A case study: using multiple-choice tests at university level courses – preparation and supporting infrastructure

Marko Čupić
Faculty of Electrical Engineering and Computing
University of Zagreb
Unska 3, Zagreb, Croatia
Phone: +385 1 6129 548 E-mail: Marko.Cupic@fer.hr

Abstract - In this paper we will present a case study of using multiple-choice tests for university level courses: Digital logics and Artificial intelligence. It is a well known fact that multiple-choice questions can be used to assess student's knowledge – not only student's recall, but also the verification of student's analysis and synthesis abilities. We will give an example of questions for assessment of all three types. We will consider how computers can be used to prepare such tests, and what is needed to automatically grade such tests. We will present an open source application that we developed to foster such a process.

I. INTRODUCTION AND MOTIVATION

Knowledge assessment is an important part of each university course, especially when considering the findings of [1] that students usually distribute their time unevenly across courses, often focusing on topics associated with assessment and nothing else. There are many methods how to assess students' knowledge, some being more convenient for small courses, and others for large courses. When analyzing assessment method, we can use Bloom's Taxonomy of cognitive domains [2], which is a well-defined and broadly accepted tool for categorizing types of thinking into six distinct levels: knowledge, comprehension, application, analysis, synthesis, and evaluation. Its revision described in [3] converts category titles to their active verb counterparts: remember, understand, apply, analyze, create, and evaluate. For example, some questions for introductory geology courses for each level of Bloom's taxonomy can be found in [4].

An often-used method of assessing student's knowledge, especially for large courses, is test comprised of multiple-choice questions. Analysis of using multiple-choice tests for assessing student's understanding and review of the related literature is given in [5]. Another study to gain more insight into the relationships between students' approaches to learning and the different components of problem-solving that were being measured using a multiple choice assessment is given in [6]. Multiple-choice tests are often perceived as a method capable of testing only students recall (i.e. category knowledge in Bloom's taxonomy), which is incorrect [7]. Interestingly, according to [8], students are more likely to employ surface learning approaches in the multiple-choice question examination context and to perceive multiple-choice question examinations as assessing knowledge-based (lower levels of) intellectual processing. This is a mistake often made by students of introductory courses, and it can have negative effects on students' results when using multiple-choice question to assess not only students recall, but also their analysis and synthesis abilities.

On Faculty of Electrical Engineering and Computing (FER), University of Zagreb, multiple-choice tests gained its popularity four years ago, when curriculum was revised and aligned with Bologna process. Since then, a lot of effort was made to foster the whole process of tests preparation and grading, and to develop open source tools which could be used by broader academic community, especially in Croatia.

In this paper a case study of using multiple-choice tests for the last four years will be presented. Developed tools and gained experience will be described.

II. MULTIPLE-CHOICE TESTS

Preparation of multiple-choice tests can be extremely challenging. In this section we will present multiple choice questions which assess student's recall, application, analysis and synthesis according to Bloom's taxonomy, taken from actual tests used on course of Digital logics, to illustrate multiple-choice questions capabilities.

A. Assessing student's recall

Typical example for this kind of question is given in Figure 1.

Which of the following is standard combinatorial module?

a) counter b) register c) decoder d) flip-flop

Figure 1. Testing student's recall

This kind of questions tests if student knows and can recall specific information.

B. Assessing student's application

Typical example for this kind of question is given in Figure 2.

Find representation of number 101_{10} in base 8. Least significant digit is then:

