Students with Practical Software Project Experience Perform Better at Related University Subjects – A Truth or a Myth?

Cs. Szabó* 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* — There is an ongoing discussion about the pros and cons of allowing Bachelor and Master students to become employees at companies and to work besides their study. Even more, in the field of Software Engineering, this issue is even seen as a prerequisite. However, are such students really achieving better results in their studies? In this paper, we deal with this question at least from the perspective of Software Project Management skills. We start with the hypothesis that their work (and project) experience yields to better results in a lecture dealing with these topics. Then we describe the setting of a first pre-study, which was embedded in a lecture about software quality and management at the Technical University of Košice. Next, we discuss the results. In our setting, it turned out that the difference is marginal in average. Even though the scores of students with previous practical experience are not so diverse at the beginning of the lecture, students with prior experience perform slightly better at the end of the course.**

I. INTRODUCTION

Teaching at Universities is always a big challenge. Usually, 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] and the way of teaching changed a lot during the last decade [13, 14].

Modern trends in teaching nowadays 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.

The interesting point here is that instead of being deductive as in the cases 1 and 2 above, simulations require predictive thinking. In addition, there is another significant quality of teaching by using simulations: the knowledge largely stems from the students' own experience or it is perceived as their own experience. This experience through simulation does not replace real practical experience, but it comes very close to it. This idea is also one motivation for using the AMEISE simulation environment [9]. AMEISE provides students of informatics, business - and economical informatics as well as subject-related university or college studies the opportunity to gather project management experience on a simulator.

Gaining practical experience (be it as Bachelors' or Masters' internals, or as side jobs) is possible and sometimes usual for computer science students. Such an activity is not forbidden at the Technical University of Košice (TUKE), but it is also not required. At the University of Klagenfurt, it is also part of the Applied Computer Science curriculum.

This means that the majority of the students at our institutions usually concentrate on studying only the first semesters, but then start working either in industry or in projects at University's departments. In the last year of their studies most of our students are either part-time or full-time employees. Being a member of a *real world* team working on *real* projects and inside a company's atmosphere suggests an increase of the level of experience in teamwork and project management. However, can this experience, can these extra skills, be successfully used at exams and other activities in different learning subjects?

In order to answer this question, we decided to do a first evaluation (pre-study) and embed it in a course called Software Quality and Management. This course takes place in the last year of our Informatics and Applied Informatics Masters' degree study programs at TUKE. The students can use their practical experience. We also implemented a radical change in the course setup according to the last years' schedule, which we called mixed-up schedule [10]. This schedule allows spreading theoretical lectures across the whole semester – with very low workload during the semester – but then to concentrate on simulation runs (i.e. labs) during one week with extremely high workload (so really focusing on only one topic).

The paper is structured as follows: In the following two sections, we introduce the setting of the experiment and our hypothesis. Then, we introduce the measurement method. Next, we present our results in textual and visual form. Finally, we discuss our results and come up with some conclusions. We close our paper with hints towards further research questions, stemming from the findings in the paper.

TABLE I. EXAMPLE OF FEEDBACK (TIME NEEDED TO WRITE A SPECIFICATION DOCUMENT) PROVIDED TO THE TRAINEES REGARDING THE CONSTELLATION AND SKILLS OF THE DEVELOPMENT TEAM.

Group	Author(s)	Spec. (hours)	Working Days
tuke-201	Richard	79.25	10
tuke-213	Stefanie, Richard	123.74	8
tuke-219	Richard, Axel, Christine	156,26	7

TABLE II. EXAMPLE OF FEEDBACK (QUALITY OF THE REVIEWING PROCESS) REGARDING THE CONSTELLATION AND SKILLS OF THE DEVELOPMENT AND REVIEWING TEAMS.

Group: tuke-209 (Specification)		Legend:
Author	Richard	Reviewing skills:
# Errors Produced	114	Bernd → high
Reviewer(s)	Christine, Stefanie	Stefanie → high
# Errors Found	35	Richard → low
Corrector(s)	Christine, Richard, Stefanie	Others → medium
# Errors Corrected	25	Specification skills:
# Errors Remaining	89	Richard → high
		Diana → low
		Others → medium

II. HYPOTHESIS

The main question, as motivated for in the introduction section, is whether experienced students are really achieving better results in their studies. Therefore, our hypothesis is:

H1: *TUKE students having practical software project experience (from e.g. being team-members on software projects, preferable at real software companies) achieve better results in AMEISE simulation runs within our mixed-up schedule.*

The zero hypothesis then would be that there is no measureable difference in the results between students with practical experience and the rest of their (less experienced) classmates.

