Problems and experiences with student projects based on real-world problems: a case study

Kresimir Fertalj, Boris Milasinovic, Ivana Nizetic Kosovic

Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia

Abstract

Involving students into real-world projects and real teamwork environment is of the great importance in software engineering education that is sometimes ignored in academic environments. The Bologna Process reforms gave the authors an opportunity for necessary changes in the teaching methodology. Although the real-world projects have generated the topics for degree theses at the authors' department for years, the reform provoked the student projects to be comprehensively included in the university curriculum. This paper describes the implementation of the real-world projects within the undergraduate course Development of Software Applications and the authors' four-year experience in teaching that course. The key issues regarding the selection and adaptation of real-world projects have been discussed for different types of student work as well. The organization and the improvements of the course through the years are presented together with course evaluation.

Key words. Real-world project, Bologna process, project-based learning, software engineering education, teamwork.

1. Introduction

Due to the implementation of Bologna Declaration in Croatia, the study of Computing at the *Faculty of Electrical Engineering and Computing* was reorganized and changes have been carried out from the academic year 2005/06. One purpose of the undergraduate study of Computing is to teach students fundamental knowledge and to prepare them for the graduate study. Simultaneously the same undergraduate study must equip students with enough practical knowledge to be skilled and competitive for professional work if they decide not to continue graduate study. Due to such dual nature of the undergraduate study, those changes included teaching paradigm shift towards project-based learning.

Traditional teaching is based on exposing theoretical fundamentals explained by hypothetical examples and case studies. Due to that, a professional having a certificate in programming would be a more desirable candidate for a potential employer then a student thought to be a programmer, and a system analyst and a software architect, but trained on hypothetical problems.

Having this in mind, we recognized a need to introduce one or more courses that will integrate theoretical fundamentals with professional practices simulating real-world problems. Such courses combined with seminars, student projects and degree theses given to solve particular real-world problems enable students to gain invaluable practical experience. Although a huge number of the problems are solved in everyday life, a limited number of them can be used for teaching. Project size, duration, and complexity, intellectual ownership and rights, inappropriate hardware and software platforms can significantly reduce range of plausible themes. Each project has to be downsized and adjusted to the organization of teaching and schedule of classes, depending on the type of the student work.

Real-world problems can be assigned to the students on several levels during their studies. Some problems can be isolated from real-world projects and assigned to students on the courses such as *Seminar* on undergraduate study or *Project* on the graduate study. Bigger project modules can be assigned to students as degree theses (bachelor, master or even doctoral thesis). Parts of a project need to be adjusted in respect to the whole project. An additional adjustment is needed whether the student works on the project with the other students or with the teaching stuff. Another problem is the integration of real-world problems within the courses, depending on the semester in which the course is held.

The focus of this article is the incorporation of the real-world projects into the course *Development of Software Applications (DSA)*, taught in the last, 6th semester of the undergraduate study. A few examples of the projects successfully performed during the course as laboratory work and homework are presented. Although the article is not focused on other types of assignments of the real-world projects such as seminars and projects, they are also commented to emphasize a different approach compared to the course-related projects.

The importance and benefits of using real-world projects in students' assignments and an overview of the related work are given in the second section. The third section discusses the dilemma between simulating a user and involving a real one. Problems in choosing and adapting real-life projects are elaborated in the fourth section. The fifth section presents a case study of a successful course organization.

2. Importance of real-world projects

"Software engineering (SE) is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software" [1]. Software engineers require both technical and social skills [2]. The traditional teaching methods, focused only on lectures and tutorials, are not sufficient for SE students to develop skills needed to solve real-world problems [3]. In traditional approach, students must complete their tasks mostly on their own, in contrary to professional practice of team environment and collaboration. Collaboration with other students in traditional approach can even be penalized.

The need for modern approach to teaching SE is not new, but the significance of supplying students with real problems and real teamwork environment is often ignored in software development in academy [4].

Project-Based Learning (PBL) which *is a student-centered instructional approach used to promote active and deep learning by involving students in investigating real-world issues in a collaborative environment* [5], is becoming recognized as the valuable approach to SE [6, 7].

