The Use of XSLT for Table Data Tasks Generation

Mikuláš Gangur
Department of Economy and Quantitative Methods
University of West Bohemia
Hradební 22, 350 11, Cheb
CZECH REPUBLIC
gangur@kem.zcu.cz

Abstract: - The paper introduces the principles of automatic generation of table data parameterized questions. Many problems for the domain of math/science are described and then solved in the form of table data. The paper describes solution from the proposal of table data XML structure to table transformation in XSL process. The style sheets for table data transformation to the Moodle XML format or to the LaTeX format are presented and final result of the question import to questions bank of LMS Moodle is shown. Finally the example of matrix tasks demonstrates the use of described principle.

Key-Words: - XML, XSL transformation, Automatic generation of tasks, Table data, Matrix, Moodle, E-learning, Matlab

1 Introduction

Electronic testing is useful for students to be aware of their learning styles, strengths and weaknesses, and from the teacher´s side to be provided with a variety of methods and approaches to choose the most suitable ones [6]. The practice tests for self-assessment of students represent an important part of learning through e-learning courses. In these tests with no time limit, the number of repetitions is not limited and the question sets an adaptive mode with a detailed commentary (feedback) on each problem. The important part of these problems especially for math/science domain consists of the table data problems. These tasks include for example the linear optimization problems solved with simplex method that processes with tables, matrixes problems etc.

There are numerous applications for automated test generation. These applications as well as the majority of Learning Management Systems employ tests generated randomly from the pool of questions in the question bank. The preparation of questions and building of such question bank is a difficult and time consuming job. We use a universal principle of automatic question generation and utilize particular parameterized questions for problems with table data. The principle simplifies and streamlines the questions bank building. A bank comprising thousands of unique problems was created using this generator as an aid courses from different areas of math/science. This was achieved by randomly generating unique tests for each student participating in the course.

The problem of automatic question generation has been solved in specific domains. In [1] the questions were generated in the area of electrical circuit analysis, another work focuses on question generation in the domain of object-oriented programming [5]. The problem of math/science tasks generation is addressed in [7]. Authors generate multiple choice questions in process of image modification with help of developed graphic tool. In this case new values of parameterized questions are not generated automatically.

In this paper in section 2 there is introduced the principle of automatic generator of tasks. Section 3 describes representation of table data tasks and the proposal of XML structure of such representation, discusses the structure of cloze question with embedded answers that are suitable for data table problem representation in generator and explains final transformation to selected output format. Section 4 presents the example of generated tasks including matrix tasks generation. Finally section 5 states the conclusion.

2 Automatic generation of tasks

The core of the whole process involves an automated generator of tasks (see [3]). The proposed generator construction follows the basic principles published in the technical report [2]. The principle of the generating system is a functional prescription of the solved problem (solver) together with the input data generator. The solver is a function with a variable number of the input parameters depending

ISBN: 978-1-61804-019-0

on a particular problem. The generator allows automatic generation of a "suitable" input data on the basis of the designated rules. These rules describe the relations within the input data. The generator algorithm is often implemented as backtrack procedure of the problem solution from the expected randomly generated problem results back to input data. The generator output is input data collection and the output of solver is the collection of requested output data.

The data, generated by means of the described procedure, are used as an input to the question generator together with the template of the question text and with the structure (template) of the universal output XML format. The generator allows processing a question requiring a numeric answer (NUM), a question with a short answer (SA) and a question with a multiple choice answer (MC). The possibility of processing a question with the embedded answer is very important. This type of question can consist of all three above mentioned question types. We use only the cloze question in the data table tasks generation.

The question generator inserts the generated input data into the question text and then the generator inserts this text together with any possible comments to the template of the requested question and the output results, calculated by the solver as answer (problem solution), as well. In the case of table data the system uses the built-in table template and generates dynamically the tables with respect to their sizes. The output of the question generator is a XML file in the universal format that is possible to transform to the format of the selected LMS by means of a particular question template and LaTeX versions of the test, consisting of both randomly chosen and randomly generated problems. The part of the output XML file is the XML description of generated table. According to the used dictionary the question can be generated in different languages. Current version of generator is able to process questions in both Czech and English language.

The application of such a generator for math/science tasks generation is implemented in Matlab. Generated problems constitute the bank of tasks implemented in the LMS Moodle (see [8]) as support math/science courses. The MoodleXML is selected as an output format of transformation for Moodle bank of tasks.

