Conducting a Joint Course on Software Engineering Based on Teamwork of Students

Zoran Budimac, Zoran Putnik, Mirjana Ivanovic, Klaus Bothe, Kay Schuetzler

Informatics in Education, 2008, Vol. 7, No. 1, 17–30

Conducting a Joint Course on Software Engineering Based on Teamwork of Students

Zoran BUDIMAC, Zoran PUTNIK, Mirjana IVANOVIĆ
Department of Mathematics and Informatics, Faculty of Science, University of Novi Sad
Trg D. Obradovica 4, 21000 Novi Sad, Serbia
e-mail: {zjb, putnik, mira}@im.ns.ac.yu

Klaus BOTHE, Kay SCHUETZLER
Institute of Informatics, Humboldt University Berlin
D-10099 Berlin, Germany
e-mail: {bothe, schuetzl}@informatik.hu-berlin.de

Received: October 2006

Abstract. For the previous six years, under the auspices of the "Stability Pact of South-Eastern Europe" and DAAD, a joint project for developing a course in "Software Engineering" has been conducted. The intention of the project was to enable usage of shared materials for software engineering courses at a wide range of universities in participating countries. During school-year 2004/05, for the first time the same course, with the same case study, and the same assignments has been conducted at the Humboldt University Berlin, and the University of Novi Sad. In this paper, we share some of the experiences obtained through conducting the same course in the two school-years: 2004/05 and 2005/06.

Keywords: common course, self-assessment, teamwork.

Introduction

During the year 1999, as a part of cooperation between professors from Humboldt University in Berlin and University of Novi Sad, an idea emerged for cooperation in creating and developing common courses in certain fields of computer science. Since the idea very much coincided with the trends in European higher education, some other universities from South-Eastern Europe, with which we already had some collaboration, have been included in the project. Under the auspices of the "Stability Pact of South-Eastern Europe" and "DAAD – Deutscher Akademischer Austausch Dienst" this project has lasted for six years now (Bringing Curriculums and Equipment Up To Date, 2002; SE Course Homepage).

Currently, there are thirteen universities from eight countries participating in the project. Humboldt University of Berlin, Germany; University of Novi Sad, Republic of Serbia; University "Cyril and Methodius", Skopje, FYROM; and University of Plovdiv, Bulgaria, are the "core" members. Besides, Universities of Belgrade, Niš, and Kraguje-

vac from Republic of Serbia; University of Zagreb and University of Rijeka from Croatia; Universities of Banja Luka and Sarajevo from Bosnia and Herzegovina; Polytechnics University of Timisoara from Romania; and University of Tirana from Albania are included in the project (Zdravkova *et al.*, 2003a; Zdravkova *et al.*, 2003b; Budimac *et al.*, 2003).

Besides the final goal of academic reconstruction of South-Eastern Europe, the project established some other goals:

- inclusion of "Software Engineering" course as a core one into universities' curricula in South-Eastern European countries;
- gaining of consensus in creation of a joint "Software Engineering" course, and determination and selection of appropriate software engineering topics as the basis for the common pool of topics;
- creation and development of joint teaching and exam materials for selected software engineering topics: slides, case-studies, assignments, exam questions, literature, etc. and
- establishment of research and education framework as the basis for the future cooperation.

These goals are implemented mostly through cooperation in creation, improvement and enhancement of teaching materials, and production of a distributed, Internet-based, multilingual university course.

The work on the project has been performed in a threefold manner:

- continually, during the preparation for classes and exercises, lecturers try to innovate and to improve and enhance the existing teaching material;
- each year, during the winter break, representatives of the "core" universities meet to prepare a plan of further activities for the current year, and decide how to distribute assignments, in order to obtain maximum efficiency;
- annually, the project members attend a workshop. At the workshop, all participants give presentations about the results they achieved, and the duties they fulfilled during the previous school year. Furthermore, many presentations give new insights and ideas for further work and possible improvements in effort to create common teaching materials (not only in the domain of software engineering, but also in some other project community courses: programming languages, compiler construction ...). So far, the following meetings have been organized:
  - September 2001, The First Seminar on Teaching Software Engineering and Reverse Engineering (TSERE), Novi Sad, FR Yugoslavia;
  - September 2002, The Second Seminar on TSERE, Plovdiv, Bulgaria;
  - September 2003, The Third Seminar on TSERE, Ohrid, FYR Macedonia;
  - September 2004, The Fourth Seminar on TSERE, Zagreb, Croatia;
  - September 2005, The Fifth Seminar on TSERE, Baile Herculane, Romania;
  - September 2006, The Sixth Seminar on TSERE, Nesebar, Bulgaria.

