World Transactions on Engineering and Technology Education *© 2017 WIETE*
Vol.15, No.3, 2017

Student projects as a means of cooperation between academia and industry: some experiences in the area of software engineering education

Viljan Mahnič

University of Ljubljana
Ljubljana, Slovenia

ABSTRACT: Cooperation between academia and industry provides an opportunity for software engineering students to experience real projects as a part of their programmes of study. In this way, they improve both professional and transferable skills. The article describes two kinds of cooperation with industry that have recently been used at the University of Ljubljana's Faculty of Computer and Information Science. The first is based on collaboration with big software vendors requiring students to use their development tools in order to develop a project of their choice and evaluate its business implication, audience, benefits and critical success factors. The second requires students to solve a real-life problem on the basis of a specification provided by a partner from industry. The focus is on providing a working solution by strictly following the prescribed development process and meeting user requirements. Advantages and disadvantages of both approaches are analysed and compared to each other. Regardless of their differences, both approaches have proved to be successful and were accepted favourably by all parties involved.

INTRODUCTION

Software engineering (SE) is a practical discipline that cannot be taught only through formal lectures, but requires a great deal of practical work. In order to fully understand and reflect on SE theory, students need to have some experience working on a *real* project, and *vice versa*. To learn successfully from an SE student project course, it is vital to have a sound understanding of the basic principles and practices from SE theory [1]. Therefore, it is suggested that the SE curriculum should include project-based classes providing an adequate combination of theory and practice. Recommendations for undergraduate software engineering curricula developed jointly by the IEEE Computer Society and the Association for Computing Machinery (ACM) also mandate that students undertake a capstone course, which covers one full year [2]. The course should integrate previously learned material, deepen their understanding of that material, extend their area of knowledge, and apply their knowledge and skills in a realistic simulation of professional experience [3].

While the university provides a suitable environment for teaching the theoretical component of these courses, it cannot provide the wealth of experience, practical applications and learning opportunities available within the environment of professional industry. These shortcomings can be alleviated through appropriate cooperation with industry. Industry can help in establishing well-defined projects with real customers and realistic requirements; thus, providing students with a working environment that significantly increases the learning experience and benefits for all parties involved.

Through practical work on an industry-relevant problem, students cannot only deepen their knowledge and further grasp theoretical concepts, but also increase their transferable skills, which are important for their employability. Teachers can use new ways of teaching, based on learning through problem-solving, while industry benefits from students being better prepared for their professional careers [4-6].

In this article, the author describes some experience of the University of Ljubljana's Faculty of Computer and Information Science, in conducting software engineering courses in cooperation with industry. Within the last decade, two kinds of cooperation have been practised. At the beginning, the parties concentrated on cooperation with big software vendors, in particular IBM and Microsoft. The main focus was on practical use of their products and the development of transferrable skills. The students had to work in groups in order to develop a prototype solution for a problem of their own choosing.

Afterwards, the author concentrates on cooperation with small software development companies, which prevail in Slovenia. In cooperation with these companies, realistic projects are defined that students must develop during the course. The main focus is on providing a working solution by strictly following the prescribed development process and meeting user requirements.

In the rest of the article, the main characteristics of both kinds of cooperation are described, and their advantages and disadvantages are analysed.

COOPERATION WITH BIG SOFTWARE VENDORS

Cooperation with big software vendors was motivated by criticism that undergraduate degree programmes in Slovenia are often too theoretical and do not give students enough practical skills that would enable them to have a smooth transition from study to a working environment. On the other hand, it was found that industry expects engineering graduates to have some grounding in business skills with an increasing emphasis on social skills, such as customer relations and team working.

In order to alleviate these problems, cooperation with IBM started in 2006 with the aim of introducing IBM software products and industry solutions to students who attended a special course on Web-based information systems development in the last year of the undergraduate computer science programme [7]. One year later, Microsoft also joined this initiative. Both companies offered their products free of charge and provided instructors who helped the faculty in teaching students how to use them.

Course Content

Course content that combined theoretical aspects of software engineering and information systems development with a practical approach was developed, based on the use of IBM/Microsoft products and solutions. The course lasted 15 weeks and consisted of three parts: lectures, laboratory practice and team project work. Since students obtained a great deal of knowledge of software engineering during the previous courses, most of time was devoted to laboratory practice and team project work.

Laboratory practice was supervised by instructors provided by IBM/Microsoft with the aim of exposing students to IBM/Microsoft software products and tools that were used later during the team project work. To deliver the best possible content, IBM/Microsoft standard courseware was used. Laboratory practice was followed by the group project, which was not only intended for the improvement of technical skills and the broadening of knowledge of software tools used, but also for the acquisition of transferrable skills like teamwork, management/leadership, planning and organising, presentation and communication, information search, etc.

