# *Paper Based and Electronic Distributed Testing*

*Petr ŠALOUN*

Department of Computer Science, FEI VSB-Technical University of Ostrava,
17. listopadu 15, 708 33 Ostrava, Czech Republic.
Petr.Saloun@vsb.cz, http://www.cs.vsb.cz/saloun

***Abstract**: The support system for university entry exam was extended to system for testing via Internet/Intranet. Texts are written in $LaTeX$ typesetting system, stored in MySQL database. Tests are generated by Java and processed by pdf$LaTeX$ to produce final PDF Forms. Forms are delivered to student's web browser and the student's answers are stored in database. The answers are evaluated on demand. The results are sent via e-mail to students and the database of results is updated too. The case study gives the first real experiences obtained by simulation of the university exam from mathematics. The structure of the task and used $LaTeX$ macros are discussed.*
***Key Words:** multichoice test, web-based distributed testing, entry exam, computer support.*

## 1 *Introduction*

*At the beginning there was of the presented system was the computer support for the university entry exam process, during with the time pressure was enormous. The staffs have duty to correct all tests during a few hours. The possibility to make a mistake is very high but undesirable. The reasons mentioned above, and reasons, such as testing without subjective factors, and distributed testing were important enough to analyse, design and develop the computer support for the multichoice testing itself.*
*The multichoice test has a form of several sheets of paper, or has a form of the electronic document. In both cases the tests have to be generated, filled-in and evaluated. Before tests are generated, the tasks must be prepared and be ready for use. This is the job for web-based system for testing, presented here.*

## 2 *Basic notions and notations*

### 2.1 *Tasks, tests and categories*

*The basic element of the* test *is a* task. *A set of related tasks creates the* category *of tasks. Fig. 1 shows the final form of a task. The displayed task belongs to the category* quadratic equations.

**1** The equation $x^2 + px + 13 = 0$ has one solution $x_1 = 4 - \sqrt{3}$. The coeficient $p$ is

a) $-9$
b) such $p$ does not exists
c) 9
d) $\sqrt{55}$
e) $-8$

*Figure 1: One task - final appearance, printed form*

### 2.2 *The task structure*

*The multichoice task consists of the task's body and few answers. The task can contain none or some figures. The number of right answers depends on author's design. At least one answer per task offers the correct solution.*

*Tasks are written in LaTeX2e . Few LaTeX macros are written for better marking of tasks, see Fig. 2. The macros and definitions are placed in LaTeX style file* genexam.sty. *Usage of styles splits up the final form (appearance) of a task and the form stored (frozen) in the database.*
*The LaTeX user can use the macros directly. Fig. 3 displays the task's LaTeX source code with macros discussed in this article. The author's source contains the number of points for the answer.*

```
%*********************************************
% typical sequence + comment:
% \kategorie{cathegory} - cathegory name
% \bex - begin example
%  \ben - begin enumerate
%    \itm[3] - item [points],
%              for student generate no points, just \itm
%   \een - end enumerate
% \eex - end example
%*********************************************
```

*Figure 2: The summary of useful macros defined in* genexam.sty.

```
\bex
The equation $x^2 + px + 13 = 0$ has one solution
$x_1 = 4 - \sqrt{3}$.
The coeficient $p$ is
 \ben
    \itm[-1]
          $-9$
    \itm[-1]
          such $p$ does not exists
    \itm[-1]
          $9$
    \itm[-1]
          $\sqrt{55}$
    \itm[3]
          $-8$
 \een
 \eex
```

*Figure 3: One task - author's LaTeX source.*

### 2.3 *Marking of the answers*

*The task of the multichoice test offers a few answers and a student can choose the correct answer(s) according to his/her knowledge. Fig. 4 gives check boxes of a paper test.*


*Figure 4: Check box array for student's answer.*

*The right answer increases the student's point total, the wrong one decreases it. Missing answer does not change the total. The test consists of certain number of tasks, usually one task per one category. There is a time limit to fill in the test. Every test is marked with a TESTID. In the case of printed test the sheet is signed by student by his/her STUDENTID, in the case of electronic form the authentication of the student is more complex. The solution is given in another section of this article.*

