath">t=0</span>
  from 0 to 8, the permanent tracking error is given as
</problemtext>
<answers>
  <choice order="4" ans="gooditem">1.00</choice>
  <choice order="2" ans="baditem">0.00</choice>
  <choice order="5" ans="baditem">0.29</choice>
  <choice order="1" ans="baditem">-1.15</choice>
  <choice order="3" ans="baditem">no other choice is correct</choice>
</answers>
</subproblem>
</problem>

```

As we can see, the problem has been transformed into a multiple choice question with one correct answer. MATLAB has provided suitable random values and results. Mathematics is typeset using the L^AT_EX notation and closed in classes `math` or `smath`.

4 Transformation to Presentation Formats

Further transformation of format independent XML file is performed using either standard XSLT engines (e.g. Saxon (Kay, 2006), XT (Clark, 2005), etc) or builtin function `xslt` in MATLAB. In our case, three output formats have been specified:

- PDF – to be used in written examination. It should exist in versions for teacher and student.
- HTML – to be shown on a usual web page, together with a hint to a correct answer.
- Moodle – to be used in e-learning environment with automatic links to students gradebooks.

These three output formats correspond to different situations: students can access some of the questions freely at the web pages together with a solution. There, they can test the knowledge without any stressing conditions. The Moodle output can be used at the beginning of seminars

to check the students. And finally, paper output is used in written part of examinations. As all questions originate from a single source pool, students get familiar with them, and the final written examination results have been greatly improved.

The paper (PDF) version uses \LaTeX as it can create high quality PDF files from plain ASCII input and thus can act as a filter without user intervention. This would be very difficult if not impossible to achieve with WYSIWYG programs (MS Word, etc).

The formatting engine \LaTeX is invoked with the modified class `exams` (Van der Meer, 2002) that puts all problems together, randomises the order of choices, and can in two runs produce two PDF files – one for students and one for the evaluator where the correct answers are marked. The resulting part of the PDF available to teacher is shown in Fig. 1. The same for short-answer type question available to student is shown in Fig. 2.

One of the major problems with typesetting of output formats is mathematics on the web. Although the MathML standard exists, it is still not implemented in all web browsers. Therefore, the \LaTeX notation is used and transformed either using MimeTeX (creates pictures of mathematical objects, Forkosh (2006)) or JsMath (uses javascript engine, Cervone (2006)). Both approaches have some advantages and drawbacks but serve well their purpose.

The purpose of plain HTML format is to produce a free pool of quiz questions that serve for preparation before the actual exam. These are published on the Internet site of the course. Here, no evaluation is performed. The questions and responses are coupled with a javascript so that student can click on a response to be sure that his/her response is correct.

An example of this solution coupled with JsMath to typeset mathematic expression is shown in Fig. 3.

The last output format if needed for LMS Moodle. The idea is to prepare a large number of similar questions that could be used in Moodle quizzes in a random way – so that every student gets possibly different questions. Quizzes are used in each week to test preparation of students for the course.

Moodle provides several ways of importing questions from a text file. These include WebCT, GIFT, etc. As import files are plain ASCII, it is not very difficult to create XSLT template that transforms the original XML file to the desired format. We have chosen Moodle XML import and our problem in Moodle is shown in Fig. 4. Here, the standard mathematical tool is MimeTeX. However, JsMath support can be installed as well.

5 Technical Part of the Conversion

5.1 Function `writetpl`

This function called from each problem is responsible for several subtasks:

- Randomise the order of answers. This is important as some of the presentation formats cannot shuffle the answers. However, as template files have correct answer on a fixed place that cannot be moved, each answer becomes a random number in the attribute order. Then XSLT routine will be responsible to process the answers in sorting order, thus actually randomising them.

Placeholders `##n##` where `n` is an integer are reserved for `writetpl` and cannot be used elsewhere.

- It tests whether the designer has filled out all templates and that all are necessary - there must be one-to-one relationship between them.

The code of the function is given below.