Importance of practical assignments and tendency to simulate real projects are discussed in several articles emphasizing the importance for students and for teaching staff. Today's students are already familiar with advanced computer features and they are not impressed by the typical and hypothetical introductory programming examples [8]. Thus, students are more eager to produce something that has value beyond the classroom [9]. In [10] authors noted that impressionable students are forming opinions of the utility of computer science and questions whether students would eventually change computer science study for another study with better chance of giving them a career with some more noble purpose. In [11] is noted that, when working on projects lacking application outside of the classroom, students are primarily interested in getting a good grade and they spend more energy guessing the intent of the instructor and adapting to grading criteria than they are focused on project requirements. In [12] authors emphasize that *choosing a problem that is not within the experience of most students forces students to develop a high-level understanding and design before coding, as early implementation is not feasible*. Through this experience, students get insight into the project requirements and constraints from the client perspective, learning how to overcome misunderstandings between clients and developers in terms of vocabulary, technology complexity and capabilities [12]. Focusing on real-world projects in courses means that student's assignments do not have predefined problem or solution sets, which makes them harder to grade, but drive the students to extend their decision making skills [13].

Using real-world projects as course projects has benefits to teaching staff as well. Such teaching paradigm may encourage instructors to use projects for more than just student grading, such as to apply experimental SE principles and to use projects as part of their research [13]. Still, it is important that in the case of using real-world projects as course projects the main objective is not on the research perspective or technology transfer perspective but on the educational goals [14].

Real-world project assignments can be applied not only to SE courses, but also to seminar themes, project themes and degree theses. This could be beneficial for all participants because it better simulates students' future jobs and it is more interesting than hypothetical problems and it could be a stepping-stone for future research and business projects.

3. Simulation of a real-world project

As noted in [6] it is imperative that a clear and concise scope statement must be formulated in order to get a list of requirements. The (in)famous software development process tree-swing comic about the design and development process [15] is a good reminder that SE is a specific discipline that deals with humans and it is prone to many misleading guides. Simulating real-world project in students' assignments should not neglect this. Design of a complete real-world system starting from a vague, or incompletely specified problem statement is an invaluable educational experience of a problem that professionals may face [12].

If a real user is involved in student projects, students will soon realize the problem of imprecise, inconsistent and changing requirements. However, teaching schedule is limited and usually it is not possible or appropriate to involve a real end user in a teaching process due to several reasons. There is no free lunch - if the user has to spend his/her time on the project, then s/he may have big expectations that could not be fulfilled regarding the scope, time constraints and educational goals of the course/project. Furthermore, if the project fails or ends with an expectations gap, then the disappointment of wasted time and effort could prevent further collaboration in future projects. Therefore, if success is not foreseeable it is better to simulate the user by teacher.

Two notable obstacles should be carefully avoided when impersonating the user. The first one is tendency of the teacher to elicit the requirements clearly and consistently, as a professional, not the user. Opposite to that, the teacher can exaggerate when making an impression of a user who changes his/her mind, makes digressions and elicits vague requirements, which may direct the students to guessing the requirements. An experiment with intentional misleading and consequences after teams have been reshuffled has been done in [16].

4. Typical issues in selection and preparation of real-life problems

Real projects can be very large and monstrously complex. Some concerns regarding selection and adaptation of real-world problems, their implementation and maintenance are listed in this section, regardless of teaching and learning context (course, thesis, etc.). The problems are related to problem decomposition, resource and time management, isolation of a student's work, intellectual property and to project maintenance.

4.1. Problem decomposition

A major problem is how to make real-world projects feasible in academic settings. Project size and complexity can be reduced by decomposition and modularization (the well-known divide-and-conquer paradigm). The partitioning can be very demanding, hence one must take care of size of student team(s), teaching schedule and teaching workload, In addition to that, the modularization adds an integration overhead.

