CHANGING THE LECTURING STYLE – THE GOOD AND THE BAD OF MIXED-UP SCHEDULES

Cs. Szabo[*] and A. Bollin[**]

[*]Department of Computers and Informatics, FEEI, Technical University of Košice, Slovak Republic
[**]Software Engineering Research Group, University of Klagenfurt, Austria
Csaba.Szabo@tuke.sk, Andreas.Bollin@aau.at

Abstract - How does a teaching block impact on students, who are used to 13-week semester subjects? Do these students have better knowledge immediately (i.e. one week) after such a teaching block compared to the case of the usual setup, where they also wait one month until the exam? We had the chance to try out such a change in the course setting within one Software Quality and Project Management course at the Technical University of Kosice. The course is located in the last year of our Informatics and Applied Informatics Masters' degree study programs. This paper presents our teaching experiment focusing on the questions presented above, together with its results. First, we present details of the mixed-up semester and block schedules. Then, we analyze time constraints, relations to other subjects, various students' problems, workload on students and teachers, and grading issues. We also evaluate the collected data and opinions, and discuss students' feedback related to this specific course organization. The presented conclusion focuses on the future application of the teaching schedules used as well as on improvements of these schedules.

I. INTRODUCTION

Teaching at universities is always a big challenge. The bigger the challenge is the faster the development and innovation is in the given field. Software engineering is a characteristic example of such a field [1, 2]. In our setting, software quality assurance and software project management represent such a teaching challenge.

Modern trends in teaching include, apart from the use of E-Learning systems and web-environments, the usage of examples [3], case studies [4], and simulations [5-8] for the following reasons:

1. Examples are one source of basic knowledge to be gained.
2. Case studies require more concentration than examples from the students to understand a process.
3. Simulations allow for the most complex type of knowledge transfer and interaction.

Instead being deductive as in the cases 1 and 2 above, simulations require predictive thinking. There is a significant positive property of teaching by using simulations: the knowledge stems from the students' own experiences. Additionally, the language used within simulation environments is less a problem, especially when compared to using foreign language teaching materials. Mária Šimková et al. argue that such materials have unwanted bad impact on native language skills of students [9]. Often, parts of these teaching materials come from industry or an inter-University cooperation, which implies also a jargon to be used – yielding another language problem and introducing inconsistencies in the used terms. Finally, experience at the Technical University of Košice (TUKE) also shows that some students still prefer cheating [10], and thrilling simulations can be seen as a chance to increase the interest in learning again.

Besides the teaching environment, the most important issue is the student himself/herself. In our research, we aimed at measuring their results [12], and adopted teaching materials, as other authors [11-15] do, to consider the students' knowledge but also to provide the required level of new knowledge to them.

Further characteristics of a teaching environment include social and technical backgrounds and habits. Here, we consider e.g. lengths of teaching and examination periods, grading and evaluation requirements, learning material styles etc.

We dedicate this paper to the memory of Ladislav Samuelis (1951-2013), the promoter of teaching software quality in Slovakia.

Many thanks go to Elke Hochmüller from the Carinthia University of Applied Sciences in Klagenfurt for helping and supporting course implementation.

This work is the result of implementation of the research and development cooperation project No. SK-AT-0024-12: "Advanced Software Engineering Education – Methods and Tools (AdSEE)."

This work was also supported by the Cultural and Educational Grant Agency of the Slovak Republic under project No. 050TUKE-4/2013: "Integration of Software Quality Processes in Software Engineering Curricula for Informatics Master Study Programme at Technical Universities – Proposal of the Structure and Realization of Selected Software Engineering Courses."

In this paper, we present the results of an experiment where we tried out a radical change in the setup of our teaching environment at TUKE. The main question was if there is an effect (and of which kind) when introducing a tight teaching block instead of following a traditional course setup on a weekly basis. We stated two hypotheses related to this question:

1. The teachers' and students' workload does not change when changing the implementation of the subject schedule from the 13-week semester model (lectures and labs) to a mixed-up model consisting of 13 weeks of lectures and a one-week long teaching block of intensive laboratory work.
2. The students' knowledge (measured by tests at the end) is higher (when compared to the past year's results) if they take place immediately in the week after the execution of the intensive teaching block.

In order to answer these questions, we examined changes in the students' behavior and workload as well as in the teachers' workload. The details on the environment changes will be presented in Section II of the paper. In Sec. III, we present and discuss selected problems with its implementation, while our student-behavior related experience is discussed in Sec. IV. In Sec. V, we evaluate our hypotheses and show further directions of our research.