Microsoft projects required students to develop a complete working solution of a problem of their own choosing, which had to be presented to the teacher, a Microsoft university relations person and a Microsoft Visual Studio expert. Each solution was judged according to the criteria of the Microsoft Imagine Cup student contest: 15% problem definition, 60% design, 15% development and 10% presentation.

On the other hand, cooperation with IBM emphasised the development of the skill of generating the big picture, requiring students to recognise that each computer program has its business implication, audience, benefits and critical success factors. Consequently, the main aim was to have students develop only a prototype of the complete solution and develop the logic to explain why this complete solution would benefit the customer.

In order to stress this aim, the presentation of IBM projects had a more formal structure and students were given training to improve their presentation skills. Student teams had to present the prototype solution of their business idea to a business panel consisting of a university professor, an IBM executive and an IBM university relations person. This group of people played the role of a board of directors of a potential customer and evaluated the proposed solution from technical, business and user viewpoints. Each presentation consisted of the following sections: business rationale, technical environment, description of the solution developed and a practical demonstration. After the presentation, each group received feedback about the quality of their project by the individual members of the panel.

The aforementioned cooperation between the University and big software vendors was considered beneficial and fruitful by all parties involved. Nevertheless, it ceased in 2009 due to an economic crisis, which forced both companies to reduce the costs of involvement of their staff in the teaching process.

Advantages and Disadvantages

The opportunity to obtain practical experience working in partnership with one of the leading IT companies was rated by students as the main advantage of the course. A survey after teaching the course for the first time revealed that the students unanimously supported the decision to run the course with a software company. All respondents except one found the course useful (31%) or useful and interesting (66%).

Apart from providing practical experience, the course was also beneficial for students, because it gave them the opportunity to:

- Get acquainted with modern software development tools and technologies;

- Solve practical problems in an almost-real business environment;
- Improve their teamwork and presentation skills;
- Obtain experience not only from the viewpoint of development, but also from the viewpoint of entrepreneurship and selling software solutions.

After completing the course, the students that worked on Microsoft projects also received certificates of attendance that were equivalent to the certificates given to participants of regular training courses provided by Microsoft Certified Partners for Learning Solutions.

The course design also benefited partner companies. By participating in the course, they expanded the user base for possible use of their products and services in the future. They were also given opportunity to attract the best students and offer them employment after completion of their studies.

With regard to teachers, the course offered an improved way of teaching by balancing theoretical and practical aspects of software development.

The main disadvantage of the course was that too little attention was devoted to the development of quality and usable software that would meet user requirements, and be robust enough to survive the encounter with the end-user. Due to time constraints, this issue was intentionally omitted in IBM projects. Instead, the focus was on innovativeness, market potential and technical feasibility of the proposed prototype solution that had to be presented to a panel of experts. Consequently, the course provided good training in transferrable skills, but lacked the practice of coding and testing.

Within Microsoft projects, the author wanted to alleviate this problem; therefore, it was envisioned that the students would provide a working solution of a problem of their own choosing. However, the decision to allow students to define their problems themselves appeared to be problematic since there was no real customer behind their projects to specify realistic user requirements. Consequently, students' solutions were often based on assumptions that could not be met in real life. When teachers wanted to point out such deficiencies, the students insisted that the solution was consistent with the requirements of the problem they envisioned. Eventually, it turned out that the students primarily benefited from mastering the use of Microsoft tools, but got not enough experience in developing software that would meet real user needs.

DEVELOPING REALISTIC SOFTWARE PROJECTS

In order to provide students with a realistic simulation of the industrial environment, another kind of collaboration with industry was defined later that still takes place within the scope of the software engineering capstone course. Instead of allowing students to solve problems of their own choosing, the instructors seek real-life projects that the students must develop in accordance with the requirements specification provided by an expert from industry. Consequently, the role of industrial partners has changed. They no longer provide training in use of (big vendors') development tools, but are asked to prepare a set of user stories that the students must develop and to help supervising student projects.

Course Content

The course content combines teaching of agile software development, in particular Scrum, with a practical team project work. The course lasts 15 weeks and is taken by undergraduate computer science students in their last (sixth) semester. Students are required to work in groups in order to develop an (almost) real project on the basis of user requirements provided by a partner from the industry. The project work is based on the Scrum framework and consists of four Sprints [8]. The first Sprint (also called Sprint 0) lasts three weeks and serves as a preparatory Sprint before the start of the project. The rest of the course is divided into three regular Scrum Sprints (called Sprint 1, Sprint 2 and Sprint 3), each lasting four weeks.