Special care should be taken to estimate if a potential theme is of an appropriate size for a particular type of student's work (a seminar, course project or thesis). Obviously, different types of student work have different aim and scope. However, the same theme can evolve from one stage of tuition to another, for instance from an individual seminar to a degree thesis. In such case, an incremental project would develop as a student performing it gradually advances through the study. For example, a student would do a seminar in system analysis, then do an individual project of system design, and finally develop a system while working on his/her degree thesis. On the other hand, a team of students can do a prototype, and only some of them would later be interested in completing a final version.

4.2. Resource and time management

Engagement of students in real-life projects is limited by a lack of previous knowledge and a lack of time. Course schedule determines the deadlines and time the students should dedicate to work on their assignments limits is limited, which leaves no room for extensive research. This is usually accompanied with the fade of interest and ambition as problems arise, because students are not used to act professionally. As novice professionals, students may lack work ethic. They may have different priorities. Some of them work during their study and give priority to honorary jobs. Some of them are preoccupied with other courses and some just have some private issues that hinder their study. All of these problems are emphasized if they are working in a team. A student's performance during his/her study can be used to assess the technical competence. The student's reliability is not so easy to asses because she or he can lag in execution of other obligations and decide not to pass a subject in favor of another one.

Thus, one of the important questions is to whom to assign a theme. To assign a theme to an average student now or save the theme for a better one enrolling next month, semester, year, …? Assigning the theme to the first student that shows interest for the theme can produce insufficient or useless deliverables. The students may be unreliable. They can leave the study, prolong it for too long or may simply be incapable of completing the task. In a company, such task would probably be reassigned to another person (although causing some waste of time and additional cost). For a one-man-project (e.g. thesis), reassignment is not possible during the semester. Consequently, the theme would become "worn out". Despite the aforementioned risk, there is no sense to wait indefinitely for a hypothetical ideal student, as we have to work with resources that we have. Furthermore, judging students' motivation and prior knowledge sometimes can be difficult and waiting for a better student can prolong work indefinitely and thus can make a theme to become obsolete.

When it comes to student teamwork, a sort of resource leveling may be accomplished, but with the degradation of project deliverables. One solution to the problem can be design-to-schedule approach, paying a special care to the team cohesion and continuous supervision of the team, forcing the team members to jointly progress, and eventually leaving out less important features in case of lacking time or manpower.

Another approach is to have some backup resources in the project. It is not unusual that some students from a group give up from the project or fail to do their assignments. Thus, having backup resources ensure that project would be finished on time. If all students complete their work on time, surplus resources could be used for implementing additional features and improving the product quality.

The third approach is to integrate students in the existing team consisting of teaching staff (a professor as project leader and the assistants as architects and developers) in which a student's tasks can be more easily reassigned to another member of a team or scaled down if necessary.

4.3. Intellectual property and work isolation

Working in a real-life project may impose some additional constraints. A student must be provided with proper (eligibly real) data to be processed and/or to test the software. The data can be generated, but data generation sometimes can be a project itself. Moreover, a student may use some software components built/bought by somebody else (e.g. university, customer). The solution is to sign a disclosure agreement and/or copyright agreement.

In bigger projects, students work should be isolated as much as possible to decrease the coupling, therewith to reduce interdependency with the rest of development being in progress. However, business needs always change in real projects and software must meet the business needs. One cannot just wait for the student to finish his/her work as initially planned (if ever). If there is software already being in use then the final product of (successful) student's work must be fully integrated with that software. As it is a tendency for students to experiment with new technologies this integration could not always be achieved, or at least it is not a trivial task and may be out of the student's engagement. Nevertheless, implementation of new techniques and technologies can be imperative, but must not become an end in itself. As noted in [14] if an empirical study is part of a course, then the research objectives should not be allowed to dominate over the educational goals.

4.4. Project maintenance

The last, but the not the least problem is related to maintenance of the product (in our case, a piece of software) delivered by student(s). When the product is not going to be used by a real customer and its purpose is only educational, it is upon the teacher to evaluate the outcomes. Otherwise, several questions may arise. Was it worth of effort? Who is going to integrate the project deliverables into the existent system? What about the maintenance and future development? Nothing worth effort is ever gained without effort. In our experience, both the teaching staff and the customer should put additional effort and time in the project in order to ensure the quality of results. They should negotiate (and even contract) their business relationship and roles prior to project implementation. As said before, the engagement of students is limited. However, the students can be engaged in part-time job based on particular project, but this is out of the scope of this paper.