3 Table data task representation

The used automatic question generation processes with parameterized questions that are represented with cloze question i.e. question with embedded answers (see [3]). The cloze questions allow more inquiries of different types within one particular question. This type of question is the most suitable for table data tasks representation. The particular cells of tables are represented with mostly numerical question.

The table is represented in XML structure (see Listing 1). The root tag is <table> tag that consists of tag <rows> representing table rows (tags <row>). The values of table are stored in cells (tag <cell>) as a simple <cell_text> or as a question that allows to enter numerical values (answers) into table. These questions are represented as subquestions and they are parts of cloze question that represents whole table. That's why the table data are surrounded by tag <subquestions> as a part of cloze type question. (see [3]).

```
<table border='2' numcols="3" numrowa="3"
format="LaTeX">
 <table_title>Multiplication a*b</table_title>
 <rows>
  <row head='yes'>
   <cell input='no' head='yes' length='3'>
    <cell_text>a/b</cell_text>
   </cell>
   <cell input='no' head='no' length='1'>
    <cell_text>1</cell_text>
   </cell>
...
</row>
<row>
<cell input='yes' head='no' length='2'><subquestion
type="numerical" score="2">
....
</subquestion></cell>
....
 </row>
</rows>
</table>
```

Listing 1: XML representation of table (Source: own)

3.1 The structure of table data question

The above described generator creates the question consisting of table data in the form of proposed XML representation (see Listing 1). The discussed cloze question is employed for representation of such problems. The example of output XML

description of generated question is listed in Listing 2.

```
<quiz>
<question type="table" score="6">
 <name>
  <text>Table data task - 1</text>
 </name>
 <questiontext format="LaTeX">
 <text>
Fill the next table of two numbers multiplication
</text>
</questiontext>

 <image></image>
 <penalty>0.5</penalty>
 <hidden>0</hidden>

<subquestions>
<table border='2' numcols="3" numrowa="3"
format=”LaTeX”>
....
</table>
</subquestions>

</question>
</quiz>
```

Listing 2: XML representation of cloze question with table data (Source: own)

The basic XML structure of cloze question is identified in the listing. This structure consists of tags group <quiz><question><name><questiontext> <subquestions>. The XML description of table with particular subquestions is included in this structure (see Listing 1).

3.2 XSL transformation of task

In the next step the generated task is transformed in XSL process to Moodle XML format or LaTeX format (see [4]). XSL templates were created for tables and matrixes transformation and they were imported to the basic Moodle (LaTeX) XSL style sheets that care of the cloze question generation in selected output format (see [3]). Listing 3.4 and Listing 7 show some of the templates implemented in these style sheets.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/
1999/XSL/Transform" version="1.0" xmlns:math=
"http://exslt.org/math">
<xsl:template match="table">
\begin{table}[h]
\begin{tabular}{|<xsl:for-each select=
"rows/ row[1]/cell">
<xsl:variable name="i" select="position()" />
<xsl:if
test="../../row/cell[$i]/@input='yes'"><xsl:variable
name="colwidth"
select="round(0.65*math:max(../../row/cell[$i]/@len
gth))" />p{<xsl:value-of select="$colwidth"
/>em}|<xsl:if test="./@head='yes'">|</xsl:if></xsl:if>
<xsl:if test="../../row/cell[$i]/@input='no'">
c|<xsl:if test="./@head='yes'">|</xsl:if>
</xsl:if></xsl:for-each>}

<xsl:apply-templates select="rows" />
\end{tabular}\end{table}
\vspace{5mm}
</xsl:template>
....
<xsl:template match="table//row">
<xsl:if test="@head='yes'">
\hline
 <xsl:apply-templates/> \\
\hline\hline</xsl:if>
 <xsl:if test="@head='no'">\hline
 <xsl:apply-templates/> \\
 <xsl:if test="position()=last()-1">\hline </xsl:if>
</xsl:if>
</xsl:template>

<xsl:template match="table//cell">
<xsl:apply-templates />
<xsl:if test="position()!=last()"> &#x0026;</xsl:if>
</xsl:template>

<xsl:template match="table//cell_text">
 <xsl:apply-templates/>
</xsl:template>
</xsl:stylesheet>
```

Listing 3: XSL style sheet with templates for transformation tables to LaTeX format (Source: own)

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/
1999/XSL/Transform" version="1.0" xmlns:math=
"http://exslt.org/math">
<xsl:template match="table">
 &lt;br /&gt;
 <xsl:apply-templates select="table_title" />
```

```
&lt;table border = "1" &gt;
 <xsl:apply-templates select="rows" />
&lt;/table&gt;
</xsl:template>
....
<xsl:template match="table//row">
&lt;tr height="5"&gt; <xsl:apply-templates/>
&lt;/tr&gt;
</xsl:template>

<xsl:template match="table//cell">
&lt;td width="<xsl:value-of
select="math:max(ancestor::rows/row/cell/@length)
" />"&gt;<xsl:apply-templates />&lt;/td&gt;
</xsl:template>

<xsl:template match="table//cell_text">
 <xsl:apply-templates/>
</xsl:template>
</xsl:stylesheet>
```