During the project, all partners played an active role in it, either by:

- contributing to some part of the materials;

- using some topics or subsets of the joint course material in their lectures;
- writing review reports based on collecting students opinion and suggestions;
- presenting some new interesting SE topics;
- analyzing possibilities and proposing new directions in developing teaching material for other courses.

The common software engineering course originated from the course that has been conducted at the Humboldt University in Berlin for several years. Its main goal was to present some introductory notions and principles of software engineering, including also a wide spectrum of sub-areas suggested by the ACM and IEEE societies (Computing Curricula, 2001) and others (SWEBOK, 2001). It has been shown that the original course held at the Humboldt University, covers more than 85% of the basic lessons suggested in "Curricular guidelines for undergraduate programs in computing" (Bothe *et al.*, 2003). The course was to be held at higher years of computer science curriculum, after students become familiar with basics of (object-oriented) programming and other important fields of computer science.

The project was conducted in three partially overlapping phases:

1. During the first phase, teaching materials for existing topics were translated from German to English, and then, through the cooperation of all the participants, refined.
2. In the second phase, new interesting SE topics were developed.
3. In the third phase, it was anticipated to create local versions in national languages (Bothe *et al.*, 2005). For the purpose of localization, a dedicated tool for work in a multi-lingual environment was developed (Bothe and Joachim, 2004).

The course is accompanied by a pool of case-studies to be discussed during lectures and processed through assignments. From this pool, lecturers from the project universities are free to select the most suitable one(s). In addition to this, there is a pool of assignments, referring both to the course contents and to the case studies, thus forming the base for specific exercises. Together with the assignments, sample solutions, correction hints, and typical errors are collected. Through these several additions, a great flexibility is added to the course.

The rest of the paper is organized as follows: In the second section, the technique used for assignment solving in teams is described. In the third section, experiences with students' solutions of assignments both in Berlin and in Novi Sad are presented. The fourth section presents an idea that emerged in Novi Sad of self-assessment between students, members of the same team, and explains teamwork organization in more details. Finally, in the fifth section, some conclusions and discussions of common insights gained so far by conducting the same teaching material at two Universities (Berlin, Novi Sad) are given.

Experiences of Teaching SE in Two Different Countries

Parts of the joint course (individual topics or complete subsets) have been taught to students in the six universities that participated in the project. However, for the first time

during the school year 2004/2005, a complete, absolutely identical course, with the same case studies, and the same assignments for students was held in Germany (Berlin) and in Serbia (Novi Sad). The course was being presented to the students of the fourth, final year of study, with more or less similar pre-knowledge, with the similar number of classes for the similar type of subjects. In addition, a form of self-evaluation inside student teams was introduced in the course in Novi Sad, which is the key contribution of this paper.

The same course lasts for one-semester in Berlin (7th) and for two semesters (7th and 8th) in Novi Sad. Still, the total number of classes is the same at both Universities. The number of students attending the course at the Humboldt University was 78, while there were 58 students in Novi Sad. The same lecture style was adopted in the subsequent school years at both Universities, and now we are able to compare the experiences and draw some conclusions based on the experiences in teaching the course twice in two consecutive school years: 2004/05 and 2005/06.

Lectures in Novi Sad were held in Serbian, while slides presentations used were in English (this was the students' decision). Lectures and slides of presentations used in Berlin were presented in German. At both institutions, read-only handouts were put on local sites – after each lecture – the Serbian and English versions in Novi Sad, and the German version in Berlin. Also, lectures lasted the same for each topic, except for "Software process models" and "Cost estimation" that lasted longer in Novi Sad, due to personal preferences and the decision of the lecturer.