II. LECTURES & LABS IN OUR EXPERIMENT

For our experiment, a course called Software Quality and Management was selected. This course takes place in the last year of our Informatics and Applied Informatics Masters' degree study programs. During our experiment, 148 students of the said specializations attended the course. In the year before our experiment, there have been 140 students attending the class. In the traditional 13-week semester setup, lectures and labs were taught by one teacher.

The theory required for the course is presented at lectures, while labs use a simulation tool for practicing software project management. The theory is also accessible in a printed form as a textbook [16]. For the simulations, an environment called AMEISE (A Media Education Initiative for Software Engineering) [6-8] was used.

The basis for an AMEISE simulation run is a so-called simulation model. It contains the different simulation settings. In our course, we made use of a quality assurance model which focuses on quality aspects, requiring the trainee to manage a 200 AFP (Adjusted Function Point) project within 9 simulation months, a budget of 225.000 €, and strict requirements concerning the quality of the final product (in terms of a maximal number of errors allowed per 1000 lines of code and a minimum percentage of AFPs required to be implemented).

A. Structure of the 13-week semester subject

The "classical" semester TUKE students are familiar with lasts 13 weeks, followed by a 6 weeks examination period. The organization of a typical semester is as follows:

- Every week there is a lecture. There are 11 or 12 lectures depending on the actual year (as there could be a lecture cancelled due to a national holiday or a conference). This also means that the lecture content is adapted to the situation – in some cases the lecture is more condensed with a shorter time reserved for discussion. The length of a lecture is 90 minutes, i.e. two lecture hours.
- Every week a lab. There is the same number of labs and lectures that take place. Lab classes are only partially related to the actual week's lecture. The main focus of the labs is on practicing software project management skills. Basically, two simulations in the AMEISE environment have to be done by each student team. Such teams usually consist of two members; in the case of an uneven number of students, one team consists of three students. The duration of the labs is also 90 minutes, i.e. two lecture hours. As the duration of one AMEISE simulation run is more than a usual lab in the schedule, some parts of the simulations are to be completed at home (summing up to about 3 extra hours in total).

To conclude, the average workload of a student is 4 lecture hours per week (12 times), summing up to 48+3 = 51 lecture hours per semester.

The lecturer has to work more, not only because of material preparation, but also because the students are from two fields of study. On average, it needs about 48 lecture-hours for preparation, giving the lecture and after-lecture work. Due to the high number of students in our experiment, 6 groups had to be formed in the lab classes. The lab workload for the teacher to be taken into account is thus 6 times 24 hours. It results in 12 lab-hours per week or 144 teaching hours per semester of workload for the teacher.

Grading is done at two stages: finishing an AMEISE simulation run successfully yields 6

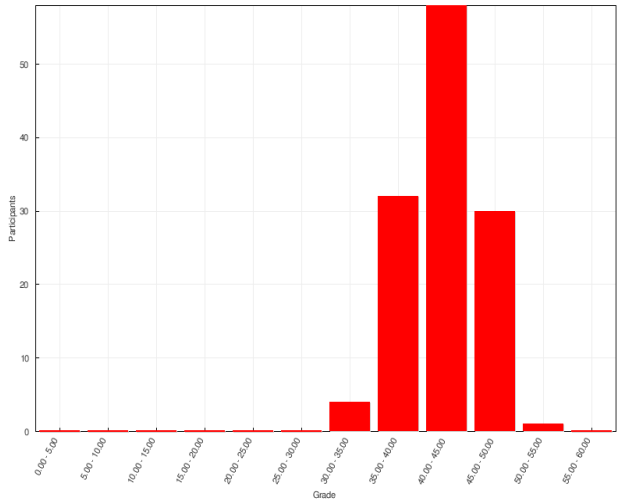

Figure 1. Course exam results from the 13-weeks variant

ECTS credits and the exam result determines the final grade at our local A to FX scale. The weight distribution is 40% and 60%; obviously, 21% of 40% for the credits respectively 31% of 60% are needed to pass the exam and are thus the minimal requirements in the ECTS system of the subject.

The examination results for a typical 13-week semester subject are displayed in Fig. 1. It presents the results from the academic year 2012/2013 in the mentioned subject.

B. Structure of the mixed-up schedule

The mixed-up schedule combines the "classical" 13-week semester of lectures with a teaching block at the end of the semester. Due to the limitations in the organization of teaching subjects at our department, this combination had to be implemented at the beginning of the examination period – further details on problems and specific solutions will be presented in Sec. III.