An answer to this question clearly depends on the definition of a "good" or a "bad result". In our case, results are measured as the level of fulfillment of simulation goals (time, resources needed, quality of final products) in two consecutive simulations, separated by one feedback session. More details on the setting and the measurement/evaluation method are now described in sections III and IV.

III. SETTING

A. The Simulation Environment

The AMEISE (A Media Education Initiative for Software Engineering) approach focuses on the simulation of software project management processes. Based on Stuttgart University's SESAM (Software Engineering Simulation by Animated Models) [11], the AMEISE tool-set allows for repeatedly experiencing the complexity of software project management within a game-like simulation environment [12].

In this environment, trainees have the opportunity to run an AMEISE simulation using the so-called quality assurance (QA) simulation model. Based on the description of the required project (simulation goals, available resources), the participants act as project managers who have to decide about adequate staffing and the allocation of software development as well as quality assurance tasks. After the simulation process, the success of each project is analyzed using the AMEISE self-assessment feature. Tables I and II present some examples of the assessment data available. At the end of a simulation run the results are also summarized (see Table III) and then used as the basis for the calculation of the scores.

B. The Course Setup

The mixed-up schedule combines the "classical" 13-week semester of lectures with a teaching block at the end of the semester for 148 TUKE students. 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.

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 take 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 week with the block.

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.

TABLE III. EXAMPLE OF RESULTS (TIME NEEDED FOR DEVELOPMENT, MONEY NEEDED, FUNCTIONALITY IMPLEMENTED, ERRORS PER LOC, FUNCTIONALITY IN THE MANUALS, ERRORS IN THE MANUALS) FOR STUDENT GROUPS 301 TO 309.

Group	Duration	Costs(€)	AFP C (%)	# E / LOC	APF Man. (%)	# E / Page
tuke-301	274	216420	95.57	12.86	91.20	0.70
tuke-302	254	204220	97.69	10.50	96.25	0.23
tuke-303	214	214060	94.25	19.08	93.60	0.56
tuke-304	218	193460	96.79	9.45	96.76	0.19
tuke-305	148	179220	93.44	20.44	94.07	0.42
tuke-306	242	178380	97.35	10.96	95.69	0.29
tuke-307	232	207580	95.94	13.77	93.49	0.55
tuke-308	204	229980	95.76	13.37	95.83	0.32
tuke-309	277	252900	87.16	43.12	91.20	0.68

TABLE IV. PART OF THE RESULTS OF THE FIRST SIMULATION RUN (TIME NEEDED FOR DEVELOPMENT, MONEY NEEDED, FUNCTIONALITY IMPLEMENTED, ERRORS PER LOC, FUNCTIONALITY IN THE MANUALS, ERRORS IN THE MANUALS) INCLUDING THE TOTAL NUMBER OF POINTS ACHIEVED.