```
function source = writetpl(infile, strings)
% WRITETPL - fill template file with concrete values

% Read from template file INFILE and write to string STR.  STRINGS
```

Problem 1. The closed loop system consists of a controlled system with transfer function of the form $G(s) = \frac{3}{s^2+5s+7}$ and a controller of the form $G_c(s) = 9 + 5s$. If the setpoint value is changed at $t=0$ from 0 to 10, the permanent tracking error is given as 3

2.06
 -1.15
 0.13
 0.29
 no other choice is correct

Figure 1: Example of a L^AT_EX generated PDF file with answer for a teacher

Problem 1. The closed loop system consists of a controlled system with transfer function of the form $G(s) = \frac{3}{s^2+5s+7}$ and a controller of the form $G_c(s) = 9 + 5s$. If the setpoint value is changed at $t = 0$ from 0 to 10, the permanent tracking error is given as 3

Answer:

Figure 2: Example of a PDF file generated by L^AT_EX that shows a short-answer type question without answer for a student

Problem 1 .) The closed loop system consists of a controlled system with transfer function of the form $G(s) = \frac{3}{s^2+5s+7}$ and a controller of the form $G_c(s) = 9 + 5s$. If the setpoint value is changed at $t=0$ from 0 to 10, the permanent tracking error is given as 3

no other choice is correct
 0.29
 -1.15
 2.06
 0.13

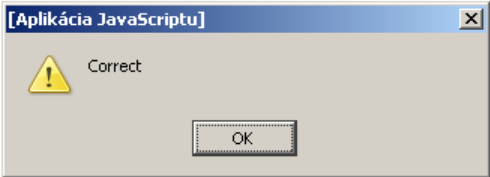


Figure 3: Example of a HTML file with JsMath

The closed loop system consists of a controlled system with transfer function of the form $G(s) = \frac{3}{s^2+5s+7}$ and a controller of the form $G_c(s) = 9 + 5s$. If the setpoint value is changed at $t=0$ from 0 to 10, the permanent tracking error is given as

Answer: a. 0.29
 b. 0.13
 c. 2.06
 d. no other choice is correct
 e. -1.15

Correct

Figure 4: Example of Moodle question with MimeTeX

```

% are defined as 2 column cell array of the form 'NAME', 'VALUE'.
% Functions searches in INFILE for ##NAME## and replaces it with
% VALUE. If the strings do not correspond with INFILE warnings are
% displayed. Reserved names are ##m## (any integer m) that are
% automatically replaced by randperm().

%read file in
fhandle=fopen(infile);
source=fscanf(fhandle,'%c',Inf);
fclose(fhandle);

n = size(strings,1);

% find how many ##integer## are in template
[start_idx, end_idx, extents, matches] = regexp(source,'##\d+##');
m = length(matches);

% add strings for permutation of answers
ii = randperm(m);
for i=1:m
% strip ## from numbers and pair them with permutations
    strings{n+i,1} = matches{i}(3:length(matches{i})-2); strings{n+i,2} = int2str(ii(i));
end

%replace all strings
for i=1:n+m
    searchstr = strcat('##',strings{i,1},'##');
    if isempty(findstr(source, searchstr)) & not(regexp(searchstr,'##\d+##'))
        res = sprintf('function writetpl (%s): searchstring %s not found', infile, searchstr);
        disp (res)
    end

    source=strrep(source, searchstr, strings{i,2});
end

%check what has left unchanged = forgotten
arr = findstr(source, '##');
if ~(isempty(arr))
    res=sprintf('function writetpl (%s): missing strings: ',infile);
    for i=1:length(arr)/2
        res = sprintf ('%s, %s', res, source(arr(2*i-1):arr(2*i)+1));
    end
    disp(res)
end

```

5.2 XML Structure

The XML structure for one problem is designed in such a way that either multiplechoice or shortanswer questions can be produced.

The main element is `<problem>` with attribute `name` defining the name of the problem. Each problem can have one or more tags of type `<subproblem>`. Its attributes define `score`. Then, there can be flag `shortans` that specifies whether it is possible to generate shortanswer type of question. It can have values `yes` and `no`. Finally, there can be flag `numerical` that specifies whether it is

possible to generate numerical type of question – it means that the result is a single number. It has value `yes`. Subproblem consists of two parts: `<problemtext>` and `<answer>`. Finally, answers contain five times `<choice>` where its attributes define whether the choice is correct (`gooditem`) and what is its position after XSLT.

Problemtext can contain one image of the form `` that should exist in the current directory.

Mathematics can be written in \LaTeX syntax enclosed in tags `` with attributes either `class='math'` or `class='smath'`. The second one can be used for formulas that can be typeset without special engines (for example `t=4` versus `t = 4`).

5.3 XML File

All problem functions return a string that holds the whole problem in XML form. However, all problems needs to be put together and a valid XML file needs to be produced.

This can again be done within MATLAB. Consider for example a script that produces 20 different questions of type `tro` shown in Section 3.