During Sprint 0 formal lectures take place in order to teach students Scrum, and how to apply user stories for requirements specification and project planning. These three weeks are also used to acquaint students with the initial Product Backlog, containing a set of prioritised user stories for the project they are going to develop. At the end of Sprint 0, each team estimates the effort required for implementation of each user story and prepares the release plan.

Sprints 1, 2 and 3 are regular Scrum Sprints that have the same structure. Each Sprint starts with a Sprint planning meeting at which student teams negotiate the contents of the next iteration with the product owner, and develop the initial version of the Sprint Backlog. During the Sprint, the teams have to meet regularly at the Daily Scrum meetings and maintain their Sprint Backlogs, while at the end of each Sprint, the Sprint review and Sprint retrospective meetings take place. At the review meeting the students present their results to the product owner, while at the retrospective meeting students and instructors meet to review the development process in the previous Sprint, giving suggestions for improvements in the next.

At the end of the course, each team presents their completed project to the whole class, instructors and partner company representatives. The presentation must follow a prescribed scenario that requires consistent operation of all user stories developed. Students get to see a variety of solutions and user interfaces for the same requirements, and compare them with each other.

Special attention is devoted to effort estimation, release and Sprint planning, and producing class software required in a real-world environment. Students must strictly follow the concept of *done*, which requires that the code developed in each Sprint must be fully tested and resistant to user errors before being used in practice. A more detailed description of the course can be found in the earlier work of Mahnič [9] and Mahnič and Časar [10].

Within the scope of the aforementioned course, the main task of the industrial partner is to provide realistic user requirements for a project suitable for completion in one semester by a team of four students. It is also highly desirable that the partner provides a domain expert that can play the role of the product owner.

At the beginning of the project, the product owner should prepare the product backlog consisting of user stories that students are going to develop. Each user story must contain a short description and a set of acceptance tests that are used to confirm that the story is fully implemented [11]. During the project, the product owner should be available to provide details about user stories, and at the end of each Sprint, he/she must verify that each user story implementation meets user requirements.

Using this approach, several projects have been conducted recently in collaboration with small Slovenian software companies, e.g. the development of a (simplified) primary healthcare information system, automation of community nursing activities, and the development of a Web portal for offering and booking different services through the Internet. Additionally, the students also worked on several practical projects that were defined by domain experts from the University, e.g. the development of a student records information system, processing of enrolment applications for study in Slovenia (strictly following the rules prescribed by Slovenian law and considering the requirements of each faculty), and the development of a Scrum-based agile project management tool.

Advantages and Disadvantages

The main advantage of this kind of project is that students are required to follow the whole path from requirements specification to the final solution in the form of working software. Cooperation with industry enables that the students work on real-world problems with all their peculiarities and tiny details that cannot be simulated in the academic environment. In this way, they experience the differences between the trivial programming projects of their previous academic experience and the actual level of effort required to produce class software required in a real-world environment.

Within the course, they become acquainted with Scrum, which is the most widespread agile method. The use of Scrum helps to develop their abilities in effort estimating and planning, as well as their skills in presenting results and communicating the solution with the product owner representing the customer. Strict enforcement of the concept of *done* contributes to the awareness that the code must be fully tested and resistant to user errors before being used in practice.

The partners from industry benefit from students' projects in two ways. They can obtain a preliminary solution that after further refinement, can be upgraded into a marketable product or just test an idea for a new product they will develop later. In the latter case, students' projects can help to improve requirements specification, solve particular implementation issues and provide a working prototype for further discussion with potential users. Industrial partners' involvement in student projects also serves as an excellent source of interns and potential hires.

By conducting the course in partnership with industry, members of the teaching staff are allowed to combine traditional lectures with more practical ways of teaching, which are more attractive and allow students to see how the principles from SE theory work in practice. A survey among students after teaching the course for the first time showed that students' opinions are overwhelmingly positive. All students found the course either useful (29%) or useful and interesting (71%); 88% felt that the course was better than other courses in the final year of their study, and nobody rated the course as worse. All students except one (98%) also found the course beneficial to their employability and professional career. The survey identified three skills with a statistically significant improvement over the students' expectations at the beginning of the course, i.e. familiarity with agile approach to software development, customer interaction skills and communication skills.

The course also benefits researchers since it provides a suitable setting for conducting empirical studies with students [12][13]. Since its inception, the course has served as a basis for studying agile estimating and planning [14], the accuracy of planning poker estimates [15], students' acceptance of Scrum and user stories [16][17], and the difference between the planning poker and team estimation game [18].