The key point of the schedule is that labs are organized in a completely different way: they are scheduled in dense blocks. The lectures stay the same and the majority of them regularly takes place during the semester. Only those lectures (and parts of it) which are needed for successfully working in the labs are also shifted to the blocked week. The layout of the lectures is as follows:

1. The first four lectures are moderated guest lectures from the Testing headquarter at a big Company. Topics of these lectures include software and test-aware project management and software testing.
2. Next, three lectures on software metrics follow.
3. The remaining lectures focus on software quality management, on actual research results and challenges in all discussed fields of software engineering.

The above lecture organization is almost identical to the 13-week schedule. Therefore, we can provide nearly the same amount of effort:

- 24 hours per semester for students, and
- 48 hours per semester for a single teacher. However, the moderation of guest lectures might be of a lower workload than preparing for own presentations.

The introduction of a teaching block is something new at TUKE. The aim is to deepen the students' knowledge intensively using simulations immediately after that lecture which introduces the simulation topic. How does it work? Well, we created the following schedule:

1. Intensive introductory lectures on software quality metrics and software project management,
2. Even more intensive lecture introducing the AMEISE simulation environment,
3. First simulation runs,
4. Feedback session on first simulations,
5. Second simulation runs,
6. Feedback session on second simulations.

To increase the students' motivation, a best simulation award was defined for the first simulation runs.

The workload measurement results for the students partaking in the experiment can be summed up to 23 hours per block. The teachers' workload considering 6 groups (as above) sums up to 83 teaching hours. This is a problem, because this would not fit into a normal one-week block. Therefore, we decided to involve 3 teachers (again, details will follow in Section III.C) in the lecture.

III. PROBLEMS & SOLUTIONS

Our final schedules looked like as presented in Fig. 2. However, it was a long way to create them. Our way was impeded by a lot of obstacles – which ones, we now present in the next sub-sections.

A. In which week should the block be?

This was (and still is) a very important question, also related to the workload of students and teachers.

(a)

Green Group (TG2) – Schedule

Date	Time	Room	Course Unit / Homework	Comment/Deadline
13.1.	12:30-15:00	BN5-ZP1	AMEISE Introduction	
13.1.	16:00-18:00	L9-A_537	Project Planning 1	Moodle DL: 13.1., 20:00
14.1.	09:00-13:30	L9-A_537	Simulation Run 1	AMEISE DL: 14.1., 13:30
14.1.	16:00-18:00	L9-A_537	Self Evaluation 1	Moodle DL: 14.1., 20:00
15.1.	09:00-11:00	BN5-ZP4	Feedback Session 1	
			HW: Project Planning 2	Moodle DL: 15.1., 15:30
15.1.	15:00-19:00 15:30-19:30	L9-A_537 L9-B_515	HW: Simulation Run 2	rooms optional, AMEISE DL: 15.1., 24:00
16.1.	12:00			Assessment Reports 2 available in Moodle
			HW: Self Evaluation 2	Moodle DL: 16.1., 15:30
16.1.	16:00-18:00	BN5-ZP4	Feedback Session 2	

(b)

Yellow Group (TG1) – Schedule

Date	Time	Room	Course Unit / Homework	Comment/Deadline
13.1.	10:00-12:30	BN5-ZP1	AMEISE Introduction	
			HW: Project Planning 1	Moodle DL: 14.1., 08:00
14.1.	08:30-12:30	L9-B_515	Simulation Run 1	AMEISE DL: 14.1., 13:30
			HW: Self Evaluation 1	Moodle DL: 14.1., 20:00
15.1.	09:00-11:00	BN5-ZP4	Feedback Session 1	
			HW: Project Planning 2	Moodle DL: 15.1., 15:30
15.1.	15:00-19:00 15:30-19:30	L9-A_537 L9-B_515	HW: Simulation Run 2	rooms optional, AMEISE DL: 15.1., 24:00
16.1.	12:00			Assessment Reports 2 available in Moodle
			HW: Self Evaluation 2	Moodle DL: 16.1., 15:30
16.1.	16:00-18:00	BN5-ZP4	Feedback Session 2	

(c)

Orange Group (CG) – Schedule

Date	Time	Room	Course Unit / Homework	Comment/Deadline
13.1.	10:00-12:30	BN5-ZP1	AMEISE Introduction	
			HW: Project Planning 1	Moodle DL: 15.1., 20:00
16.1.	08:30-13:30	L9-A_537 L9-A_509	Simulation Run 1	AMEISE DL: 16.1., 13:30
17.1.	14:00-16:00	L9-A_509	Feedback Session 1	
			HW: Project Planning 2	Moodle DL: 17.1, 24:00
			Simulation Run 2	AMEISE DL: 19.1., 24:00
?	?		Feedback Session 2	by Csaba Szabó

Figure 2. Final green (a), yellow (b) and orange (c) group schedules for the week with AMEISE

The usual semester organization at TUKE (but also other Universities) prescribes that every student must have a lesson every week in every of his teaching subjects. This implies a huge complication when organizing a one-week teaching block. Students cannot be taken off from their usual schedule.