5. Case study

5.1. Representative project examples

In the past few years, authors ran several real-world projects involving students. Some representative examples are described in this section.

The first example is Flora Croatica Database (CROFlora), [17]. Started as a business project and a standalone single user application [18] circa ten years ago it evolved to a rather complex information system with multiuser web applications for the systematic and taxonomic classification of the Croatian flora including bibliography, herbaria, observations, gallery, distribution maps, spatial analysis, and so forth, representing the most important botanic resource in Croatia. Finally, it has found its usage as a valuable resource for seminars and degree theses for the test of new technologies with real users and real data. It has been used as a project within the course DSA and some CROFlora components were developed for the degree theses. Some of those components were integrated in CROFlora almost as-is. Some of the components completed as prototypes (although planned as a fully working versions), due to significant flaws in implementation or due to the problems in integration. As in “real” software projects, the success varied from project to project (thesis). Some of them could be considered a failure due to poor design, failure to set and manage expectations or just because being behind the schedule.

The second example is Fauna Croatica (CROFauna) [19], started as a system specification study for Croatian State Institute for Nature Protection and some of the CROFauna subsystems later became themes for degree theses. Two subsystems (wolf tracking and marine mammal monitoring) have been further expanded and adapted by students making another two success stories.

Tracking and prediction of movement of wolves started as Wild life observer (WLO) mobile application to support field observations, implementing triangulation of VHF collars and tracking of GPS collars. It was developed for a veterinarian as an assignment in course Project. One year thereafter the same students received the Rector’s award [20] and applied for Microsoft Imagine Cup competition and won 2nd place in Croatia. Upon graduation, the team members did some development professionally (for a fee). WLO also became a case study project for the course DSA and yielded a theme for doctoral thesis [21].

The fourth example is Marine Mammal Monitoring Database system [22], developed as a degree thesis under additional supervision of the mentor and assistants. The system was built by using an application framework developed at the department and based on CSLA [23].

5.2. Organizing the course on applications development

The course DSA is held in the last semester of Software Engineering undergraduate degree program. The course elaborates SE concepts, principles and techniques and prepares students for development of complex interactive applications, particularly database applications. The course provides knowledge for successful design, construction and implementation of application software. The weekly schedule of themes is given in Table 1. There are three hours of lecture and one hour laboratory exercises each week.

The aim of the course is to provide students with fundamentals and with practical knowledge of software development, as a prerequisite for the upcoming master courses Information systems development and Project management.

Unlike in [12] and although [8] points that one semester might be too short time for students to work on real-world projects, in our case the real-world project has to be integrated into the SE course within the syllabus (13 weeks of semester effectively – see Table 1). According to [14], integration of the real-world projects in the existing SE course is the most suitable way in which empirical SE can be taught in bachelor level of study.

Table 1. Weekly schedule for the course DSA

Week	Theme
Week 1	Software engineering fundamentals. Software development life cycle.
Week 2	Project definition. Project plan.
Week 3	Requirements specification. Unified modeling language basics.
Week 4	Coding standards. Programming techniques.
Week 5	Graphical user interface.
Week 6	Data access logic.
Week 7	Object-relational mapping.
Week 8	Software architectures.
Week 9	Multi-layered applications.
Week 10	Universal and self-adaptable program modules. Report design.
Week 11	Web applications.
Week 12	Service oriented architecture.
Week 13	Interactive help and software documentation. Software release.

About one hundred students enroll the course DSA each academic year, which complicates forming, monitoring and control of the teams. In contrast to that, in [24] the authors presented the successful implementation of the SE course which enrolled a group of 10 to 20 students. According to [25] truly real-world problems, such as those engaging students in internships, cannot easily be brought to such a number of undergraduate students.