During the entire course, students have to solve seven relatively small assignments. To do that in an appropriate SE manner, they are divided into teams, according to their own choice. This approach has several advantages (Bielikova and Navrat, 2004). The first is simplicity from the managerial point of view. Second is that opportunity for a student to sign up for the team of her/his choice creates a tendency to base the choice on personal relationships. Thus, the time needed for adjustments and adaptation of team members is drastically shortened. Third, efficiency of teams created in this manner tends to be rather high.

All of the mentioned advantages are inclined to minimize most of the real-life problems arising. Teams *do* get started without additional effort, extra meetings are much easier organized if students know each other – and often have similar preferences, and usually there is a natural pressure on team members "not to disappoint their friends". The only obvious disadvantage of this approach is a risk that team quality can (and usually does) vary significantly.

In Berlin there were 26 teams having up to three persons in each team, while in Novi Sad there were 12 teams, each having four or five members. Later, in the second year, due to the increased number of students, in Novi Sad 15 teams of four or five students were created. The same assignments were given to each team, which were to be solved in three weeks. The assignments were as follows:

1) review of "preliminary requirements specification" and "requirements specification",
2) application of the function-point method on a given preliminary requirements,
3) review of a product model developed after structured analysis,

4) development of a part of a static model, through creation of a use-case diagram, and a class diagram for a given problem,
5) review of a solution of the 4th assignment of a different team,
6) definition of formal specification (Z and algebraic) for several given operations, and
7) installation, testing of and commenting on a tool for software quality metrics.

The main reason for this type of work is a widely accepted objective of a good teaching style and appropriate curriculum – to prepare students for their professional career, to make them able to survive the ever changing demands of the surrounding world, and to make them capable of solving complex, demanding tasks. One of the recognized practices is to involve teamwork in education, as it is critical for solving the complexity of software tasks. The most valuable points we hoped our students would acquire were: to show abilities to work and communicate within a team, and to cooperate sufficiently to plan and develop the needed output of their project.

Some Observations about the Assignments

In practice, assignment solving functions as follows: Teams are given tasks and expected to produce results in a given time (no less than three weeks). Each member of a team is expected to read, think about, and reflect on a task *before* the team meeting, so that the whole team is prepared for discussing and solving the task through (possible) several meetings. After all the solutions are submitted and evaluated by lecturers, one class is organized where the most interesting and provoking solution is presented by the members of the team, that submitted it.

The solution presented is either the “best”, or the “worst”, or simply the “best defended” by creators. A class is organized so that one team presents their solution and the rest of the students are involved in discussing, criticizing, glorifying, or simply commenting on it, while the lecturer is just a moderator of the discussion. After the presentation, the “proper” solution is pointed out – if such a thing exists in software engineering – by a lecturer, i.e., a solution collected by several years of combined experience of Humboldt and Novi Sad Universities.

A number of points have been awarded to each assignment. In Germany, a student must earn at least 50% of the points to enter the final exam, where these points are of no interest anymore. In Novi Sad, the similar condition holds, but the earned points directly influence the final mark.

Typical students’ errors and omissions were almost the same at both universities. The same stands for the distribution of points among teams.

For example,

- The *first assignment* – review of preliminary specifications – *was the most vague* (as it is, after all, also in real life) for both groups of students, producing the whole spectrum of observations.
- Connected to the same assignment we observed that *the mark for the first assignment was the best mark* of each team, compared to their other marks.