As a solution, we used the second week of the examination period for our block. However, with that, another problem appeared – there were possible collisions with exams in other subjects this time.

We solved the second problem by reorganizing students into larger groups: green, yellow and orange. This affected the schedule as well, but could decrease the number of teachers' hours due to a smaller number of groups.

B. How to grade the students?

Grading should be a transparent process in teaching. Fortunately, there are several attributes reported back as the result of a simulation run. In the simulation setting a software system had to be "produced", and the following attributes

(simulation goals) were used as a basis for the grading:

1. Duration of the project (in simulation time),
2. Costs (needed for delivering a product),
3. Number of Adjusted Function Point covered by the code,
4. Number of errors per 1000 lines of code,
5. Adjusted Function Point % covered by the manuals,
6. Number of errors per page in the manuals.

We defined lower limits of success, but as there have been two simulation runs, the most important factor was *how* the teams *improved* their simulation results. In order to pass this part of the lecture, we defined that at least in one of the above factors an improvement was required.

We used the reporting interface of the AMEISE server to retrieve all the relevant data. Then, we put these data into a Spreadsheet application, applied our limits, and summed up the results for each team separately.

To make it easier for the teams to improve, the first feedback session included personalized suggestions on how to do that.

C. *How to decrease workload?*

Our strategy in decreasing the workload was to reorganize groups and their schedule. From the first guessed value of 24 hours workload on students, the final organization using only three groups instead of six served with 23 hours workload. In the case of the experiment, the involved teachers managed to decrease the value from 83 to 40 teaching/lab hours.

D. *In which week should be the exam?*

According to the intensive block teaching, the question of the examination date was also important. Long-time memory and short-time memory effects play a major role in this case. And, the different learning styles of students also have to be considered. Results from brain-friendly teaching [17] indicate that using simulations (and other activities) in teaching already yields an optimal result and a lot of knowledge is moved from short to long-term memory. This means that the exams can be at the end of the block.

Additionally, the mixed-up schedule supports for both, short and long-term memory support. Almost every theoretical lecture took place one month (or longer) before, and then the simulation block repeats and demands the gained knowledge again.

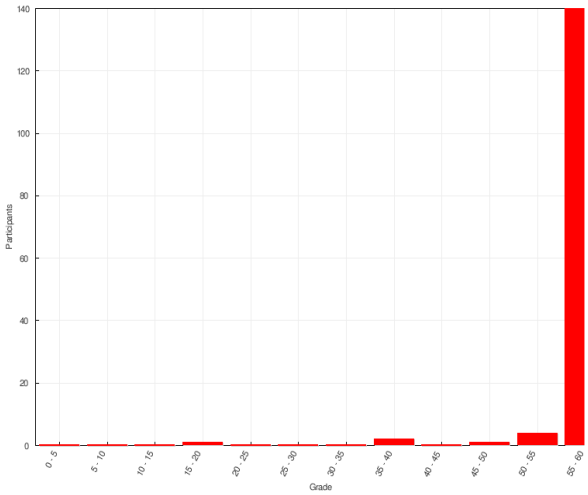

Figure 3. Course exam results from the intensive examination week after the mixed-up schedule

In our case, we defined three examination days (with 9 exams) in the week after the lectures' block. Another three exams were defined for the next three weeks of the examination period. But, with the exception of one student, all students passed the exams during the intensive week. The examination results for the intensive examination week are shown in Fig. 3.

E. *Technical problems & solutions*

As usual, technical problems also appeared. The most significant issue was the AMEISE server that was located at the Alpen-Adria University in Klagenfurt, Austria. The clients were connected to the server via the wireless computer network at TUKE and produced too much traffic. The bottleneck caused response time problems in all the clients. And, this unstable connectivity, as a consequence, also slowed down the AMEISE server and led to inconsistencies between the clients and the server. As a solution, we will use a local server in the future and improve/update the client software setting to limit unnecessary network traffic.