```
n=20;
str=cell(n);
for i=1:n
    str{i}='tro';
end
writemain('quiz.xml', str);
```

The script uses function `writemain` and outputs a file `quiz.xml`.

The source for `writemain.m` is following

```
function writemain(filename, strings)
% WRITEMAIN - generate XML file from questions in STRINGS

% Write a XML file into FILENAME that contains questions form cell
% vector STRINGS.

fw=fopen(filename,'w');
fprintf(fw,'<?xml version="1.0" encoding="utf-8"?\>\n');
fprintf(fw,'<!DOCTYPE quiz SYSTEM "quiz.dtd">\n');
fprintf(fw,'<n<quiz>\n');

for i=1:length(strings)
    str=eval(strings{i});
    fprintf(fw, '%c', str);
end

fprintf(fw,'</quiz>');
fclose(fw);
```

It includes proper XML header and a root element `<quiz>`.

The file `quiz.xml` can then be further processed using XSLT transformations for example in MATLAB as

```
XSLT('quiz.xml', 'moodle.xsl', 'moodle.txt')
```

where `moodle.txt` file will be produced using `moodle.xsl` transformation rules.

5.4 Figures

Note: figure handling is not satisfactory and works well only for \LaTeX presentation format.

Some of the questions can have one figure. If multiple questions of the same type are to be generated, each figure needs a unique name. Thus, the MATLAB part can be as follows:

1. Figure generation

```
step(b0, [1 a0]);
set(1, 'PaperUnits', 'centimeters')
set(1, 'PaperPosition', [3 10 11 8.25])

fileno = fix(rand(1)*100000+1);
filename = sprintf('file%d.eps',fileno);
print('-deps', filename)

valstr= {'NO', int2str(fileno) , ...
```

2. template

```
<problemtext>
  See figure 
</problemtext>
```

We can see that eps file has been produced and XML file holds only filename without extension eps. This is directly suitable for \LaTeX output but not for HTML. In the latter case we propose to either generate not only eps but png files as well, or to use batch conversion tools like `convert` from the package ImageMagic that can transform all files. The missing extension in HTML case can then be added during XSLT operation.

The issue how to store figures with HTML on the server depends much on the exact output presentation. For example Moodle allows in its XML input format figures to be encoded in base64. Therefore, after XSLT transformation some more work will be needed to correct the output file using (for example) some perl scripts. An example of one-liner in perl can be given as

```
perl -MMIME::Base64 -0777 -ne 'print encode_base64($_)'
  < file.jpg > file.jpg.base64
```

Another possibility is to upload all figures to web and specify only a link to them from quiz questions.

5.5 XSLT Transformations

Here we give some more comments on XSL transformations to desired output presentation formats. These transformations in principle define templates that describe what should be done with each element and attribute. The XSL language contains commands for repetition, branching, sorting, etc.

5.5.1 \LaTeX Output

The output for \LaTeX is specified by two transformations. One (`tex-mc.xsl`) produces pure multiplechoice (MC) questions and the other (`tex-sa.xsl`) mainly shortanswer (SA) questions or multiplechoice if it makes no sense with SA.

For both of them the output is a \LaTeX file suitable with modified class exam. The modifications of the class file were necessary as the original expected each question in one file linked to main document. We have changed this so that only one file is needed.

tex-mc.xml The desired output is given for one question below.

```

\documentclass[answers,nosep,scores]{exams1}
%% more definitions ...
\begin{exam}{\today}

\question{}{}{
\begin{problem}[\split]
\score{3}
  The closed loop system consists of a controlled system
  with transfer function of the form
  
$$G_p(s) = \frac{5}{s^2 + 3s + 5}$$

  and a controller of the form
  
$$G_R(s) = 7 + 10s$$
.
  If the setpoint value is changed at  $t=0$ 
  from 0 to 8, the permanent tracking error is given as
\begin{choice}
\baditem{-1.15}
\baditem{0.00}
\baditem{no other choice is correct}
\gooditem{1.00}
\baditem{0.29}
\end{choice}

\end{problem}
}

\end{exam}
\end{document}

```

Here we give some (not complete) parts of the XSL file with explanations.

The element `<quiz>` inserts at the beginning \LaTeX preamble, opens exam and then cycles over all elements of type `<problem>`. After all problems are processed, exam is closed and document finishes.