a) 0 b) 2 c) 5 d) 7

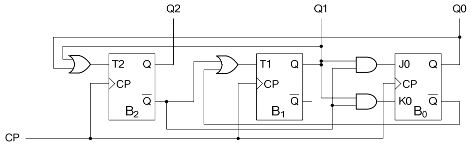
Figure 2. Testing student's application

This kind of questions tests if student knows certain algorithms, and if he is able to apply them to given problem. A care must be taken to ensure than student can not get to correct solution simply by elimination of obviously invalid solutions (e.g. if digit 9 is offered, it can simply be discarded because it is an invalid digit for octal numbers). Also, an often made mistake is to offer solutions prone to reverse checking. To illustrate this, let's say that instead of asking the student only for the least significant digit, question asked for complete number representation, and offered 4 valid octal numbers (e.g. 145, 213, 711, 351). Student not knowing the algorithm of successive division with base can easily check for correct solution if he understands the idea of number representation in different bases. In the given example it is enough to check if $1 \cdot 8^2 + 4 \cdot 8^1 + 5 \cdot 8^0$ equals 101, to accept or reject offered answer as correct. Providing only a partial solution can effectively prevent this kind of cheating.

C. Assessing student's analysis

An example for this kind of question is given in Figure 3.

For shown digital circuit find maximal clock frequency. Parameters of T flip-flops are: setup time 15 ns, delay time 30 ns, hold time 10 ns. Logical gates have delay of 5 ns.



a) 10MHz b) 20 MHz c) 25 MHz d) 50 MHz

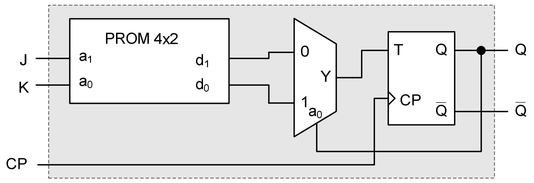
Figure 3. Testing student's analysis

To correctly answer this question, student must be able to analyze clock pulse signal propagation paths, determine the longest path and based on this analysis calculate maximal clock pulse frequency.

D. Assessing student's synthesis

An example for this kind of question is given in Figure 4.

PROM, multiplexor and T flip-flop are connected as show on picture below. How must be PROM be programmed in order to create a JK flip-flop? Answers are given as hexadecimal memory content, started from location 0.



a) 0, 1, 2, 3 b) 2, 0, 1, 3 c) 1, 0, 3, 2 d) 2, 1, 0, 3

Figure 4. Testing student's synthesis

To correctly answer this question, student must be able to synthesize required circuit. He must understand the function of PROM, multiplexor and T and JK flip-flops. He must be able to analyze how they are interconnected, and must be able to find such a programming for PROM which will ensure that the whole circuit acts as a JK flip-flop. Another example for question testing students' synthesis abilities is shown in Figure 5.

Using three T flip-flops create a synchronous counter which counts in cycle 2, 1, 0, 4, 7, 5, 6, 3. Flip-flops outputs must be counter outputs. Let FF0 represents flip-flop holding least significant and FF2 most significant output bit. Input T of flip-flop FF2 can be shown as a sum of minterms:

a) 0, 6 b) 1 c) 1, 3, 4 d) 0, 4, 5, 7

Figure 5. Testing student's synthesis

Again, to correctly solve this problem, student must understand functioning of each part, and must be able to synthesize complete circuit that obeys given specification. Since provided solution is only partial (functions of inputs of flip-flops FF1 and FF0 are not given), student can not simple reverse-engineer provided solution and find the correct one.

Our experience from Digital logic course indicates that even the slightest variations in this kind of problem can be insuperably difficult for students practicing surface learning.

III. PREPARING MULTIPLE-CHOICE TESTS

Preparation of multiple-choice tests is a three-folded process consisting of preparation of questions, preparation of test sheets and preparation of answers sheets.

Preparation of multiple-choice questions is mentally difficult and long-lasting task. For each question, wording must be carefully selected and options crafted in a way to provide just enough information to allow problem solving, but simultaneously not too much of information in order to prevent other (unwanted) ways of checking for option correctness without actually solving the problem. At the present time, this task can not be automated.

Preparation of test sheets can be either handled manually, or automated by using appropriate software. In the course of Digital logic we use manual preparation. Exams are typically composed of 15 to 20 questions. Since the questions are rather verbose and often include graphical elements (e.g. schematics), they tend to use large amount of paper space. Since we put a limit on four A4 pages (two papers) for exam, manual interventions are often required (even paper margin adjustments, question reordering and precise positioning).