Another technical problem was the capacity of rooms available for the labs. Distributing the groups over several labs solved this problem, but also complicated the schedule (and the presence of the teacher) in several points.

IV. STUDENT FEEDBACK

We collected qualitative feedback from the students during the execution of the mixed-up schedule systematically using questionnaires. Also, follow-up feedbacks were gathered, but in a less systematic way: students were allowed to report on their feelings about the schedule after the exams verbally, and, last but not least, indirect feedback was collected by reading student forums

and by asking students (selected randomly) who were not enrolled to the subject.

A. Immediate feedback

As immediate feedback we consider feedback gained during the simulation sessions, i.e. before the end of the teaching block. This feedback was manifold; negative feedback is that several students had language or technical problem.

However, what can be seen as a great success: the vast majority of the students welcomed the new kind of organization and gave overall positive feedback on the system, guiding activities and amount of new knowledge gained during the blocked lecture.

B. Delayed feedback

Delayed feedback is given not later than two weeks after the teaching block finished. This includes feedback presented at the exams or informal feedback from several students.

The results of the evaluation show that there is mostly positive feedback again. One possible reason of missing negative feedback might be that we only asked students who already passed the exam positively. We know that these results are subjective, but the willingness of students to give a feedback is already an achievement, and to some extent a success of the teachers.

C. Late feedback

Late feedback is a feedback received later than two weeks after finishing the teaching subject. Here, no students were directly asked. But, by looking at students' forums, information was collected regarding to students' recommendations (to other students) on the Software Quality and Management course.

These types of recommendations are statistically incomplete, but as human factor evaluations, they reflect student satisfaction. We hope, this satisfaction remains and will not be teacher-related. The name AMEISE gained a positive meaning between our students, and in total 31 students had a very positive attitude to the course. But, we also know about 3 students who expressed a strong negative feeling about it. Other less negative feedback came from a very small group of students – about a size of 10 students.

V. RESULTS & CONCLUSION

The most significant lesson that we learned is that a mixed-up schedule implies a huge hidden workload on the organization, especially when it is un-natural to our teaching environments/settings. On the other hand, it represents a new and fresh point in a student's life – a change that electrifies him or her. This motivation is really needed in the all-days life.

A blocked schedule is, due to the facts presented in Section II.B, hard to improve. The strict and classical organization of lecture hours at most of the Universities yields problems in finding proper dates during the semester and the situation cannot be solved without changing local policies in Universities' teaching. One way out might be the organization of winter or summer schools for extra credits.

Finally, and based on our personal experience with the lack of space in small laboratories, the extension of the three-teacher team by additional members seems to help when running into a room capacity problem. In the case of the experiment, a fourth teacher would have been needed in order to ensure two guiding teachers per lab.

Looking at the findings above, our first hypothesis is not confirmed. The workload in such a un-natural setup is higher than in the case of a 13-week semester.

The evaluation of the second hypothesis is more difficult. The feedback sessions were obviously the most powerful sources of knowledge. Students were able to learn from their own mistakes as well as from the mistakes of the others. We claim that these sessions caused the positive feedback on the course organization at all.

The mixed-up schedule showed up as a very good solution regarding to the short-time and long-time memory of students. Our students had to use both types of their knowledge at the exam. The results suggest that it also increased their effectiveness in comparison to the previous year's results on the same question bank. Looking at Fig. 1 and Fig. 3, cheating might still be an issue, and an improvement of our question database might be needed.

So, if we do not consider possible cheating, then the results are amazing, and we can state that the hypothesis is proven to be true.

Another lesson learned is that language does really matter even in software engineering education – there are students who claim problems with English (e.g. one issue in the feedback is that a lecturer was speaking too fast). Probably, and due to the fact that the common teaching language is Slovak, we will never have students with equal and very good English language skills at TUKE, but lectures given by external lecturers are one way in improving the situation a bit.

In the end, our results should encourage everyone to try out new settings. The way of

learning is changing, and we also should think about changing our lecture style. Of course, one will run into good and bad issues, but at the end of the day increased knowledge and increased motivation is something we might harvest from it.

REFERENCES

[1] L. Samuelis, "Notes on the emerging science of software evolution." In: Handbook of Research on Modern Systems Analysis and Design Technologies and Applications. - Hershey : Information Science Refernce, 2008 P. 161-167.

[2] L. Samuelis and Cs. Szabó, "On the role of the incrementality principle in software evolution." Egyptian Computer Science Journal. Vol. 29, no. 2 (2007), p. 107-112.