### 2.4 *IT support*

*The support of information technology consists of:*

- *the computer support for task's writing,*
- *the database support for task's storage and maintenance,*
- *the tests generator,*
- *the $LaTeX$ macro package and text processing for high quality typesetting,*
- *distributed "hand made" collecting of answers, answers storage and evaluation; or distributed electronic filling of tests-forms and sending the answers via HTTP,*
- *the output for statistical analysis of results.*

*The IT solution is based on a distributed processing, using a Java Web Server (because of servlets), Java programming language and technology (generation of tests), JDBC communication protocol (SQL database access), pdf$LaTeX$ and PDF Forms for electronic forms, Adobe Acrobat Reader for displaying PDF documents and forms, and industry standard web browser such as Netscape Communicator or MS Internet Explorer. The Linux operating system, Apache web server, and the MySQL database are used for the operating environment. The mentioned software is free of charge. It sounds very well especially for academic purposes. For more details about the IT solution see [2].*

### 2.5 *Database for tasks*

*The task's structure is preserved in the text database structure. The $LaTeX$ sources are parsed and divided into the same parts, as the mentioned structure. Implemented parser is based on recursive descent technique, dividing the task into the parts. The divided task is stored into the database as a member of the proper category. The task can contain none or few figures. The figure filename is placed in special environment ($LaTeX$ macro). It allows us to recognise the figure filename and to store its content into the database. The supported formats of figures are PNG and $LaTeX$ PIC, EPS figures have to be converted by Perl script. The figures are stored as BLOBs (Binary Large OBjects) with link to the task.*

### 2.6 *Tests' generation*

*Tests are generated from the database of tasks. The set of tests, generated at the same time, is called* generation*. The conditions for one generation are the same. It means, that the database of tests does not vary from the generation. The identification of the generation is a part of the TESTID. The TESTID is the starting point for the random*

*number generator - used for deterministic permutations of answers and order of tasks in the test. The coincidence of two tests is small enough. It is necessary to have adequate number of tasks per category, and answers per task.*

### 2.7 *Tests data collecting and evaluation*

*Since they are two forms of tests, the data collection is discussed separately. The tests paper form is discussed here. The collecting phase of the answers of each test (set of tasks) is anonymous. The people without knowledge of their validity and identity process collected sheets. It allows us to employ not only teachers. The evaluation of collected data is done with the computer support. The same process of permutations, as for generation of tests, is used during evaluation of answers. The results can be stored for future statistical processing. A total of each student is printed with his/her name. All this data is coupled from both databases: personal one and result's one. The output contains the number of points (total points) obtained by each student. The list of totals is given to the dean for the admission process.*

*If the answers are collected, the evaluation process does not differentiate the test's form. The key point for evaluation of student's answers is the TESTID. It is used in the generating phase and also in the evaluating phase as an initial value of the random number generator.*

### 2.8 *Pure electronic testing*

*Filling up tests can be abstractly generalised as filling up forms, what is often realised by using a combination of web browser and HTML code. A user enters data into form field, browser sends them to the server for processing. There is, of course, a great limitation, because the specification of HTML does not support any possibility for typesetting, such as scaled view, vectors images, math formulas and so on. That's why it was necessary to find a product, which supports:*

- *data sending to specify server via HTTP,*
- *good look and functions for page description,*
- *security of test source,*
- *platform independence (UNIX and Windows environments).*

*Acrobat Reader in version 3.02 or 3.01 with plug-in Acrobat Forms fulfils these requirements. The Acrobat makes it possible to send data via HTTP, but must run inside a web browser as its plug-in. The format of sending data is FDF (an internal Acrobat format) or HTML - very useful for our project. The main purpose of the project of electronic testing was to create extension of the project University Entrance more effectively and verification on high schools", see [1, 2].*

*The application can dynamically generate tests with $LaTeX$ output. This solution is very portable. So, we had to choose a tool, witch allows us to create PDF document directly from $LaTeX$ sources including figures. We found out the pdf$LaTeX$ as most suitable, though it does not contain support for different forms and images other than PDF, JPEG or PNG. It is possible to insert a form field as a sequence of PDF 1.2 commands, but it would have no effect. Hyperref package contains macros, which support putting forms very well. EPS images can be converted to PDF using Perl script.*