Another important aspect of using multiple-choice tests is cheating prevention, since it is extremely easy to copy other student's answers. To prevent this, multiple test groups must be prepared. Here, again, there are two approaches: (1) create groups by reordering questions and/or question answers on test sheet, and (2) create groups by creating question variants, with or without question reordering.

When done by hand, first approach can become very tedious, and is extremely error-prone. For example, if Microsoft Word is used for test sheet preparation, moving a single question can often provoke nondeterministic movement of other text boxes with images around the same or even nearby pages. On the other hand, using computer-based tools for the job can be very helpful. For example, in the course Artificial intelligence we prepared test sheets by help of Enthusiast [9]. Enthusiast is a flexible tool for the automatic generation of pen-and-pencil multiple-choice test sheets. Enthusiast uses a database of multiple-choice questions and a test specification provided by the user to generate randomized multiple-choice test sheets suitable for machine-scoring. Using this tool multiple groups can be easily prepared, which can have reordered questions, reordered answers and even randomly selected question variants (if variants were entered in question database).

Second approach – creating question variants – is no less demanding. It is even more time consuming and must be performed manually. We use this approach in Digital logics. We typically prepare 4 groups, which means that for each question we create four question variants. When creating those variants, a special care is taken to ensure that all variants represent equally difficult problems. For example, when creating a variants of problem "*Find a minimal sum-of-product representation of Boolean function f=... using K-tables.*" we select four different Boolean functions which have in minimal form equal number of products. This effectively means that during question preparation each question must be created and solved at least four times (sometimes, a number is even larger if problem variants difficulty is not the same from the first attempt).

Finally, preparation of answers sheets will enable tests to be machine-scored. Answers sheet is comprised of machine readable student designation (e.g. bar-coded student identifier), machine readable group designation (e.g. bar-coded group or matrix where student can blacken given group) and machine readable answers matrix, where student can for each question blacken the answer he thinks to be the correct one. Additional information can also be provided on answers sheet, such as examination room designation (for exams taken in multiple rooms simultaneously), student's full name and student number in that examination room. Preparation of such answers sheets, for mass courses will imply a lot of work, starting with creation of a schedule for students and rooms, making that schedule public, and finally creating answers sheets with all of the required data. This can be handled manually or using appropriate software. For example, to prepare answers sheets manually, a template can be created using Microsoft Word and by installing special required fonts which will provide functionality to create barcode and answers sheets. Student schedule can then be prepared as formatted textual file containing a line for each student. This file can then be used by Microsoft Word MailMerge add-on, which can read that file and for each student based on the previously prepared template create a new document containing answers sheets.

Alternative approach is to use a software offering answers sheets preparation. Example of such a software is Ferko [10] – which is an open source course management system, designed and implemented together by faculty staff and faculty students. This system enabled us to easily create student schedules and publish them privately into students' personal calendars, and to prepare answers sheets and complete exam administration in several mouse clicks.

Many times we have heard a question – does it pay off to prepare multiple-choice tests? Computer scoring grading as opposed to human scoring is a promise we all like, but is there any savings in time? Preparing classical test with 10 open-ended questions is a much easier task. However, in courses with several hundred students or even more than a thousand students, human grading is time consuming, subjective, tedious and error-prone job which requires several people (or even one person per problem) to grade the exam in any reasonable time. What is the cost of preparing multiple-choice tests?

In Digital logics, test sheets preparation is a three-step process. First, a single person creates a test proposal. Test proposal is pencil-on-paper draft containing all of the questions and answers (15 or 20 questions), as well as a hint indicating which parameters be varied. To create this draft, a single working day is required (8 h). This draft is then revised and modified by two to three course lectures (approximately 1 h). Step 2 is the creation of four test variants, which includes creation of four variants for each question, drawing all schematics and required images for each version of question (typically using Microsoft Visio) and finally creation of test documents (in this case using Microsoft Word) for all test groups. Usually, step 2 is handled also by a single person and takes two working days (8 h + 8 h). Finally, when groups are created, at least two test groups are solved by each of three lecturers (it takes about 1 h per group to carefully read questions, check that there is no imprecise phrasing and no duplicate answers etc). This accounts for another 6 h of work. Finally, if any problems are found, they are corrected (another 1 h). Additional exam administration takes another 1 h. Tests are then multiplied. Since we prepare two-paged tests, groups can not be auto shuffled during multiplication, so this also has to be done by hand. Tests must be divided for each examination room. For a thousand students this takes about 3 h. And finally, machine-scoring and results publication for a thousand of students takes another 2 h. This totals to 40 h invested into test preparation and scoring.