	Project Duration [d]	Project Costs [EUR]	Code AFPs [%]	Code Errors [#]	Manuals AFPs [%]	Manuals Errors [#]	# Students with PM	Project Duration [d]	Project Costs [EUR]	Code AFPs [%]	Code Errors [#]	Manuals AFPs [%]	Manuals Errors [#]	Total # POINTS
Objectives	**270**	**225000**	**95**	**12**	**95**	**0.5**	**Experience**	**16**	**16**	**18**	**18**	**16**	**16**	**100**
Group														
tuke-405	319	416,060.00	93.59	18.89	92.68	0.56	1	0.00	0.00	5.40	1.80	4.80	6.40	**18.40**
tuke-319	158	138,380.00	89.26	30.95	0.00	0.00	2	14.40	14.40	0.00	0.00	0.00	0.00	**28.80**
tuke-205	177	186,940.00	88.96	37.23	78.48	0.43	1	14.40	14.40	0.00	0.00	0.00	9.60	**38.40**
tuke-313	262	208,980.00	89.98	29.81	92.88	0.55	1	14.40	14.40	0.00	0.00	4.80	6.40	**40.00**
tuke-318	224	228,740.00	92.46	29.78	93.07	0.63	1	14.40	11.20	3.60	0.00	6.40	4.80	**40.40**
tuke-214	163	238,740.00	92.45	24.29	93.31	0.54	1	14.40	11.20	3.60	0.00	6.40	6.40	**42.00**
tuke-415	268	264,940.00	93.15	17.01	93.36	0.56	2	16.00	6.40	5.40	3.60	6.40	6.40	**44.20**
tuke-204	232	230,580.00	95.60	15.30	77.65	0.52	1	14.40	11.20	12.60	3.60	0.00	6.40	**48.20**
tuke-420	240	194,180.00	93.12	23.89	93.65	0.47	1	14.40	14.40	5.40	0.00	6.40	9.60	**50.20**
tuke-305	148	179,220.00	93.44	20.44	94.07	0.42	1	14.40	14.40	5.40	0.00	8.00	9.60	**51.80**
tuke-409	211	169,100.00	91.54	34.74	96.09	0.31	1	14.40	14.40	1.80	0.00	12.80	11.20	**54.60**
tuke-201	183	234,020.00	92.67	24.99	97.25	0.22	1	14.40	11.20	3.60	0.00	14.40	12.80	**56.40**
tuke-216	164	209,140.00	94.63	18.79	95.59	0.30	1	14.40	14.40	7.20	1.80	11.20	11.20	**60.20**
tuke-421	241	183,140.00	96.67	10.40	96.69	0.28	1	14.40	14.40	14.40	10.80	12.80	12.80	**79.60**
tuke-212	189	179,940.00	96.68	9.88	96.51	0.21	1	14.40	14.40	14.40	12.60	12.80	12.80	**81.40**
tuke-206	211	227,620.00	97.84	7.58	97.14	0.20	1	14.40	11.20	16.20	14.40	14.40	12.80	**83.40**
tuke-211	219	215,180.00	97.40	11.79	98.25	0.30	1	14.40	16.00	16.20	10.80	16.00	11.20	**84.60**
tuke-315	361	317,500.00	84.50	34.87	90.94	0.71	0	0.00	0.00	0.00	0.00	0.00	0.00	**0.00**
tuke-418	261	313,340.00	79.62	63.42	0.00	0.00	0	14.40	0.00	0.00	0.00	0.00	0.00	**14.40**
tuke-410	290	329,700.00	92.52	27.56	91.20	0.73	0	8.00	0.00	3.60	0.00	3.20	0.00	**14.80**
tuke-408	273	322,260.00	85.28	50.51	91.20	0.70	0	16.00	0.00	0.00	0.00	3.20	0.00	**19.20**

4. Feedback session on first simulations.
5. Second simulation runs.
6. Feedback session on second simulations.

IV. MEASUREMENT

The pre-study and, related to it, the measurement was split into three steps:

1. A pre-test on the profile of the students (e.g. Software Project Management skills and work experience).
2. Extracting the results from the simulation runs using the AMEISE reporting interface (retrieving the data collected in the AMEISE database).
3. Calculation of scores (results) for each simulation run.

A. *Profile of the Experimental Subjects*

The course consisted of 148 students who were split then into 74 groups of two students each. We had 10 female students and 138 male students of age between 21 and 25. 19 students (only one of them was female) stated that he or she had prior experience in leading projects. That included resource/time management, staffing, customer care and effort estimation. 12 students did not provide this information to us. 28 students also already had at least one project management course before, but only five of them had practical project management experience.

After splitting the students into groups of two, we ended up with 2 groups where both student members had project management experience, 15 groups where one student member had project management experience, and 57 groups with no project management experience at all.

B. *Measurement of Simulation Results*

Several attributes are reported back from the AMEISE environment as the result of a simulation run (see Table III). In our simulation setting a software system had to be "produced", and the following attributes (simulation goals) were used as a basis for the evaluation:

1. The duration of the project (in simulation time) which should not exceed 270 working days.
2. Costs (needed for delivering a product) which should not exceed 225.000 EUR.
3. The number of Adjusted Function Points (AFPs in percentages) covered by the code. Here, the customer requires at least 95% of the functionality to be implemented in the delivered system.
4. The number of errors per 1000 lines of code. The customer requires that this number is blow 12 (which, by the way, is easy to achieve with modest quality assurance steps).
5. The number of Adjusted Function Points (AFPs in percentages) described in the manuals. Here, again, the customer requires at least 95 percent to be covered.
6. The number of errors per page in the manuals. The customer expects not to find more than one error every second page.

For the final grading (as it was a lecture) we had to defined some lower limits of success, but as there have been two simulation runs, the most important factor was *how* the teams *improved* their simulation results.

For the pre-study, however, we followed a slightly difference strategy which is described in the next section.