The main issue in running the course is how to ensure the cooperation of an appropriate product owner. Ideally, an industrial partner representative should play this role. However, our experience has shown that playing this role takes a substantial amount of time. Therefore, it is often unrealistic to expect that a partner from industry could provide a person who would perform all required tasks. It happened only twice, when the teaching staff collaborated with a small start-up established by one of the former students, that a partner representative alone completely prepared and maintained the Product Backlog, participated in Sprint planning, Daily Scrum and Sprint review meetings, and was always available to students for detailed explanation of user stories. In both cases, the partner company was ready to invest additional effort, because they planned to upgrade students' solutions into a marketable product and motivate the best students to join the company.

More often, industrial partners only provide a rough description of the project and help one of the instructors (usually the teacher) to develop the Product Backlog and monitor the student projects. Afterwards, a partner representative only attends the Sprint review meetings and comments students' solutions, while the teacher actually plays the role of the product owner. Playing the role of the product owner requires additional effort on the part of teaching staff, which often exceeds the effort that is usually required by formal lectures. However, it is rewarding, considering the course outcomes in both student progress and satisfaction, as well as in the results of their projects.

COMPARISON OF BOTH APPROACHES

A common characteristic of both approaches is that they stress the importance of practical work. For that reason, the student opinions on the corresponding courses (i.e., the Web-based information systems development course using the first approach and the software engineering capstone course using the second approach) have been overwhelmingly positive. In addition to deepening students' technical knowledge, both approaches contribute significantly to the development of their transferable skills and increase their employability.

Both approaches also benefit the teaching staff and participating companies. The teaching staff is given many opportunities to use modern ways of teaching through project-based learning [19], while participating companies can pursue several intangible benefits, in particular enhancing organisational reputation and outreach that substantially improve their recruitment efforts. In such a way, a win-win situation is achieved for all the parties involved, one that is often reported in the scientific literature (e.g. [20][21]).

Both approaches differ most in the nature of the projects that the students have to develop. The approach the teaching staff used when working with big software vendors was less formalised and allowed students complete freedom in choosing the problem they wanted to solve. Since most of their projects were not based on real client's requirements, the teaching staff did not have an appropriate basis for validating the appropriateness of the proposed solutions. Within cooperation with IBM, this deficiency was compensated for with a greater emphasis on development of transferrable skills. Instead of insisting on providing a fully completed working solution, the students were required to demonstrate the viability of their projects in terms of market potential, development costs, possible users, etc. Participation of IBM experts at the final presentation helped to create an atmosphere similar to the presentation of a business idea to a potential investor.

On the other hand, the new approach that the teaching staff use within the scope of the software engineering course is based on realistic projects defined in cooperation with partner companies. The main emphasis is on full implementation of the required functionality specified in the Product Backlog as a set of user stories, while strictly following the prescribed development process. Using this approach, the students get a realistic working experience and learn how to collaborate with a customer. By strictly following Scrum, they become acquainted with the most widespread agile software development, and gain the necessary teamwork and project management skills.

The partner companies obtain student-developed software solutions that can be further refined in order to be incorporated into a product release, while the teaching staff are given additional possibilities of using modern ways of teaching state-of-the-art topics, such as Scrum and agile software development.

There are also some other differences between the two approaches. While the cooperation with IBM and Microsoft required the partner companies to provide instructors for their development tools, the new approach expects the partner companies to delegate a representative to play the role of the product owner. As stated above, this is a time consuming role and it is often difficult to find a person from industry that can devote enough time to perform all the required tasks. Therefore, it is crucial for the success of student projects that somebody from the teaching staff also knows the requirements of the proposed project and provides students with details of user stories.

Another difference refers to the use of development tools. While the first approach mandated the use of IBM/Microsoft products, the students can now choose the development tools by themselves. Consequently, a shift from commercial to open source tools can be noticed.

CONCLUSIONS

Partnership between academia and industry provides an opportunity for students to experience real software projects as a part of their programmes of study. Working in teams, students demonstrate what they have learned about the software engineering process and directly experience many of the challenges of true software engineering professionals. Through practical work, they not only improve their technical skills, but can also improve their transferable and business skills; thus, gaining much of the useful real-world knowledge that is not available in a pure academic environment.

In this article, two approaches to university-industry cooperation that have been practiced at the University of Ljubljana, Faculty of Computer and Information Science, were described. Both approaches expose students to team project work, but differ in the nature of projects that students must develop. The first approach allowed students to define their own projects, thus giving them more possibilities for creativity and innovation, while less emphasis was placed on strict monitoring of the development process and complete implementation of the intended functionality. Due to cooperation with big software vendors, students were required to use their development tools.