To create a test comprised of 10 open-ended questions it takes about 1 h. To score it manually for thousand students, it requires approximately 10 persons each scoring one of those 10 questions for 2.5 working days (20 h). This totals 200 h. So, multiple-choice tests with one and a half to two times more questions require only 20% of time. However, it is important to notice the distribution of time. When preparing multiple-choice tests, total invested amount of time is practically a function of number of questions; a number of students here is almost of no importance, since tests are machine graded. Tests with open-ended questions present exactly reversed situation. Majority of invested time goes to human scoring, which is a function of both: number of questions and number of students. From this analysis it follows that tests with open-ended questions are cheaper for small to moderate number of students, while multiple-choice tests are cheaper for larger courses.

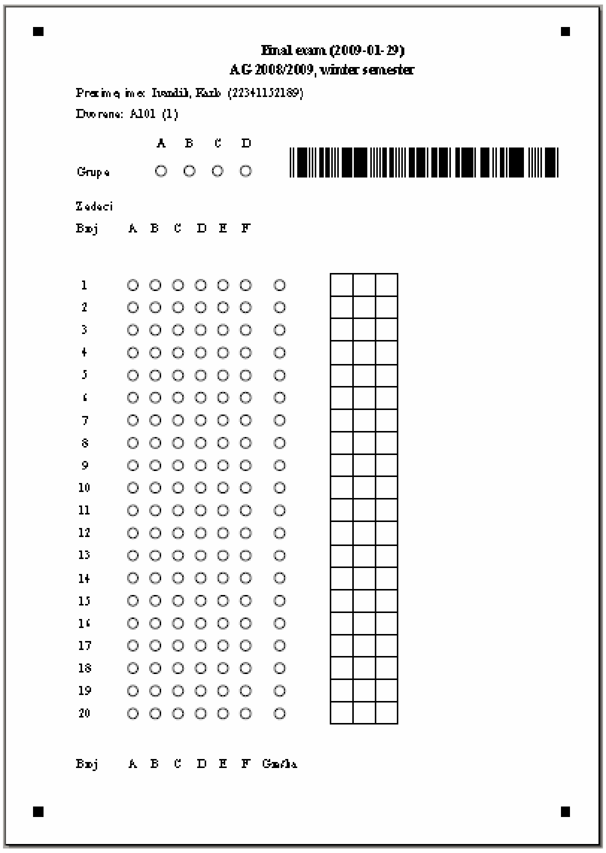
There is also one aspect that we haven't addressed so far that is important from student's point of view: how fast can they get feedback after the exam is done, and how objective is the scoring process? Both of those questions are immensely important for students, and they both give significant advantage to multiple-choice tests.

IV. MULTIPLE-CHOICE TESTS SCENARIOS

On Faculty of Electrical Engineering and Computing, multiple-choice tests are used in majority of courses. Since we have been directly involved in 5 courses, here we will summarize different usage scenarios.

A. Classical multiple-choice tests

Most frequently test sheets and answers sheets are prepared as separate documents. In this case, answers sheets can be prepared either manually, or by specialized software. Example of an answers sheet prepared by Ferko is show in Figure 6. This approach was used for mid-term exams and final exams in the following courses: Digital logics, Interactive computer graphics, Artificial intelligence and Scripting languages. In course Computer architecture 2, exams were composed of theoretical part (by multiple-choice tests) and problems (open-ended questions scored by hand).