TABLE V. PART OF THE RESULTS OF THE SECOND SIMULATION RUN (TIME NEEDED FOR DEVELOPMENT, MONEY NEEDED, FUNCTIONALITY IMPLEMENTED, ERRORS PER LOC, FUNCTIONALITY IN THE MANUALS, ERRORS IN THE MANUALS) INCLUDING THE TOTAL NUMBER OF POINTS ACHIEVED.

	Project Duration [d]	Project Costs [EUR]	Code AFPs [%]	Code Errors [#]	Manuals AFPs [%]	Manuals Errors [#]	# Students with PM	Project Duration [d]	Project Costs [EUR]	Code AFPs [%]	Code Errors [#]	Manuals AFPs [%]	Manuals Errors [#]	Total # POINTS
Objectives	**270**	**225000**	**95**	**12**	**95**	**0.5**	**Experience**	**16**	**16**	**18**	**18**	**16**	**16**	**100**
Group														
tuke-319	254	254,940.00	94.28	17.24	24.72	0.76	2	**14.40**	**9.60**	**7.20**	**3.60**	**0.00**	**0.00**	34.80
tuke-420	247	289,220.00	96.43	13.38	50.76	0.20	1	**14.40**	**4.80**	**12.60**	**5.40**	**0.00**	**12.80**	50.00
tuke-214	275	304,020.00	98.55	5.67	92.96	0.64	1	**16.00**	**0.00**	**16.20**	**16.20**	**4.80**	**4.80**	58.00
tuke-305	270	233,420.00	95.58	22.64	95.90	0.30	1	**16.00**	**11.20**	**10.80**	**0.00**	**9.60**	**11.20**	58.80
tuke-415	304	288,020.00	96.82	9.02	96.90	0.21	2	**8.00**	**4.80**	**12.60**	**12.60**	**11.20**	**12.80**	62.00
tuke-201	269	249,500.00	97.21	12.96	96.90	0.22	1	**16.00**	**9.60**	**14.40**	**7.20**	**11.20**	**12.80**	71.20
tuke-318	242	219,580.00	95.68	14.02	97.96	0.14	1	**14.40**	**16.00**	**10.80**	**5.40**	**12.80**	**14.40**	73.80
tuke-405	269	242,140.00	97.97	8.71	97.02	0.19	1	**16.00**	**9.60**	**14.40**	**12.60**	**12.80**	**14.40**	79.80
tuke-421	283	215,700.00	98.48	5.67	97.56	0.23	1	**8.00**	**16.00**	**16.20**	**16.20**	**12.80**	**12.80**	82.00
tuke-211	256	228,300.00	97.69	9.26	98.01	0.14	1	**14.40**	**12.80**	**14.40**	**12.60**	**14.40**	**14.40**	83.00
tuke-204	274	187,220.00	97.78	7.69	96.90	0.21	1	**16.00**	**14.40**	**14.40**	**14.40**	**11.20**	**12.80**	83.20
tuke-206	249	226,180.00	98.27	6.41	97.14	0.20	1	**14.40**	**12.80**	**16.20**	**14.40**	**12.80**	**12.80**	83.40
tuke-216	274	251,540.00	98.29	5.79	97.14	0.19	1	**16.00**	**9.60**	**16.20**	**16.20**	**12.80**	**14.40**	85.20
tuke-409	227	193,580.00	98.17	6.11	97.51	0.17	1	**14.40**	**14.40**	**16.20**	**14.40**	**12.80**	**14.40**	86.60
tuke-212	269	217,780.00	98.07	7.56	97.40	0.20	1	**16.00**	**16.00**	**16.20**	**14.40**	**12.80**	**12.80**	88.20
tuke-205	270	219,180.00	98.39	6.92	97.56	0.20	1	**16.00**	**16.00**	**16.20**	**14.40**	**12.80**	**12.80**	88.20
tuke-313	273	221,820.00	97.39	9.91	99.01	0.08	1	**16.00**	**16.00**	**14.40**	**12.60**	**16.00**	**16.00**	91.00
tuke-309	280	266,260.00	93.02	21.95	73.99	0.24	0	**8.00**	**4.80**	**5.40**	**0.00**	**0.00**	**12.80**	31.00
tuke-301	267	151,300.00	93.24	22.45	91.20	0.70	0	**16.00**	**14.40**	**5.40**	**0.00**	**3.20**	**0.00**	39.00
tuke-406	325	294,860.00	96.74	15.12	96.87	0.23	0	**0.00**	**4.80**	**12.60**	**3.60**	**11.20**	**12.80**	45.00
tuke-426	283	204,820.00	92.90	24.28	96.78	0.30	0	**8.00**	**14.40**	**3.60**	**0.00**	**11.20**	**11.20**	48.40

We used the reporting interface of the AMEISE server to retrieve all the relevant data from the database. Then, we put these data into a Spreadsheet application, assigned points to ranges, and summed up the results for each team separately. Tables IV and V present some of the achieved results for the first and second simulation runs.