5.3. Organizational context

Project preparation

To make student project feasible, the real project scope must be reduced and project priorities must be redefined. For example, although a GIS map is the key feature of CROFlora and CROFauna, it has the lowest priority when considering those projects as student projects. Since the aim of the course is to teach the students applications development through the project life cycle phases, every student has to practice the whole process from requirements specification to software documentation. (The students tend to be para-professionals and specialists devoted to only some development techniques).

Every year we select a different project, where all student teams solve the same problem. As opposite, on the master courses Information systems development and Project management students work individually. The course DSA is the first undergraduate course where students learn about the software development life cycle in practice and thus we believe they should performed in a controlled environment.

Forming teams

The team is the key component for successful student outcomes and individual learning according to PBL. Team is committed to a common goal and the members share responsibilities and tasks in line with individual goals.

The teams are formed during the first week of a semester and become operational within the next two weeks. The teams have five to seven members. There are several strategies to form a team, based on students' personality, knowledge and ambition. Jenkins [26, 27] identifies four categories of students: “rocket scientists” (already proficient programmers.), “copers” (those who would find the module challenging, but who would cope and eventually pass reasonably well), “strugglers” (those who would find the module difficult, and who would not pass without significant extra support), “competents” (those who remain, who will pass with limited support provided as and when needed). Teams can be selected so that academically weaker students gain the advantage of working with academically stronger students [3].

One cannot know the category a student belongs to, so the teams are formed based on the available information such as the module of study a student enrolled in (it is assumed that students who enrolled SE module are better programmers than the others) and repeating enrollment. The objective is to form mixed but equally competent teams.

Preparation of development environment and collaboration environment

Obligatory development environment is *Microsoft Visual Studio (VS) Team System* (VS is obligatory Integrated Development Environment (IDE) for the course *Development of Software Applications*. In the course *Information system development* students can choose the IDE they like best). Students have access to *Microsoft Team Foundation Server (TFS)* where they can control the source code of their solutions, assign the tasks and so on. They also have access to common database management system (*Microsoft SQL Server*) where they can create and manipulate the team database. The obligatory programming language is C#.

Students' projects are compiled and built automatically on a regular basis (according to *Daily Build and Smoke Test* [28]), after which students and teachers receive e-mail notifications of the build results. This procedure enables detection of the linking errors and avoids situations when the software "works at home, but not in the lab". The described development environment is very similar to that in professional real-world projects.

5.4. Course organization and grading

The course consists of the following assignments (Since this academic year due to faculty regulations, there would be only one mid-term exam instead of two):
- Six homework assignments
- Three short tests on computer
- Classroom activity
- Two midterm exams
- Final exam

The students' final product on the course is the implementation of the given real-world project. The midterms and homeworks represent the project milestones.

The homework assignments and the exams evaluate the project progress and different aspects of development life cycle. Homework and exam are not separate problems. They are rather small units of the students' project.

Although written at home (or in the classrooms and halls) homework assignments and reviewed by assistants in labs, in front of students. Those reviews are very important for students because they get the feedback of quality of their solutions and they can make the corrections on time.

Short tests consist of multi-choice questions with one correct answer. They evaluate general knowledge of software development. The test questions examine conceptual knowledge, which is also important in the context of SE [8, 29].

Classroom activity considers attendance to lectures and collaboration between students and teachers. The interview with the user (either real or emulated) is also considered a class activity.

Each team member works on one segment of the problem domain of a project. Although all teams solve the same problem, every project has a unique work breakdown structure and distribution of its segments (tasks) amongst team members, because they depend on the quality of systems analysis and design and on the team performance. Consequently, the solutions differ in terms of high-level design, coding style, naming convention, data models, user interface design etc. All this makes cheating impossible because it is easier to write own piece of code then to copy, modify (and debug!) somebody else's code.

The overall course grade for a student is calculated by converting the scores on the exams and the homework into a grade-point score. The grading scheme is given in Table 2.

Table 2. The grading scheme for the course DSA

Type of assignment	**Min. of points to pass**	**Percent of grade**	
		Group part	**Individual part**
Homework	50%	0%	15%
Short exams	50%	0%	30%
Class activity	0%	0%	5%
Midterm exams	50%	8%	22%
Final exam	50%	4%	16%
Total	50%	12%	88%

Points of all assignments except exams are strictly individual. Points on exams are earned for individual and group work separately, where individual work prevails. Individual points bring 88% and group points 12% of overall points. The exams look like verification and validation where the assistants evaluate the technical aspects and the teacher (acting as a customer) validates the solution against the requirements. Students have to reach minimum of 50% of the points on each of the assignments and for overall points to pass the course.