- For the *second assignment* – application of the function-point method – *no team received the maximal number of points* at both universities *ever*.
- Solutions to the *second assignment* produced *very different values* for function points ranging from 170–300, with the similar common errors and omissions.
- *Similar* observation can be made *about other results* that were required. Value for the man/month ranges from 1.6 to 3.2. Values for influencing factors in connection with other applications were ranging from 0 to 5, i.e., through the whole scale. Still – the function point method is in theory characterized as a very subjective method, so this kind of results has been expected.
- *Errors in the second assignment* (if any), were connected to *the same* thing – not recognizing either the complex function that should be divided into several functionalities, or several simple functions that actually produce a single functionality (and should be therefore treated together).
- The *third assignment* – review of a product model developed after structured analysis – had a twofold importance. Schemes of Data Flow Diagrams were taken from a software engineering essentials book (Balzert, 1998), including all errors. It was a part of an attempt to convince students that even in a book and much more in real life, *errors and omissions occur in software engineering* documents and models.
- The second task for students in the *third assignment* was to try to find the errors and correct them. The *results were very good* – more than 90% of teams gained more than half of the points.
- The *third assignment* was in both institutions the most interesting one because of *over-creative results*. Considering themselves "experienced software engineers" by now, there were several teams that were more interested in creating their own solutions, than in finding errors in the existing solution (their original task). Moreover, those teams "requested" better marks for their creations.
- The *fourth and fifth assignments* were obviously connected, and teams produced *similar solutions*. It was the second half of the course when this assignment was given. Students had already gained some experience, so even when they produced errors in their solutions, they defended those errors reasonably and knowledgeably.
- The *fifth assignment* – review a solution of another team – will be discussed in *more details in the following section*, concerning the relationship among teams, and between teams and lecturers.
- The *last two assignments* – formal specification, and installation and testing of a software package – were quite *straightforward*, and *did not cause too many problems* to students. Their main purpose turned out to be collection of additional points and improvement of the final mark. Yet, we would like to emphasize here that by this time, all of the teams *had* enough points to pass this part of the exam.

In Fig. 1 are presented results of assignment evaluation for the assignments in Berlin and in Novi Sad for the two finished years. Results of the third year are still not available.

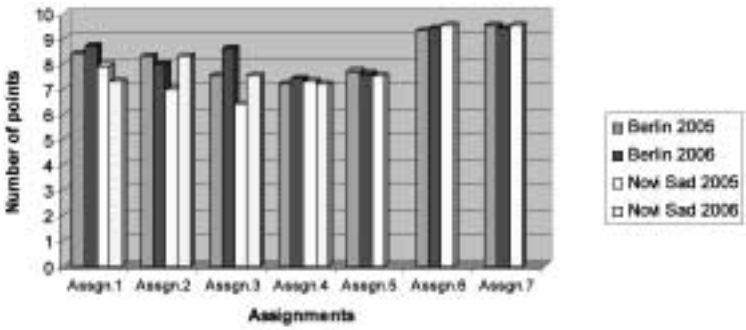

Fig. 1. Results of assignments evaluation.

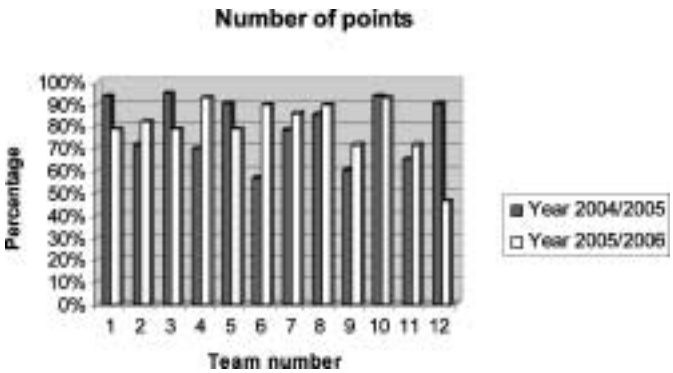

Fig. 2. Total percentage of assignment points in Novi Sad.

At the end, all of the teams collected more than 60% of the points, so that there was no problem with fulfilling the requirements for the exam. In Berlin, all of the teams fulfilled the required tasks and collected more than 50% of points – as most of them usually do each year.

In Fig. 2 the total percentages of points gained for the assignments in Novi Sad are given.

It is interesting to note that there were teams that *stopped* doing the exercises as soon as they acquired the minimum number of points needed. In addition, while usually everyone met criteria as long he/she stayed with the course, there were teams or persons who *left* the course during the semester (some not taking the exam at all). As far as the statistical data is concerned, gained points ranged from 62 to 93 percent of the total amount available both in Novi Sad and Berlin.