C. *Simulation Results' Calculation*

In order to have a stable and precise way of assessing the simulation results, we said that only the top 10% of the results (when being sorted) count for 100% of the points. Then, the next 10% got 90% of the points, and so on.

This approach leads to a very strict way of assigning individual points, but in our opinion, it seems to be a good discriminator for skills as the results are ranked according to the "big picture" of all achieved goals, and the top 10% of the students do get the full amount of points (and so on).

The resulting sets of score points were then used to calculate statistical values (median, quartiles, outliers, etc.) for both groups of interest: the ones with project management experience from practice (called "Exp. Trainee") and others without such experience (called "Unexp. Trainee" in our figures and tables).

V. RESULTS

In this section, we present the results that we got during and at the end of the experiment.

Table V presents the results of the 17 "experienced" student pairs in the first simulation run (continuing with sample data for the inexperienced pairs), and Table VI shows the results then for the second simulation run. The colors used also indicate the big difference in the goals achievements. For the lecture only an improvement in the goal achievements was necessary, but for the pre-study we did not look at these deltas but again only tool the absolute values of the single goal achievements into account.

Table VI summarizes some statistical data concerning the first and the second simulation run. One can see the marginal difference between the medians 50.2 and 52.2 for the first simulation run. However, the median then differs much more for the second simulation run. The difference is significant: 82 for experienced trainees versus 74 for the inexperienced ones.

A much clearer view is provided when looking at the related box-plots (see Figures I and II).

VI. DISCUSSION

The evaluation of the simulation results based on team member experience (i.e. using two distinct groups) showed that the advantage of initial knowledge used in the first simulation has no real effect on the results. The values of the median are almost identical in our setting. Only the diversity is significantly less for the group of experienced trainees. This might be caused by common routines learned in the practice that cannot be mapped to the topics covered by the lecture – thus representing

TABLE VI. STATISTICAL EVALUATION CONCERNING THE RESULTS OF THE FIRST AND THE SECOND SIMULATION RUN

	Simulation I		**Simulation II**	
	Exp. Trainee	Unexp. Trainee	Exp. Trainee	Unexp. Trainee
Min	18.4	0	34.8	31
Q1	40.4	34.4	62	60.6
Median	50.2	52.2	82	74
Q3	60.2	68	85.2	83
Max	84.6	83	91	98.4
IQR	19.8	33.6	23.2	22.4

FIGURE I. BOX PLOT FOR THE RESULTS OF THE 1ST SIMULATION RUN FOR THE EXPERIENCED AND UNEXPERIENCED TRAINEES. (N=74)

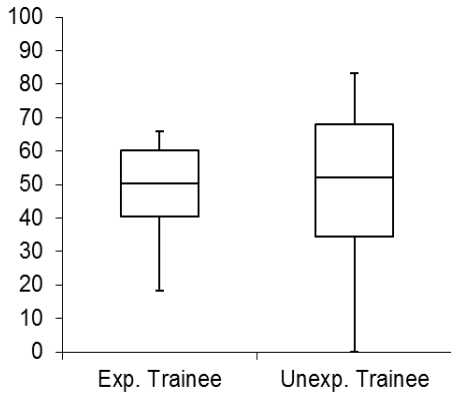
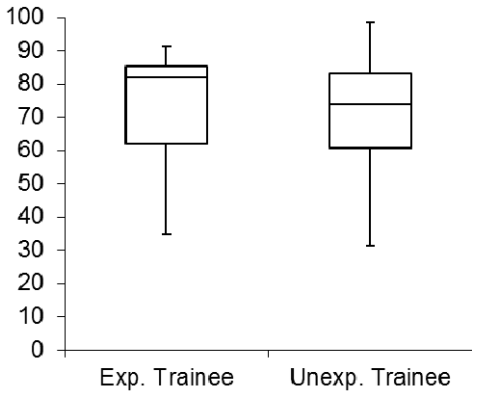


FIGURE II. BOX PLOT FOR THE RESULTS OF THE 2ND SIMULATION RUN FOR THE EXPERIENCED AND UNEXPERIENCED TRAINEES. (N=74)



practical knowledge that were not able to be transformed into utilizable knowledge.