The second approach is based on realistic requirements specification and requires full implementation of the required functionality, strictly following the prescribed development process. In this way, student projects also serve as the basis for conducting various empirical studies. Regardless of these differences, both approaches proved to be successful and make it possible to achieve a win-win situation for all participants.

REFERENCES

1. Borman, D., Should software engineering projects be the backbone or the tail of computing curricula? *Proc. 23rd Conf. Software Engng. Educ. and Training,* Pittsburgh, PA, USA, 153-156 (2010).
2. Joint Task Force on Computing Curricula, Software Engineering 2014: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. IEEE CS and ACM (2015), 17 July 2017, https://www.computer.org/cms/peb/docs/se2014.pdf
3. Tilley, S., Huang, S., Wong, K. and Smith, S., Report from the 2nd international workshop on software engineering course projects (SWECP 2005). *Proc. 19th Conf. on Software Engng. Educ. & Training,* Turtle Bay, Hawaii, USA, 87-94 (2006).
4. Smith, S., Mannion, M. and Hastie, C., *Encouraging the Development of Transferable Skills through Effective Group Project Work*. In: Uso, J-L., Mitic, P. and Sucharov, L.J. (Eds), Software Engineering in Higher Education II, Southampton, UK: Computational Mechanics Publications, 19-26 (1996).
5. Mohan, A., Merle, D., Jackson, C., Lannin, J. and Nair, S.S., Professional skills in the engineering curriculum. *IEEE Trans. on Educ.*, 53, **4**, 562-571 (2010).
6. Karunasekera, S. and Bedse, K., Preparing software engineering graduates for an industry career. *Proc. 20th Conf. on Software Engng. Educ. and Training*, Dublin, Ireland, 97-106 (2007).
7. Mahnič, V. and Grum, B., An undergraduate course in information systems technology at the University of Ljubljana in partnership with IBM. *Proc. 10th UICEE Annual Conf. on Engng. Educ.*, Bangkok, Thailand, 185-188 (2007).
8. Schwaber, K., *Agile Project Management with Scrum*. Redmond, WA: Microsoft Press (2004).
9. Mahnič, V., A capstone course on agile software development using Scrum. *IEEE Trans. on Educ.*, 55, **1**, 99-106 (2012).
10. Mahnič, V. and Časar, A., A computerized support tool for conducting a Scrum-based software engineering capstone course. *Inter. J. of Engng. Educ.*, 32, **1A**, 278-293 (2016).
11. Cohn, M., *User Stories Applied*. Boston, MA: Addison-Wesley (2004).
12. Carver, J.C., Jaccheri, L., Morasca, S. and Shull F., Issues in using students in empirical studies in software engineering education. *Proc. 9th Inter. Software Metrics Symp.*, Sydney, Australia, 239-249 (2003).
13. Carver, J.C., Jaccheri, L., Morasca, S. and Shull, F., A checklist for integrating student empirical studies with research and teaching goals. *Empirical Software Engng.*, 15, 35-59 (2010).
14. Mahnič, V., A case study on agile estimating and planning using Scrum, *Electronics and Electrical Engng.*, 5 (111), 123-128 (2011).
15. Mahnič, V. and Hovelja, T., On using planning poker for estimating user stories. *J. of Systems and Software,* 85, **9**, 2086-2095 (2012).
16. Mahnič, V. and Hovelja, T., Teaching user stories within the scope of a software engineering capstone course: analysis of students' opinions. *Inter. J. of Engng. Educ.*, 30, **4**, 901-915 (2014).
17. Mahnič, V. and Hovelja, T., The influence of diffusion of innovation theory factors on undergraduate students' adoption of Scrum. *Inter. J. of Engng. Educ.*, 32, **5A**, 2121-2133 (2016).
18. Poženel, M. and Mahnič, V., Studying agile software estimation techniques: the design of an empirical study with students. *Global J. of Engn. Educ.*, 18, **2**, 53-58 (2016).
19. Mahnič, V., The capstone course as a means for teaching agile software development through project-based learning. *World Trans. on Engng. and Technol. Educ.*, 13, **3**, 225-230 (2015).
20. Christiensen, K. and Rundus, D., The capstone senior design course: an initiative in partnering with industry. *Proc. 33rd ASEE/IEEE Frontiers in Educ. Conf.*, Boulder, CO, S2B12-S2B-17 (2003).
21. Fornaro, R.J., Heil, M.R. and Tharp, A.L., Reflections on 10 years of sponsored senior design projects: students win–clients win! *J. of Systems and Software,* 80, **8**, 1209-1216 (2007).