```

<xsl:template match="quiz">
\documentclass[answers,nosep,scores]{exams1}
%% more definitions ...
\begin{exam}{\today}
<xsl:for-each select="problem">
\question{}{}{
\begin{problem}[\split]
<xsl:apply-templates/>
\end{problem}
}
</xsl:for-each>
\end{exam}
\end{document}
</xsl:template>

```

The preceding code calls other elements using the command `<xsl:apply-templates/>`. As `<problem>` contains one or more elements of type `<subproblem>`, the corresponding template is searched. It outputs the score from its attribute and ignores other attribute `shortans`.

```

<xsl:template match="subproblem">
<xsl:text>\score{</xsl:text><xsl:value-of select="@score"/><xsl:apply-templates/>
</xsl:template>

```

Afterwards it calls other templates: `<problemtext>` and `<answers>`. The first one is not defined and default rule is applied – its contents is read.

The template `<answers>` cycles over all choices, sorts them according to attribute `order` and prints them with correct attribute `ans` that can either be `gooditem` or `baditem`.

```
<xsl:template match="answers">
<xsl:text>\begin{choice}</xsl:text>
<xsl:for-each select="choice">
  <xsl:sort data-type="number" select="@order"/>
  <xsl:text>\</xsl:text>
  <xsl:value-of select="@ans"/>
  <xsl:text>{</xsl:text>
  <xsl:apply-templates/>
  <xsl:text>}</xsl:text>
</xsl:for-each>
<xsl:text>\end{choice}</xsl:text>
</xsl:template>
```

If any of preceding contains elements `` or `` these will be transformed to environment `minipage` or to `mathematic mode` surrounded by dollars.

```
<xsl:template match="img">
  \begin{minipage}{0.39\textwidth}
  \includegraphics[width=\textwidth]{<xsl:value-of select="@src"/>}
</xsl:template>

<xsl:template match="span">
  <xsl:text>${</xsl:text>
  <xsl:value-of select="."/>
  <xsl:text>${</xsl:text>
</xsl:template>
</xsl:stylesheet>
```

`tex-sa.xml` The desired output is given for one question below.

```
\documentclass[answers,nosep,scores]{exams1}
\begin{exam}{\today}
\question{}{}{
\begin{problem}[\split]
\score{3}
  The closed loop system consists of a controlled system
  with transfer function of the form
  
$$G_p(s) = \frac{5}{s^2 + 3s + 5}$$

  and a controller of the form
  
$$G_R(s) = 7 + 10s$$
.
  If the setpoint value is changed at  $t=0$ 
  from 0 to 8, the permanent tracking error is given as
\shortanswer{1.00}
\end{problem}
}
\end{exam}
\end{document}
```

It differs from the previous only in the answer part where only good choice is specified. The XSL file should check whether the question can be of SA type. If not, MC should result.

We modify `<subproblem>` that holds the needed attribute `shortans`. This is tested against possible values and appropriate action is taken.

```

<xsl:template match="subproblem">
<xsl:text>\score{</xsl:text><xsl:value-of select="@score"/>}
<xsl:apply-templates select="problemtext"/>
<xsl:if test="@shortans = 'yes'">
\shortanswer{<xsl:apply-templates select="answers/choice" mode="sa"/>}
</xsl:if>
<xsl:if test="@shortans = 'no'">
\begin{choice}
<xsl:for-each select="answers/choice">
  <xsl:sort data-type="number" select="@order"/>
  <xsl:text>\</xsl:text>
  <xsl:value-of select="@ans"/>
  <xsl:text>{</xsl:text>
  <xsl:apply-templates/>
  <xsl:text>}</xsl:text>
</xsl:for-each>
\end{choice}
</xsl:if>
</xsl:template>

<xsl:template match="choice" mode="sa">
<xsl:if test="@ans = 'gooditem'">
<xsl:apply-templates/>
</xsl:if>
</xsl:template>

```

`moodle.xml` LMS Moodle supports many formats that can be used to import questions.

Moodle supports both MC and SA questions, but SA type is not very interesting as we need the quiz to be marked automatically. On the other hand, if some questions contain more subproblems, both MC and SA cannot be used by moodle as one question.

A possible outcome could be the multianswer (MA) type of question that can combine more subproblems. However in this type of question, answers can only contain pure text. Thus, the implementation is rather tricky.

Therefore for the MC type, we take only the first subproblem from each problem and transform it to moodle.

An example of Moodle XML import file for MC questions is shown below.