It is also interesting to mention that sessions where solutions were discussed, analyzed, and criticized, always provoked arguing and strong exchange of opinions. This produced very creative classes. As can be found in various readings, in order to create a realistic environment, the amount of freedom and supervision during the teamwork process should be balanced. Simulating reality, a high degree of freedom should be given to students in their solutions, discussions, and opinions. Yet, since students usually have

little or no experience with (large) projects and teamwork, some level of monitoring, guidance and supervision is needed to ensure advancements and successful results.

The main argument of the teams was that the specification of problems and appropriate documentation that were the subject of their assignments were 'imprecise' and 'vague'. In fact, this was their first contact with the real-life problems, where there are no definite solutions. These sessions became a significant part of the whole course – exchanging ideas and opinions, and seeing different solutions helped students to see the real-life problems more clearly. The comparison of obtained results and solutions of students' tasks in other universities and in other courses, together with realization that they thought similarly, proved to be reassuring to students. In their own words, they felt "satisfied because their thinking, judgment, and ideas were correct."

Teamwork and Self-Assessment

As mentioned earlier in the paper, a form of self-evaluation of student teams was performed in Novi Sad. For assignment types used in this course, where each team was required to produce a 5–20 pages long report on their solution, depending on the type of assignment, it was impossible to "cheat". Still, we were worried that team members would not be equally involved, and would not equally contribute throughout the whole process of solving a given task. We were interested to find out how they felt about that, and eager to try to teach them to value good teamwork, and recognize a bad one.

Besides trying to find out if there were sleeping members inside teams, we tried to develop among students a feeling for teamwork, self-assessment and assessment of their team colleagues. Inspired by Srikanth *et al.* (2004), where the similar principle was applied to pair programming, we asked students to fill in anonymously a questionnaire assessing the contribution of other team members to the final result. Each member of a team answered the following questions about the other members of his or her team:

- Did each team member read the assignment and the preparation material *before* the beginning of teamwork?
- Did each team member made an equal contribution to the final solution?
- Did each team member explicitly and creatively contribute to the final solution?
- Was each team member cooperative during the work?

The answer for each question was given as a number of points – between 0 and 25. They were promised that those marks would not be shown to other members of the team, to ensure their honesty, and that they would not influence their final mark, and would be used *only* for research purposes. Although two years are a relatively short period for a more formal statistical evaluation of the self-assessments, we can present some preliminary experiences.

With each assessment, lecturers pleaded the team members not to "cover" their non-working colleagues, and reminded them that agreeing to give each other higher marks would add nothing to their exam marks. Still, it seems that those pleas didn't have much effect. The following numerical results give the reason for this statement:

- 67% of marks were maximum – 25 points.
- Additional 21% of marks were 20 points or more – again “excellent” marks.
- Two teams gave each other maximum number of points for each assignment, to each member.
- Another five teams gave each other such marks that the average mark for each member of a team was higher than 20 points.

Fig. 3 presents results of a team self-assessment.

Still, if we disregard two teams with average marks of 25, all of the other teams showed the same trends:

- In each team, there was always the “best” member, and the “worst” member. The best member had 2–5 marks lower than 25, while the worst member had 10 or more marks lower than 25. Even though those marks were still rather high (mostly over 20) the difference was evident.
- The person who was marked as the best was usually the one to give lower marks most freely to the other members of a team. This person was not necessarily the “team leader” (selected by students), yet it was not unusual that she/he would give the leader a much lower mark.
- In the opposite case, the person who was given the lowest marks by her or his colleagues would usually give the other team members all the highest marks. We guess that it was a way to try to achieve better marks for oneself.
- There was an equal number of students whose marks changed in either direction: students that started inferior, but received much higher marks later, and students that started “perfect”, but gained only 5–10 points on later assignments.
- An interesting, while not that important, note can be made about the distribution of the marks. As students are also “mathematicians”, the round marks 5, 10, 15, 20, and 25 cover 89% of all the marks.
- Finally, as we expected, the worst marks were given to those students who were not regularly attending classes. Namely, students are required to attend 50% of classes in order to get the signature from the professor, verifying “regular attendance”, ensuring enrollment into the next semester without an additional fee. While the