5.5. Course evaluation and feedback

Students consider the course hard but very useful, which are the main characteristics of the real-world projects. We gained good feedback from students who were employed after graduation, and also from their employers. Furthermore, students appreciate the effort assistants and teacher put in permanent collaboration and consultations provided almost as 24/7 service.

Distribution of the overall point on the course DSA through the four years is presented in Figure 1. The differences in distribution between the years are minor. Distribution of points (for students that have passed the exam) is similar to Gauss distribution (Figure 1). Distributions of the points earned on the assignments separately are Gauss-like as well.

We believe that the grading scheme presented in Table 2 with emphasis on minimum of 50% on each assignment to pass the course have caused the Gauss-like distribution, which is not the case for many other courses at authors' faculty which doesn't have the minimum of the points on individual assignments.

We continuously improve the course based on our experience and on the comments from students' surveys. Here are some notable changes we have made during the years:

Decreased number of homework assignments. In the first two years of the course, students had to deliver their homework once a week (twelve homeworks altogether). They were complaining on homework frequency, so next two years we recombined existing homeworks into six homeworks. The total points for homeworks and the amount of work were the same as for the first two years, but the delivery was less frequent, thus less stressful for students, who stopped to complain on homework.

Common database server. Every student team develops a common database. In the beginning, the students designed and integrated their databases locally and then delivered it to the source control system TFS. For each update, the database file had to be re-attached in database server on each personal computer, by team members and assistants, which was time consuming and error-prone. Setting up the common database server eased the database development and the evaluation of the students' tasks significantly.

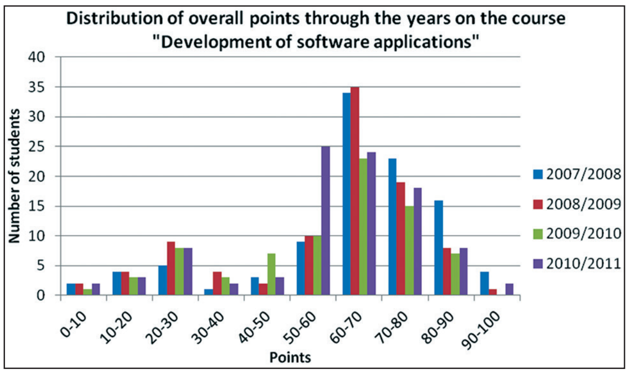

Figure 1. Distribution of overall points for the course DSA

Automated build (and smoke test). The major problem in the first two years of the course was discrepancy in versions of students' solutions at home computers and at assistants' computers as they have often forgotten to check-in changes in their projects. This led to negative evaluation of students' homework when assistants were unable to build students' solutions. Last year, we enabled automated build at the server so the students could check if their solutions were built successfully and had chance to make corrections before coming to formal verification. Regardless of whether the students had enough time to correct the errors, at least bugs discovered on exams did not surprise them.

Obligatory laboratories. At the beginning, the laboratories were not obligatory and only a small number of students ever showed up. As the students were not present at the evaluation of their homework, they did not get the appropriate feedback. So the laboratories became obligatory and now the students have to demonstrate their uploads. The students can explain their solution and assistants can give them suggestions about the corrections and future work.

Student assistants. Student assistants are the students who participated the course antecedent year and who help the actual students in labs and share their experience about the course. Often, students feel freer to ask student assistants for help, rather than asking teaching assistants.