```

<?xml version="1.0" encoding="utf-8"?>
<quiz>
  <question type="multichoice">
    <name>
      <text>tro</text>
    </name>
    <questiontext format="moodle_auto_format">
      <text>
The closed loop system consists of a controlled system
with transfer function of the form

$$G_p(s) = \frac{5}{s^2 + 3s + 5}$$

and a controller of the form

$$G_R(s) = 7 + 10s$$
.
      </text>
    </questiontext>
  </question>

```

If the setpoint value is changed at $t=0$ from 0 to 8, the permanent tracking error is given as

```

</text>
  </questiontext>
  <image/>
  <image_base64/>
  <penalty>0.1</penalty>
  <hidden>0</hidden>
  <shuffleanswers>1</shuffleanswers>
  <single>true</single>

  <answer fraction="100">
    <text>1.00</text>
    <feedback>
      <text>Correct Answer!</text>
    </feedback>
  </answer>
  <answer fraction="0">
    <text>0.29</text>
    <feedback>
      <text>Sorry!</text>
    </feedback>
  </answer>
</question>
</quiz>

```

We also have implemented MA export that is able to deal with subproblems. These will be transformed to multiple MC questions internally. The corresponding XSL file is `moodle-cloze.xsl` and its output is shown below.

```

<?xml version="1.0" encoding="utf-8"?>
<quiz>
  <question type="cloze">
    <name>
      <text>tro</text>
    </name>
    <questiontext>
      <text>
        The closed loop system consists of a controlled system
        with transfer function of the form
        $$$G_p(s) = \frac{5}{s^2 + 3 s + 5}$$$
        and a controller of the form
        $$$G_R(s) = 7 + 10 s $$$ .
        If the setpoint value is changed at  $t=0$ 
        from 0 to 8, the permanent tracking error is given as
        A: 1
        B: 0.29
        Choose one answer: {2:MULTICHOICE:=A#Correct Answer!~B#Sorry!}
      </text>
    </questiontext>
    <image/>
    <image_base64/>
    <penalty>0.1</penalty>
    <shuffleanswers>0</shuffleanswers>
  </question>

```

</quiz>

We also have implemented MA export that can output MA numerical type of questions in sub-problems. These will be transformed to multiple MC na numerical questions internally. Numerical questions are produced from subproblems with flag `numerical="yes"`. The corresponding XSL file is `moodle-cloze-num.xsl` and its output is shown below.

```
<?xml version="1.0" encoding="utf-8"?>
<quiz>
  <question type="cloze">
    <name>
      <text>tro</text>
    </name>
    <questiontext format="html">
      <text><![CDATA[
The closed loop system consists of a controlled system
with transfer function of the form

$$G_p(s) = \frac{5}{s^2 + 3s + 5}$$

and a controller of the form

$$G_R(s) = 7 + 10s$$
.
If the setpoint value is changed at t=0
from 0 to 8, the permanent tracking error is given as
{2:NUMERICAL:~=1:0.001#Correct Answer!}<br/>
]]></text>
    </questiontext>
    <image/>
    <image_base64/>
    <penalty>0.1</penalty>
    <shuffleanswers>0</shuffleanswers>
  </question>
</quiz>
```

It can be noted that class `smath` is copied verbatim, class `math` contains double dollars and will be handled by `Mimetex`. Element `image_base64` can hold encoded figure. However, figure handling is not implemented for the time being.

6 Conclusions

We have described a procedure of creating quizzes in output neutral XML representation using MATLAB as a scripting engine. This has advantages of simplified creation of new questions and it makes possible to generate quizzes in many different ways that correspond to different pedagogical needs.

Acknowledgments

The authors are pleased to acknowledge the financial support of the Scientific Grant Agency of the Slovak Republic under grant No. 1/3081/06 and support of the Cultural and Education Grant Agency under the grant No. 3/3121/05.

References

D. Cervone. jsMath: A method of including mathematics in web pages, 2006. <http://www.math.union.edu/~dpvc/jsMath/> (online, 20.01.2006).

- J. Clark. XT – XSLT processor, 2005. <http://jclark.com/xml/xt.html> (online, 02.02.2005).
- J. Forkosh. MimeTeX: embedded LaTeX equations in html pages, 2006. <http://www.forksosh.com/mimetex.html>, (online, 20.1.2006).
- M. Kay. Saxon – the XSLT and XQuery processor, 2006. <http://saxon.sourceforge.net/> (online, 20.01.2006).
- H. Van der Meer. The exams package for LaTeX, 2002. www.ctan.org.