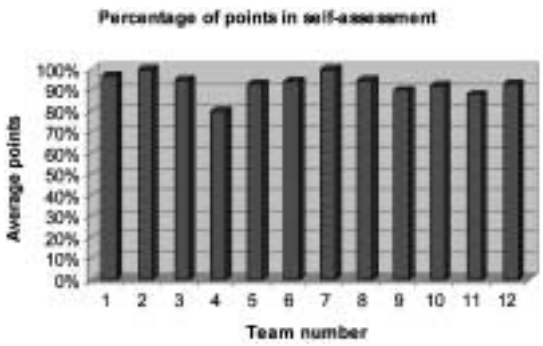

Fig. 3. Percentage of points gained per team during self-assessment.

percentage of students earning the signature is very high, considered normal for the students of the final year, those few who did not attend enough classes, were also marked by their colleagues with the lowest marks.

The trends present in assessments showed that at first the students were very friendly towards their colleagues, giving them only the best marks. Yet, as time went by, marks were beginning to lower, and become more regularly distributed. The only difference between the students is "how low" the lowest mark would go – yet, if we scale marks for the last assignments, we can see that they are leaning towards normal distribution.

Teamwork, especially in a variant of self-elected teams, has one additional advantage. During the two-semester work, we encountered several conflicting situations inside teams, including situations when a single member was excluded by a decision of the rest of the team. In our opinion, students who experienced such realistic situations are much better prepared for actual life and work situations, and for their professional career.

As discussed in (Bielikova and Navrat, 2004), there is also a problem of the size of a team. Namely, the bigger the team is, the more difficult it is for its members to fully participate in all of the activities. So, subgroups are formed, individual roles become overlapping and unclear, all of this becoming a source of a possible additional conflict. On the other hand, if teams are too small, their creativity and flexibility of them decreases. In (Bielikova and Navrat, 2004) the suggested team size is between 4 and 8 members, optimally 5 or 6. In our case, the decision was also led by our wish to make a better distinction between team-members, since points gained through assignment solving directly influenced their final mark. So we were inclining towards smaller teams, 4 members on the average.

Considering the assignments, we decided to follow the suggestions of our DAAD project and present each team with the same task, each time connected to the lectures presented earlier. Yet, one more experiment has been conducted. After each assignment, the gained team marks were presented to all of the students. After complaints about "low marks" for the first three assignments, we decided to ask students to mark their colleagues. So, teams with contradictory solutions were opposed to each other, anonymously, of course, yet *none of the teams* knew whose solution was good, or how many points they had gained. We asked them to discuss and evaluate the other solution as their fifth assignment. As we expected, yet as a great surprise for students, those marks were significantly lower than marks given by lecturers. On a bad side, this experiment showed that there were also few teams who were not quite sure about their solution, changed their views, and praised the completely opposite solution by another team. This showed that students tend to underestimate the complexity of a problem, while overestimating their knowledge and capacity.

Conclusion

General opinion of all of the project members is that the project was very successful and useful, mainly based on the following characteristics:

- Time for the course preparation is drastically shortened.
- Students are enabled to learn in accordance with contemporary contents, principles, and European standards.
- Course compatibility, both general and concrete, is achieved.
- An excellent base for usage of distance learning principles is created.
- Experiences, methods, and learning activities of lecturers from several different countries are adopted.
- Possibilities for different kinds of cooperation with the project participants are promoted.

Concerning experience obtained from teaching and exercising software engineering in the same way in Novi Sad and in Berlin, and the process of marking, teamwork, assessments, and self-assessments, we would like to emphasize a solid (or higher) dose of humor and good will shown both in the texts explaining the solution, and in the later discussions. This, together with a great amount of defending their attitudes, opinions, and positions on each of the questions and concepts, represents a mature and expert thinking from the students. We consider this as a good indication leading towards creation of persons capable of quality teamwork, persons aware of their abilities, responsibilities, and knowledge, thus prepared for working on a real-life software projects.