The results for the second simulation runs are more interesting in some sense. Here, we found a larger difference in numbers when looking at the median value – the score differs between 82 for the experienced trainees and 74 in the case of inexperienced ones. The upper and lower bounds of the quintiles are still close to each other, yes, even closer than in the case of the first run, but the experienced students are a bit better than the rest. It is hard to interpret this increase in the achieved number of points, but we assume that experienced students are faster in mapping their experiences (and skills) to similar challenges. What is commonly treated as "maturity" would mean to be able to adapt faster to new situations.

Looking at the findings above, we might come to the following conclusion. Our hypothesis H1 was not confirmed by our experiment. The first simulation run demonstrates that the experienced groups are not doing a better job.

However, the diversity of the groups changed a lot between the first and the second simulation run, and there might be several factors influencing this. We assume that there is a big influence of the feedback session given between the first and the second simulation run. For similar studies we definitively need to eliminate this possible side effect.

VII. SUMMARY

To check the hypothesis that students with work-experience in software engineering do perform better in software engineering lectures, we implemented an experiment where we tested project management skills before, between and after two software project simulation runs within the AMEISE simulation environment. The pre-study was conducted during a course about software quality and project management in the last year of the Master study program at the TUKE. 148 students attended the course during our experiment.

We used questionnaires to collect personal data about the practical experience of each student. This data was then used to separate them into the two groups of interest (our experimental subjects): the set of "experienced" and the set of "inexperienced" attendees.

After every simulation run we used the AMEISE server's services to evaluate the run according to predefined criteria as presented above.

Though we collected a lot of data and expected that experienced students do perform better, our hypothesis was not confirmed. They achieved slightly better results at the end of the lecture, but definitely only achieved comparable low results at the first simulation run. Their better results in the second simulation run are documented in the data, but the reasons for that are questionable. It might be several factors: faster learning skills of experienced students, the quality of the lecture and the feedback session or even other factors not known so far.

The identification of the possible reasons and their proofs are now on the focus of our further investigations concerning this topic.

Finally, it can be said that both groups improved and learned a lot during the simulation runs. Qualitative feedback collected during the lecture (which is not presented here in this paper), clearly tells us that the feedback session was of great importance. Students told us that it was *the* part of the lecture where they learned most. In that sense, it seems to be in fact a very influencing factor.

ACKNOWLEDGMENT

Many thanks go to Elke Hochmüller from the Carinthia University of Applied Sciences in Klagenfurt and Barbara Sabitzer from the University of Klagenfurt for helping and supporting us during the 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. 019TUKE-4/2014: "Integration of the Basic Theories of Software Engineering into Courses for Informatics Master Study Programmes at Technical Universities – Proposal and Implementation."

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] A. Bollin. E. Hochmüller, and R. Mittermeir, “Teaching Software Project Management using Simulations.” In: J. B. Thompson, E. Oh Navarro, D. Port (Ed.): 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T 2011) Proceedings. New York (NY): IEEE Computer Society Press, 2011, pp. 81-90.

[10] Cs. Szabó and A. Bollin, “On a Mixed-up Schedule for Teaching Software Quality and Project Management – An Experience Report.” In: Proceedings of the 4th International Conference on Information Technology and Development of Education ITRO 2014. Zbornik Radova. Zrenjanin, June 2014.

[11] A. Drappa, J. Ludewig, “Simulation in Software Engineering Training.” In: Proc. 23rd International Conference on Software Engineering, IEEE-CS and ACM, May 2001: 199 - 208.

[12] R.T. Mittermeir, E. Hochmüller, A. Bollin, S. Jäger, M. Nusser, “AMEISE - A Media Education Initiative for Software Engineering: Concepts, the Environment and Initial Experiences.” In Auer (ed): Proceedings International Workshop ICL – Interactive Computer Aided Learning, Villach, Sept. 2003, ISBN 3-89958-029-X.

[13] M. Shaw, “Software Engineering Education: A Roadmap.” In A. Finkelstein (ed.): Future of Software Engineering 2000; ACM, 2000, pp. 373 – 380.

[14] J. B. Thompson, “Software Engineering Practice and Education: An International View.” In: Proc. SEESE’08, ACM, 2008, pp. 95 – 102