Final exam (2009-01-29)
AG 2008/2009, winter semester

Prezime, ime: Ivandil, Karlo (22341152189)

Dvorana: A101 (1)

	A	B	C	D
Grupa	○	○	○	○

Zadaci

Broj	A	B	C	D	E	F	Greška
1	○	○	○	○	○	○	○
2	○	○	○	○	○	○	○
3	○	○	○	○	○	○	○
4	○	○	○	○	○	○	○
5	○	○	○	○	○	○	○
6	○	○	○	○	○	○	○
7	○	○	○	○	○	○	○
8	○	○	○	○	○	○	○
9	○	○	○	○	○	○	○
10	○	○	○	○	○	○	○
11	○	○	○	○	○	○	○
12	○	○	○	○	○	○	○
13	○	○	○	○	○	○	○
14	○	○	○	○	○	○	○
15	○	○	○	○	○	○	○
16	○	○	○	○	○	○	○
17	○	○	○	○	○	○	○
18	○	○	○	○	○	○	○
19	○	○	○	○	○	○	○
20	○	○	○	○	○	○	○

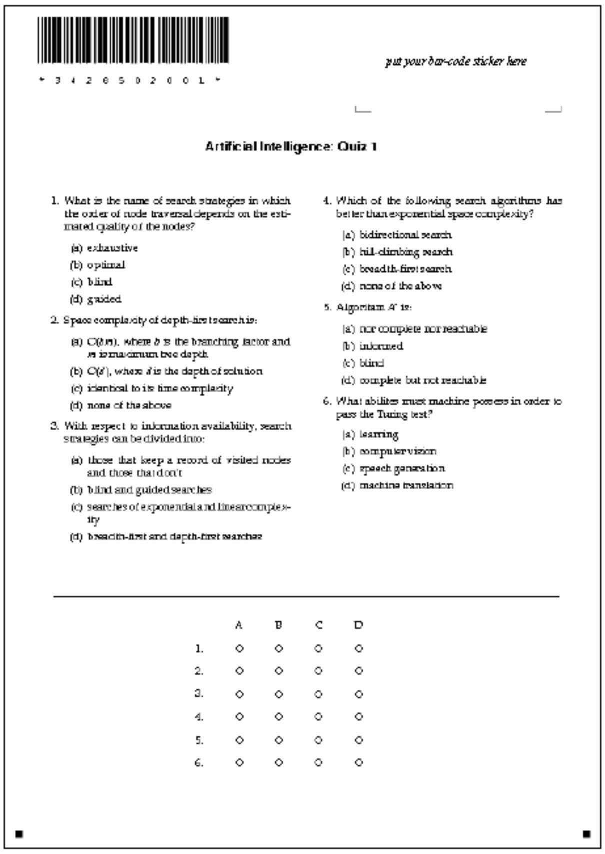
Figure 6. Stand-alone answers sheet prepared by Ferko

Multiple-choice tests were also employed in Scripting languages and Computer architecture 2 for laboratory exit-exams.

B. Integrated test and answers sheets

In two courses we wanted to more frequently assess students' knowledge, as well as to provide them more quickly with feedback. We introduced additional five-minute tests as a part of course lectures. Two challenges arose: (1) how to prevent cheating when students were sitting next to each other in lecture room, and (2) how to provide each student his own personalized answers sheet in such a short time?

To answer first challenge, we used Enthusiast software, to create as many groups as there were students by selecting provided question variants and reordering questions. Additionally, we integrated test sheet and answers sheet into a single document, as shown in Figure 7. Each test had its group bar-coded in upper-left part of the page. Since we decided early to make six to nine short assessments during semester, we prepared for each student nine stickers with their student identifier bar-coded, and handed them to students at the beginning of semester. Students were advised always to carry them to classes. Having tests prepared in this way, we were able to randomly divide tests to students, they would simply put their bar-code stickers to designated place on the test sheet and begin solving the test – all in no more than seven minutes. Tests were scored and results published to students approximately one hour later. First year, bar-code stickers were created by hand-written LaTeX document; however, this is now also provided by Ferko with two mouse clicks.