However, there are still a lot of open issues concerning this kind of teamwork. Organization of projects similar to real-life ones, trying to present industrial conditions, and time, cost, and resource restrictions is clearly desirable, but hardly an achievable and realistic goal. Still, software engineering course relying only on an academic environment could not be really successful, and should try to get at least some kind of “industrial”, “real-life” involvement.

Moreover, it is interesting to notice that the two rather different types of students/professors/countries produced very similar results. Considering the pre-knowledge of the students, we think that it was very similar at both universities. Yet, lecturers in Berlin were performing the same course for the last sixth years, while in Novi Sad, the course has been held twice for undergraduate students, and twice for postgraduate students. In addition, University of Novi Sad, and specially Department of Mathematics and Informatics is going through reforms, the same as in the whole country, quite different to Germany.

An additional difference lies in the method of passing the exam. In Germany, a student is allowed to try to pass an exam only once, except with a medical confirmation of illness, when one additional attempt is allowed. Quite differently, until recently, in Serbia students were allowed to try to pass the exam an indefinite number of times, having around 10 exam periods during the year. Furthermore, students in Serbia have the right to have the exam on the same subjects that were presented to them several years ago, even though the course had been changed in the meantime. Starting from this year, according to Bologna declaration, students will be allowed to try to pass the exam exactly three times during one year. In addition, students gain points during the school-year, which make a part of the final mark.

The previously mentioned principles were used in the “Software engineering” course. More than 70% of the students passed the exam during the first two exam periods (for other courses, this percentage is around 35).

Two more interesting moments have to be mentioned. First, the common course in “Software engineering” is the first course at the Department of Mathematics and Informatics in Novi Sad using *English* slides as a basis for a course. This decision had no negative effect.

Second, a common practice in Novi Sad among students is that the best students “cover” for others, “not-so-good” students. Even if that means that a professor will not have a chance to distinguish between good and other students, it is a usual practice for good students to risk their own marks, trying to help others in cheating, or to cover for them by performing their duties. By insisting on anonymity, and explanations why this practice is wrong, in Novi Sad we tried to convince students to abandon it. Since this is the first course with such an intention, we are very pleased with the results.

With all of the mentioned differences, it is important to notice that the students reacted similarly, had the similar distribution of points, made the same errors, complained about similar issues, . . . Consequently, we believe that an introduction of teamwork was a good idea, worthwhile for students.

References

Balzert, H. (1998). *Lehrbuch der Software-Technik*, Band 1 und 2. Spektrum Akademischer Verlag.

Bielikova, M. and Navrat, P. (2004). Experiences with designing a team project module for teaching teamwork to students. *Journal of Computing and Information Technology*, **13**(1), 1–10.

Bothe, K. and Joachim, S. (2004). Tool support for developing multi-lingual course materials. In *ONLINE EDUCA, 10th Intl. Conference on Technology Supported Learning & Training*, Berlin, Germany.

Bothe, K., Schuetzler, K., Budimac, Z., Zdravkova, K., Bojic, D. and Stoyanov, S. (2003). Technical and managerial principles of a distributed cooperative development of a multi-lingual educational course. In *1st Balkan Conference in Informatics*. Thessaloniki, Greece, 112–120.

Bothe, K., Schützler, K., Budimac, Z. and Zdravkova, K. (2005). Collaborative development of a multi-lingual software engineering course across countries. In *35th ASEE/IEEE Frontiers in Education Conference*. Indianapolis, USA, T1A-1–T1A-5.

Bringing Curriculums and Equipment Up To Date, *DAAD*, **3**, September 2002.

Budimac, Z., Putnik, Z. and Bothe, K. (2003). International educational project – some experiences. In *Proc. of II Seminar on Teaching Computer Science “NaRa 2003”*. Novi Sad, Serbia. Published as a CD Proceedings.

CC2001, Computing Curricula (2001). *ACM and the Computer Society of the IEEE*. `http://www.acm.org`

SE Course Homepage, `http://www.informatik.hu-berlin.de/swt/intkoop/jcse/`

Srikanth, H., Williams, L., Wiebe, E., Miller, C. and Balik, S. (2004). On pair rotation in the computer science course. In *17th Conference on Software Engineering Education and Training* (*CSEE&T 2004*). Norfolk, Virginia, USA, 144–149.