### 2.9 *The Life Cycle*

*Students connect to the server during appointed week, and enter its authorisation request. Web and database server generates tests for them from database of tasks and inserts macros for PDF forms into $LaTeX$ source files. Names of the form field have relation to a number of tests and questions in database tables, what is necessary for server's recognition process. Application pdf$LaTeX$ can create required PDF from $LaTeX$ files and included images. Then the PDF files are sent to users via HTTP and web browser with plug-in for Acrobat to show them these. The appearance of a task in PDF form is given on Fig. 5. After fulfilling Acrobat returns the form data back. Server side recognises the TESTID, numbers of tasks, corresponding answers, and updates the database with students' answers. The teacher could demand the evaluation of the tests and the server informs him about the results via E-mail later. The student obtains his point total via e-mail too. It is possible to update a faculty information server or publish results on the web.*

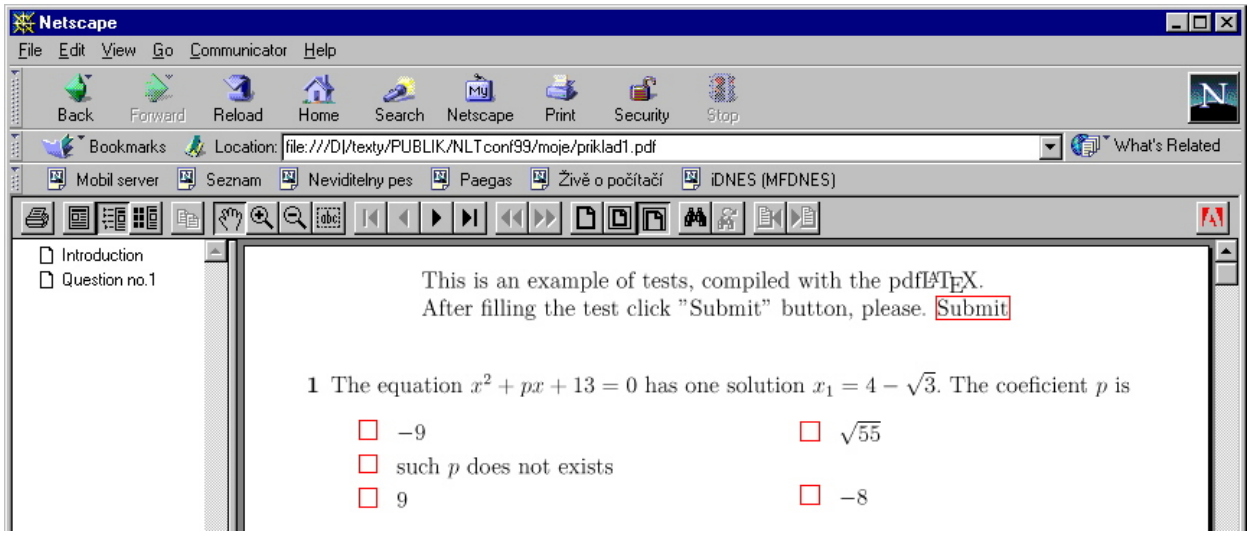


*Figure 5: Electronic form of multichoice task.*

### 2.10 *The security*

- *The user (student) must correctly enter the authorisation request. The server can accept checked requests only from fixed set of computers (classroom-only physically present student can fill up a test). It is possible to disable another net protocols for their computers and this disables communication among students. This kind of security restriction is suitable only for intranet electronic testing.*
- *Access to tests for unpermitted persons is not allowed. Tests are not saved in files. The set of tasks is placed in database tables only. Server generates them for specified list of students. The test generator inserts new record test into a database table, the test as a file can be created later, on demand. A test file in PDF format is placed on the server since student requests are deleted automatically after sending. On the client side the security guarantee is worse, but it is possible to check required web browser plug-in or hide the PDF source code.*
- *Student's questions are stored in a database table, so it is possible to restore the tests anytime in case of requirement.*

## 3 *Case study - University entry exam*

*The first usage of multichoice testing was done in autumn 1998. The university entry exam from mathematics was simulated. The approach to the testing of mathematics gives the paradigm for the other courses, no matter what kind of tests forms are available.*