* 3 4 2 6 5 0 2 0 0 1 *

put your bar-code sticker here

Artificial Intelligence: Quiz 1

1. What is the name of search strategies in which the order of node traversal depends on the estimated quality of the nodes?
   (a) exhaustive
   (b) optimal
   (c) blind
   (d) guided
2. Space complexity of depth-first search is:
   (a) $O(bm)$, where b is the branching factor and m is maximum tree depth
   (b) $O(b^d)$, where d is the depth of solution
   (c) identical to its time complexity
   (d) none of the above
3. With respect to information availability, search strategies can be divided into:
   (a) those that keep a record of visited nodes and those that don't
   (b) blind and guided searches
   (c) searches of exponential and linear complexity
   (d) breadth-first and depth-first searches
4. Which of the following search algorithms has better than exponential space complexity?
   (a) bidirectional search
   (b) hill-climbing search
   (c) breadth-first search
   (d) none of the above
5. Algorithm A^* is:
   (a) nor complete nor reachable
   (b) informed
   (c) blind
   (d) complete but not reachable
6. What abilites must machine possess in order to pass the Turing test?
   (a) learning
   (b) computer vision
   (c) speech generation
   (d) machine translation

	A	B	C	D
1.	◇	◇	◇	◇
2.	◇	◇	◇	◇
3.	◇	◇	◇	◇
4.	◇	◇	◇	◇
5.	◇	◇	◇	◇
6.	◇	◇	◇	◇

Figure 7. Integrated test and answers sheet prepared by Enthusiast

In course Interactive computer graphics, we decided to have three short assessments during semester, so we did not prepare bar-code stickers. Since the number of students was relatively small (less than 200), we decided that it is acceptable to have semi-automatic sheet recognition, where software could perform all except student identifier recognition – human operator entered this for each sheet when the program asked for it.

V. ANSWERS SHEET PROCESSING

After exam is done, all of the answers sheets must be scanned, information from those scans must be extracted (student identifier, test group and blackened answers). In all five courses, this task was handled by two open-source applications we developed: FreeFormScanner [11] and FreeFormReader [1]. FreeFormScanner is a Java based desktop application which can communicate with scanners, and perform digitalization of answers sheets. We take 256 shades of gray scans at 200 dpi. Currently, application can communicate with scanners only in Microsoft Windows since we implemented support for TWAIN interface. Additional effort is required to create appropriate interface that would allow communication with scanners on other operating systems (for example, SANE interface on Linux).

After the scanning process is done, scans are processed by FreeFormScanner. This is also a Java based desktop application we created for extracting information from scanned answers sheets. The application offers a set of recognizers – components specialized in recognition of various information encodings (e.g. bar-coded strings, answer matrix, group matrix). To process scanned answers sheets, a new project must be created, and a template must be added. Any scanned answers sheet can be used as a template. On that template, recognizers are added and configured. After the template is done, all scans are added into project and information extraction is started. During this process, human operator intervention can be required – for example, when question is allowed to have only one answer and student selected more than one option. This occurs often when student blacken one option by mistake, and then blacken additional. To foster this kind of correction processing, we added to answers sheet letter table (see Figure 6) on the right side of answers matrix. In this table, student can write letter representing the option he deems correct. An effort is made to implement a recognizer that could automatically read those corrections and so additionally speed-up information extraction. Also, effort is made to implement additional types of recognizers that would enable efficient and reliable encodings of student identifiers created by students at the time of exam (experiments with promising results were made with 7-segment display-like template where students would blacken segments of each number to create correct representation of theirs identifiers).

Both of described applications have been tested and successfully used for several years. For example, the usage of FreeFormReader started in academic year 2006/2007. Its usage is summarized in Table 1.

Column "Exams" in Table 1 contains a number of multiple-choice tests that were created and processed in specific instance of course as a part of mid-term exams and final exams. Column "Lab" contains a number of multiple-choice tests created and processed for laboratory exams, and column "5-min" contains a number of multiple-choice tests created and processed for short lecture examinations.