SWEBOK (2001). In P. Bourque and R. Dupuis (Eds.), *Guide to the Software Engineering Body of Knowledge SWEBOK*. IEEE Computer Science Press.

Zdravkova, K., K. Bothe, K. and Budimac, Z. (2003a). SETT-Net: A network for software engineering training and teaching. In *ITI-Information Technology Interfaces*. Cavtat, Croatia, 281–286.

Zdravkova, K., Bothe, K. and Budimac, Z. (2003b). The structure of SETT-Net. In *Eurocon 2003*, vol. 2. Ljubljana, Slovenia, 126–129.

Z. Budimac is a professor at Faculty of Science, Department of Mathematics and Informatics, University of Novi Sad. He graduated in 1983 (informatics), received master's degree (computer science) in 1991 and doctor's degree (computer science) in 1994. His research interests include: mobile agents, e-learning, software engineering, case-based reasoning, implementation of programming languages. He has been project leader for several international and several national projects. He has published over 100 scientific papers in proceedings of international conferences and journals, has written more than 10 university textbooks in different fields of informatics. He is head of computer science chair.

Z. Putnik is a research assistant at Faculty of Science, Department of Mathematics and Informatics, University of Novi Sad. He received master's degree (computer science) in 2004. He is currently working on his doctor's degree in computer science at the University of Novi Sad. His research interest is currently focused on e-learning and virtual learning environments. He is a member of the Yugoslav Society for Applied and Industrial Mathematics, JUPIM, and European Association for Programming Languages and Systems.

M. Ivanović is a professor at Faculty of Science, Department of Mathematics and Informatics, University of Novi Sad. She graduated in 1983 (informatics), received master's degree (discrete mathematics and programming) in 1988 and doctor's degree (computer science) in 1992. Her research interests include: multi-agent systems, e-learning and web-based learning, data mining, case-based reasoning, programming languages and tools. She actively participates in more than 10 international and several national projects. She has published over 100 scientific papers in proceedings of international conferences and journals, has written more than 10 university textbooks in the field of informatics and ICT.

K. Bothe is a professor at the Institute of Informatics, Humboldt University Berlin. His research interests cover: compiler construction, software engineering, software testing methodology and tools, programming languages, e-learning, and logic programming. During the last years, he was the grantholder of an international DAAD project and of an EU Tempus project concerned with the development of a Joint Master's curriculum in software engineering and the construction of teaching material repositories.

K. Schuetzler has been a consultant at BearingPoint since 2007. As a member of the Competence Group Integration Services in Public Services/Infrastructure Services he is mainly engaged in IT-consulting tasks. Prior he was employed by Humboldt-University Berlin, Germany as a research assistant at the Software Engineering Group of the Computer Science Department. He graduated in 2001 (computer science) at Humboldt-University Berlin. He is about to finish his doctoral studies in computer science within the next few months. As a research assistant in the Software Engineering Group Mr. Schuetzler was responsible for the organization of project seminars and exercise to the group's lectures. His research interests include: reverse and reengineering processes, requirements management, software quality assurance (including test) and software architectures.

Programinės įrangos inžinerijos kurso mokymas taikant grupinio darbo metodą

Zoran BUDIMAC, Zoran PUTNIK, Mirjana IVANOVIĆ, Klaus BOTHE, Kay SCHUETZLER

Paskutiniuosius šešis metus pagal Pietryčių Europos stabilumo sutartį ir DAAD buvo kuriamas bendrasis programinės įrangos inžinerijos kursas. Šio projekto tikslas buvo suteikti daugumai bendradarbiaujančių šalių universitetų galimybę naudotis programinės įrangos inžinerijos kurso medžiaga. 2004/05 mokslo metais pirmą kartą kursas išbandytas dvejose aukštosiose mokyklose: Berlyno Humboldto ir Serbijos Novi Sad universitetuose. Šiame straipsnyje aptariami tyrimo duomenys (įgyta patirtis) – analizuojamas šio kurso įgyvendinimas 2004/05 ir 2005/06 mokslo metais.