### 3.1 *High school math database of tasks*

*The empty database of tasks makes no sense. The tasks creation is the long duration highly sophisticated process. At its beginning it is necessary to do statistical analysis of requirements such as: number of students, number of categories for testing, and number of answers, which are offered. The result of the analysis is the minimal number of tasks per category. That is a good starting point for task's creation process.*
*Let us focus on high school mathematics. The point of view is the university exam process. There are twenty categories of mathematical tasks, see Fig. 6. Statistical analysis says, it is sufficient to have twenty tasks in each category, if the number of applicant students is not greater than eight thousand.*

- *Theory of numbers and divisibility;*
- *sets, predicates;*
- *algebraic expressions;*
- *quadratic equations;*
- *linear equations and functions with absolute value;*
- *domains, unequations;*
- *graphs of functions;*

- *linear plane figures;*
- *trigonometric expressions;*
- *logarithmic expressions;*
- *logarithmic equations;*
- *trigonometric equations;*
- *exponential equations;*
- *analytic geometry in the plane;*
- *quadric curves;*
- *functions and sequences;*
- *geometrical sequences;*
- *arithmetical sequences;*
- *combinatorial expressions;*
- *verbal tasks.*

*Figure 6: The categories of the mathematical tasks.*

*The mathematical tasks were verified on high schools in spring 1998 and on the Faculty of Economy, Technical University of Ostrava in September 1998. For more details see [1].*

### 3.2 *Experiences with real Entry Exam*

*The IT support and mathematical tasks were successfully used at the first time for entry exam in 1999. The printed form of tests was used. This section gives the outline of useful results and remarks.*
*The number of prepared tests was 4000; twenty categories per task; one task per category; five answers per task; 3, 0, -1 points for right, none, and wrong answer respectively. Time for printing and binding of tests is approximately 40 hours for one person and Hewlett-Packard LaserJet 4000NT PostScript printer. Tests were 5 to 7 pages long, depends on a number of figures. The used disk space for tests was less than 0.5 GB.*
*Netscape Navigator 4, 12 PC based workstation with Microsoft Windows NT 4.0 was used for the distributed collecting of answers. The university network (intranet) was used. The server was Intel Pentium II 300 MHz based PC, 64 MB RAM, Debian Linux OS, Apache Web Server and MySQL database. There was enough of power of the server for 12 simultaneously working persons.*


*Figure 7: Bar code array for student's answer.*

*In September 1998 the usage of bar codes and bar codes reader was tested, see Fig. 7. The aim of testing will be the number of mistakes during the bar code reader usage in comparison with keyboard typing. The test fails because of human factor. They were problems with constant velocity of the bar code reader during process of reading. The problems seem to be caused by nonpredictable individual features of a person.*

## 4 *Conclusion*

*The given approach allows us to produce tests in both forms, the printed one, and the electronic one. The electronic test is produced by pdf$LaTeX$ and is readable by Acrobat Reader, or by web browser with Acrobat Reader plug-ins. If the conditions are satisfied, safe and effective distributed electronic testing is possible. The goal is to prepare a database of tasks for some courses given by the Department of Computer Science. The electronic tests will be used in autumn 1999 in the course "Programming language C".*
*The up to date information about the project can be found on* www.cs.vsb.cz/saloun.

This research has been conducted at the Department of Computer Science and the Department of Mathematical Methods in Economy, and was partially supported by FRVS CR grants No. 97/1475 and No. 98/0170 University

entrance computer support and collaboration with high schools" and University entrance more effectively and verification on high schools" and is currently supported by FRVS grant No. 99/543 "Safe testing on Internet".

***References***

1. *ŠALOUNOVÁ, D. - MADRYOVÁ, A. - ŠALOUN, P. - SZTURC, R. - SNĚHOTOVÁ, P. - ĎURÁKOVÁ, D.: University Entrance more effectively and verification on high schools. Final report of the FRVŠ ČR grant No. 98/0170, Ostrava, Czech Republic, 1998, 35 pages, (In Czech).*

2. *ŠALOUNOVÁ, D. - ŠALOUN, P. - SZTURC, R. - SNĚHOTOVÁ, P. - ĎURÁKOVÁ, D.: University Entrance more effectively and verification on high schools, user's handbook. Supplement of the Final report of the FRVŠ ČR grant No. 98/0170, Ostrava, Czech Republic, 64 pages, ISBN 80-85988-28-3, (In Czech).*