TABLE I
Usage of FreeFormReader

Year	**Students**	**Exams**	**Lab**	**5-min**	**Total**
Digital logics					
0607	959	2859	0	0	2859
0708	835	2536	0	0	2536
0809	827	2361	0	0	2361
Scripting languages					
0708	141	371	357	0	728
Interactive computer graphics					
0708	172	252	0	284	536
Artificial intelligence					
0708	94	266	0	454	720
Computer architecture 2					
0809	101	257	279	0	536

To additionally foster processing of multiple-choice questions, FreeFormReader has a built-in module for information export into two web applications used for results distribution at our Faculty (one of which is Ferko), as well as module for automatic grading (provided that correct answers and grading policy are specified).

VI. CONCLUSION AND FUTURE WORK

In this paper we have presented a case study of using multiple-choice tests at university level courses. We have described the successful usage of multiple-choice tests at five courses for various kinds of knowledge assessment: midterm and final exams, laboratory exercise exams and short lecture assessments.

We have illustrated that multiple-choice questions can assess not only students recall, but also higher skills, such as analysis and synthesis capabilities, according to Bloom's taxonomy. We have illustrated the process of preparing test and answers sheets, with respect to computer supported tools available to foster this process. Various scenarios in which multiple-choice tests can be used were described, and implications to design of answers sheets elaborated. Finally, open-source tools for answers sheet digitalization and processing were described.

As a part of future work we plan to develop an integrated solution and publicly available web portal that will enable broader academic community easier preparation and usage of multiple-choice tests, by offering instructions and needed open-source tools.

REFERENCES

[1] G. Gibbs and C. Simpson, "How Assessment Influences Student Learning: A conceptual overview", SSRG, 42/2002, Student Support Research Group, Centre for Higher Education Practice, Open University UK, 2002. Retrieved February 07 2009 from: http://www2.open.ac.uk/cehep/ssrg/reports/documents/42_02.pdf

[2] B. S. Bloom, D. R. Krathwohl and B. B. Masia, "Taxonomy of Educational Objectives: The Classification of Educational Goals", New York, NY: D. McKay, 1956.

[3] L. W. Anderson, D. R. Krathwohl, and B. S. Bloom, "A Taxonomy for Learning, Teaching, and Assessing a Revision of Bloom's Taxonomy of Educational Objectives", New York, NY: Longman, 2001.

[4] D. A. McConnell, D. N. Steer and K. D. Owens, "Assessment and active learning strategies for introductory geology courses", Journal of Geoscience Education, vol. 51, p. 205-216, 2003.

[5] M. G. Simkin and W. L. Kuechler, "Multiple-choice tests and student understanding : what is the connection?", Decision Sciences Journal of Innovative Education vol. 3, no. 1, p. 73-97, 2005.

[6] F. J. R. C. Dochy, D. Gijbels and G. van de Watering, "The relationship between students' approaches to learning and the assesment of learning outcomes", European Journal of Psychology of Education, vol. XX, no. 4, p. 327-341, 2005.

[7] K. Woodford and P. Bancroft, "Multiple choice questions not considered harmful", ACE 2005. Australian Computer Society, 2005.

[8] K. M. Scouller, "The influence of assessment method on students' learning approaches: Multiple choice question examination versus assignment essay", Higher Education, vol. 35, no. 4, p. 453-472, 1998.

[9] J. Šnajder, M. Čupić, B. Dalbelo Bašić and S. Petrović, Enthusiast: An authoring tool for automatic generation of paper-and-pencil multiple-choice tests, ICL 2008, Villach, 2008.

[10] M. Čupić, Ferko – A Course management system,
URL: https://ferko.fer.hr/ferko

[11] M. Čupić, Free Form Scanner,
URL: http://morgoth.zemris.fer.hr/FreeFormScanner

[12] M. Čupić, Free Form Reader,
URL: http://morgoth.zemris.fer.hr/FreeFormReader