In [30] framework for evaluating the impact of assignments on students learning has been proposed. Although it was not our primary motivation, we are happy that the course satisfies all eleven conditions proposed by the framework. The students' effort is evenly distributed across topics and weeks (conditions 1 and 2 from [30]). Real-world projects engage students in productive learning activity and put high expectations to students (conditions 3 and 4 from [30]). They have to attend homework reviews getting accurate feedback very fast. Sometimes they get all points for homework even if they have made some minor mistakes, but have to fix them until the exam. Such approach can increase motivation, release pressure and help students to keep the focus on the problem. Given feedback is used to improve their work to be delivered for the exams. With such organization of assignments' reviews, conditions 5 to 11 from [30] are satisfied.

6. Conclusion

Teaching based on real-world projects is challenging and requires special effort and time to devise and adapt the student projects and to coordinate or emulate the real user. The course DSA presented in the article deals with software development principles, not the technology, although the technology is extensively used in order to achieve the project goals. As the principles do not change often, the basic structure of the course remain the same since the course was introduced five years ago. The technology changes more frequently, so the source code examples need to be refreshed and upgraded to new versions every year.

When assigning real-world projects for seminars and degree theses, a special attention should be paid to selection of students in order to ensure feasibility of student projects. Student's work brings uncertainty about code quality, integration possibilities and development duration so it would be the best if a student can join an existing team of teachers and assistants working for a real customer. Exception can be made for students that previously showed good results. The business relations with the customer should be agreed (preferably contracted) in advance, defining the responsibilities for the future development and maintenance.

In authors' opinion use of real-world projects in teaching process is win-win-win combination for all stakeholders in the process – potential users, students and teaching staff in which a potential user gets value for money (or value for no money). For students it a valuable experience. Real-world projects increase students' motivation, ensuring that students are not anymore primarily interested just in getting a good grade without too much effort. The students are aware that it is a great way for them to increase their experience and they are willing to work harder if they find their work useful for the customer or beneficial for students' future. From the authors' experience, uncertainty of tasks provokes creativity, but can make the students less satisfied at the same time. We believe that the course described in this paper can be a good basis for students to decide whether they want to become software engineers and software developers or not. We face them with near-the-professional reality and in turn hear comments like "I spent too much time in debugging", "It works at my computer…", "I don't know what user wants…", so we can easily answer "Welcome to the club".

In order to use real-world projects in teaching environment, teaching staff have to devote an additional effort. Although the workload, project performance and software deliverables can be measured, some things are intangible, such as pleasure, pride, experience, frustration, etc. Successful implementation of real-world projects makes teaching staff more attractive to students, up to date with implementations of new technology and enables creation of new courses. Good deliverables make teaching staff satisfied, although sometimes frustration could occur as a side effect of effort devoted to unsuccessful projects. However, real-world examples can make education more interesting, simultaneously serving as foundation for future research or startup of business projects. Finally, they can be a good reference too.

Acknowledgement

This work was carried out within the research project "Sustainable development of information systems", supported by the Ministry of Science, Education and Sports of the Republic of Croatia.