Listing 4: XSL style sheet with templates for transformation tables to Moodle XML format (Source: own)

The main rule for table (matrix) LaTeX transformation is divided to instructions for columns with input cell(s) and without input cells with respect to column width definition in the table head (see Listing 3). The used predicates *test="../../row/cell[$i]/@input='yes/no'"* indicate for every column $i whether the column contains input cell(s) or not. The outer loop processes over all cells in first row and evaluates $i, as a cell position in the row i.e. column index. The indicator *c* is the output in the case of no input column and the *p{<width>em}* is the result in the case of input column. The column width is determined in variable colwidth definition *<xsl:variable name="colwidth" select="round(0.65*math:max(../../row/cell[$i]/@length))" />* as maximum width of all cells in the column. The function math:max from namespace math is employed for columns width determination as well as in the case of table in Moodle XML (see Listing 4). The final output defines the main structure of LaTeX table with no input cells for example

```
\left(\begin{array}{|c|c|c|c|c|}
.... table (matrix) body ....
\end{array}\right)
```

and the structure of table with any input cells for example

```
\left(\begin{array}{|p{3em}|c|c|p|{4em}|c|}
.... table (matrix) body ....
\end{array}\right)
```

The matrix type definition follows the concept of table data task. The matrix question is special case of table data question (see Listing 5). On the opposite of table data that are transformed to the Moodle XML as a HTML tags (see Listing 4) the matrix data are part of mathematic expression inserted to HTML page in LaTeX format (see Listing 6). These expressions are filtered by appropriate filters during HTML page loading.

```
<math>
<matrix numcols="3" numrows="3"
format=”Moodle”>
<rows>
 <row>
...
 <cell input='no' head='yes' length=’1’>
<cell_text>-3</cell_text></cell>
....
</row>
.....
 </rows>
</matrix>
</math>
```

Listing 5: XML representation of matrix (Source: own)

Listing 6 shows transformation rules for matrix to LaTeX. In the case of matrix with non-input data this rule is also used for transformation to Moodle XML format as mathematic expression. The matrixes with input data (see predicate *test="../../row/cell[$i]/ @input='yes/no'"* in Listing 3) are transformed as a table with templates for transformation to Moodle XML format (see Listing 4).

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/
1999/XSL/Transform" version="1.0" xmlns:math=
"http://exslt.org/math">

<xsl:template match="matrix">\left(
\begin{array}{<xsl:for-each select="rows/row[1]/
cell">
<xsl:variable name="i" select="position()" />
<xsl:if
test="../../row/cell[$i]/@input='yes'"><xsl:variable
name="colwidth"
```

```
select="round(0.65*math:max(../../row/cell[$i]/@length))" />p{<xsl:value-of select="$colwidth" />em}
</xsl:if>
<xsl:if test="../../row/cell[$i]/@input='no'">
c</xsl:if></xsl:for-each>}
<xsl:apply-templates select="rows" />
\end{array}\right)
\vspace{5mm}
</xsl:template>

<xsl:template match="matrix/rows">
 <xsl:apply-templates/>
</xsl:template>

<xsl:template match="matrix//row">
<xsl:apply-templates/> \\
</xsl:template>

<xsl:template match="matrix//cell">
<xsl:apply-templates />
<xsl:if test="position()&lt;last()-1">
 &#x0026;</xsl:if>
</xsl:template>
...
</xsl:stylesheet>
```

Listing 6: XSL templates for transformation of matrixes to LaTeX (Source: own)

4 The example of generated task

The described principles were employed for generation of inverse matrix task. The Listing 7 illustrates the input text of such parameterized problem. The parameter ##matrix## is replaced with randomly generated input matrix and the subquestion tag represents the object for inverse matrix elements inserting. This object is replaced with output inverse matrix.