[3] M. Sabo, J. Porubän, D. Lakatoš, and M. Kreutzová, "Computer Language Notation Specification through Program Examples." In: FedCSIS : Proceedings of the Federated Conference on Computer Science and Information Systems : September 18-21, 2011, Szczecin, Poland. - Los Alamitos : IEEE Computer Society Press, 2011 P. 895-898.

[4] Z. Juhász, M. Juhás, L. Samuelis, and Cs. Szabó, "Teaching Java programming using case studies." Teaching Mathematics and Computer Science. Vol. 6, no. 2 (2008), p. 245-256.

[5] A. Bollin and L. Samuelis, "Experiences gained in teaching software project management by simulation." In: Informatics 2009 : Proceedings of the Tenth International Conference on Informatics: Herľany, Slovakia, November 23-25, 2009. - Košice : Elfa, 2009 P. 244-247.

[6] A. Bollin, E. Hochmüller, and L. Samuelis, "Teaching Software Project Management using Simulations - The AMEISE Environment : from Concepts to Class Room Experience." In: Software Engineering Education and Training : CSEE&T 2012: proceedings : 25th IEEE Conference : 17. - 19. april 2012, Nanjing, Jiangsu, China. - Los Alamitos, California : IEEE Computer Society, 2012 P. 85-86.

[7] A. Bollin, E. Hochmüller, R. Mittermeir, and L. Samuelis, "Experiences with Integrating Simulation into a Software Engineering Curriculum." In: Software Engineering Education and Training : CSEE&T 2012 : proceedings : 25th IEEE Conference : 17. - 19. april 2012, Nanjing, Jiangsu, China. - Los Alamitos, California : IEEE Computer Society, 2012 P. 62-71.

[8] A. Bollin, E. Hochmüller, and L. Samuelis, "Teaching Software Development Processes by Simulation : Quality Assurance as a Factor of Success." In: CSEE&T : co located with ICSE 2013 : 26th Conference on Software Engineering Education and Training : 19. - 21.5.2013, San Francisco. - Piscataway : IEEE, 2013 P. 364-366.

[9] M. Šimková, R. Garabík, K. Gajdošová, M. Laclavík, S. Ondrejovič, J. Juhár, J. Genči, K. Furdík, H. Ivoríková, and J. Ivanecký, "The Slovak Language in the Digital Age." 1st Ed., Berlin Heidelberg : Springer-Verlag - 2012. - 85 p.

[10] Z. Putnik, M. Ivanovic, Z. Budimac, and L. Samuelis, "Wiki - A useful tool to fight classroom cheating?" In: ICWL 2012 : the 11th International Conference on Web-based Learning : proceedings : 2-4 September 2012, Sinaia, Romania. - S.l. : Springer, 2012 P. 31-40.

[11] J. Genči, "Methods to ensure higher variability of knowledge tests in the moodle LMS environment." In: Emerging Trends in Computing, Informatics, Systems Sciences, and Engineering : Lecture Notes in Electrical Engineering 151. - New York : Springer, 2013 P. 447-453.

[12] M. Juhás, Z. Juhász, L. Samuelis, and Cs. Szabó, "Measuring the complexity of students' assignments." In: Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae. Vol. 31 (2009), p. 203-215.

[13] J. Genči, "A few reflections regarding assessment in an e-learning environment." In: CISSE 2010 : Computer, Information, Systems Sciences and Engineering : International Joint Conferences : December 3-12, 2010, Bridgeport. - [Bridgeport : Bridgeport University], 2011 P. 1-4.

[14] H. Telepovska and M. Toth, "Support of relational algebra knowledge assessment." In: Emerging Trends in Computing, Informatics, Systems Sciences, and Engineering. - New York : Springer-Verlag, 2013 P. 475-485.

[15] H. Telepovská and Z. Havlice, "Relational Algebra Knowledge Assessment in Practice." In: Journal of Communication and Computer. Vol. 9, no. 2 (2012), p. 226-233.

[16] L. Samuelis, A. Bollin, K. Frühauf, J. Ludewig, and H. Sandmayr, "Software Engineering Fundamentals Measuring, Comprehending, and Managing your Software Projects." 1st Ed. - Košice : TU - 2012. - 208 p.

[17] P. Antonitsch and B. Sabitzer, "On Competence-Based Learning and Neuroscience." In: Informatics in Schools. Sustainable Informatics Education für Pupils of all Ages, ISSEP 2013 (Springer Verlag GmbH), p. 171-183.