References

1. *Ehgineers, T.I.o.E.a.E., IEEE Standard Glossary of Software Engineering Terminology 1990.*
2. *Car Z., Pripuzic K., and Belani H., Teaching Project Management to Graduate Students of Electrical Engineering and Computing. Technics Technologies Education Management, 2010; 5(1): 73-81.*
3. *Brodie L., Zhou H., and Gibbons A., Steps in developing an advanced software engineering course using problem based learning, Loughborough University, Loughborough, United Kingdom: Higher Education Academy Engineering Subject Centre, 2008; 3: 2-12.*
4. *Su H., Jodis S., and Zhang H., Providing an integrated software development environment for undergraduate software engineering courses. J. Comput. Sci. Coll., 2007; 23(2): 143-149.*
5. *Yam L.H.S. and Rossini P., Implementing a Project-Based Learning Approach in an Introductory Property Course, in 16th Pacific Rim Real Estate Society Conference. Wellington, New Zealand, 2010.*
6. *Savage R.N., Chen K.C., and Vanasupa L., Integrating Project-based Learning Throughout the Undergraduate Engineering Curriculum. Journal of STEM Education: Innovation and Research, 2007.*
7. *MacLaren I., Tutoring Project-based Learning: A Case Study of a Third Year Software Engineering Module at NUI Maynooth, in Handbook of Enquiry and Problem-Based Learning: Irish Case Studies and International Perspectives. , D.D.a.G. Mitchell, Editor. 2006.*
8. *Meyer B., Software Engineering in the Academy. Computer, 2001; 34(5): 28-35.*
9. *Using Real World Projects To Help Students Meet High Standards in Education and the Workplace.2000 Available from: http://www.jff.org/publications/education/using-real-world-projects-help-students-/250, 14.02.2012.*
10. *Layman L., Williams L., and Slaten K., Note to self: make assignments meaningful. SIGCSE Bull., 2007; 39(1): 459-463.*
11. *Buckley M., et al., Benefits of using socially-relevant projects in computer science and engineering education. SIGCSE Bull., 2004; 36(1): 482-486.*
12. *Kershner H., et al., Partnering with Social Service Organizations to Develop Socially-Relevant Projects in Computer Science and Engineering.*
13. *Hayes J.H., Energizing Software Engineering Education through Real-World Projects as Experimental Studies, in Proceedings of the 15th Conference on Software Engineering Education and Training. IEEE Computer Society. 2002; p. 192.*
14. *Wohlin C., Empirical Software Engineering: Teaching Methods and Conducting Studies, in Empirical Software Engineering - Dagstuhl Seminar Proceedings (LNCS), D.R. V. Basili, K. Schneider, B. Kitchenham, D. Pfahl and R. Selby, Editor. Springer Verlag. 2007; p.135-142.*
15. *Awad E.M., Systems Analysis and Design. The Irwin series in information and decision sciences. Homewood, Ill. : R.D. Irwin, 1985.*
16. *Ljubović V., Emulating real-life in student projects in software engineering, in 10th Workshop on Software Engineering Education and Reverse Engineering. Ivanjica, Serbia, 2010.*
17. *FCD. Flora Croatica Database. 2004 [07.11.2011.]; Available from: http://hirc.botanic.hr/fcd/.*
18. *Nikolić T., et al., CROFlora, a Database Application to Handle Croatian Vascular Flora. Acta Botanica Croatica, 2001; 60(1): 31-48.*
19. *Fertalj K., Milašinović B. and Nižetić I., Cro-fauna system specification for State institute for nature protection. DZZP: Zagreb, 2007.*
20. *Katanić N., Ivanac V., and Kopčak G., Croatian Rector's price: WildLife Observer - programska potpora praćenju divljih životinja. FER. 2009.*
21. *Nižetić I., and Fertalj K., Automation of the Moving Objects Movement Prediction Process Independent of the Application Area. Computer Science and Information Systems, 2010; 7(4): 931-945.*
22. *CROdolphin. CROdolphin - Marine mammal monitoring database. 2010 Available from: http://crodolphin.vef.hr. 07.11.2011.*
23. *Lhotka, R., Expert C# 2008 Business Objects. Apress. 2008.*
24. *Blake M.B. and Cornett T. Teaching an Object-Oriented Software Development Lifecycle in Un-*

dergraduate Software Engineering Education, in Proceedings of the 15th Conference on Software Engineering Education and Training. IEEE Computer Society. 2002; p. 234.

25. *Gehrke M., et al., Reporting about industrial strength software engineering courses for undergraduates, in Proceedings of the 24th International Conference on Software Engineering. ACM: Orlando, Florida. 2002; p. 395-405.*

26. *Jenkins T. and Davy J. Diversity and Motivation in Introductory Programming.2004.*

27. *Jenkins T., The motivation of students of programming. SIGCSE Bull., 2001; 33(3): 53-56.*

28. *Mcconnell S., Daily Build and Smoke Test. IEEE Softw., 1996; 13(4): 144.*

29. *Liu S., et al., Teaching formal methods in the context of software engineering. SIGCSE Bull., 2009; 41(2): 17-23.*

30. *Gibbs G. and Simpson C., Does your assessment support your students' learning? Centre for Higher Education Practice, Open University. 2002.*

Corresponding Author
Boris Milasinovic,
Faculty of Electrical Engineering and Computing,
University of Zagreb,
Zagreb,
Croatia,
E-mail: boris.milasinovic@fer.hr