```
Find inverse matrix to matrix A = ##matrix##

<subquestion type="table" id="1"><text></text>
</subquestion>
```

Listing 7: The problem of inverse matrix (Source: own)

Figure 1 shows Moodle assignment looking for inverse matrix. The inverse matrix elements are inserted in HTML <input> tags of <form> tag. All these <input> tags are located in the HTML table cells. Figure 2 shows the result of an XSL transformation of the same task as a part of the test generated in the LaTeX (PDF) format.

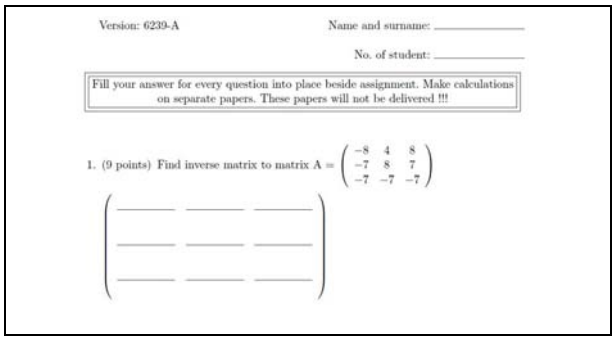

Figure 1: The inverse matrix task as a part of quiz in PDF format (Source: own)

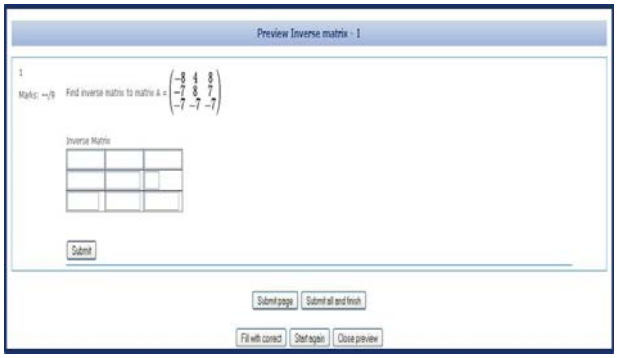

Figure 2: The inverse matrix task as a Moodle assignment from questions bank (Source: own)

5 Conclusion

This paper demonstrates the possibility of implementation of table data tasks such as questions with embedded answers (cloze question) in the LMS Moodle or as the part of quiz in LaTeX (PDF) format. This solution allows one to specify a task with several answers i.e. to fill the values of particular table cells. The principle of table structure generation is applied to process with matrixes. Two output formats are presented – Moodle TeX and LaTeX. The described procedure allows creating a lot of math/science tasks easily and readily. It is useful for effective practice of tasks for students in LMS (every student has got unique problem to solve) and for random generation of quizzes.

It is possible to implement table (matrix) style sheets for other output formats in the future work. We plan to develop the templates for XSL-FO transformation and transform the generated objects directly to PDF.

ISBN: 978-1-61804-019-0

Acknowledgment

This work is supported by Czech Ministry of Education, Youth and Sports under grant FRVŠ No. 1150/2011.

References:

[1] P.D. Cristea, R. Tuduce, Automatic Generation of Exercises for Self-Testing in Adaptive E-Learning Environments: Exercises on AC Circuits, *Inter. Workshop on Authoring of Adaptive and Adaptable Educational Hypermedia (Part of WBE)*, 2005

[2] M. Fikar, On Automatic Generation of Quizzes using MATLAB and XML in Control Engineering Education. *Technical Report fik07xml, OIRP UIAM FCHPT STU, 2007* [on-line][cit. 2010-10-10] Available at: http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=348

[3] M. Gangur, Automatic generation of cloze questions. *Proceedings of 3rd International Conference on Computer Supported Education, 2011, CSEDU'11*

[4] S. Holzner, *Inside XSLT*, New Rider's Publishing, 2002

[5] I. Hsiao, P. Brusilovsky, S. Sosnovsky, Web-based Parameterized Questions for Object-Oriented Programming, *World Conf. on ELearning in Corporate, Government, Healthcare, and Higher Education (ELEARN)*, 2008

[6] I. Šimonová, P. Poulová, M. Bílek, Learning styles within eLearning: Didactic strategies, *10th WSEAS International Conference on Applied Computer and Applied Computational Science, ACACOS'11*, pp. 160-164

[7] H.F. Ugurdag, E. Argali, O.E. Eker, A. Basaran, S. Gören, H. Özcan, Smart question (sQ): Tool for generating multiple-choice test questions, *Proceedings of the 8th WSEAS International Conference on Education and Educational Technology, EDU '09* , pp. 173-177

[8] Moodle, 2007 - A Free, Open Source Course Management System for Online Learning, [on-line][cit. 2011-01-11] Available at: http://moodle.org