e-Informatica Software Engineering Journal, Volume 7, Issue 1, 2013, pages: 77–85, DOI 10.5277/e-Inf130108

Software Engineering Team Project – lessons learned

Bogumiła Hnatkowska*

*Faculty of Computer Science and Management, Institute of Informatics, Wrocław University of Technology

bogumila.hnatkowska@pwr.wroc.pl

Abstract

In the 2010/11 academic year the Institute of Informatics at Wroclaw University of Technology issued 'Software Engineering Team Project' as a course being a part of the final exam to earn bachelor's degree. The main assumption about the course was that it should simulate the real environment (a virtual IT company) for its participants. The course was aimed to introduce issues regarding programming in the medium scale, project planning and management. It was a real challenge as the course was offered for more than 140 students. The number of staff members involved in its preparation and performance was more than 15. The paper presents the lessons learned from the first course edition as well as more detailed qualitative and quantitative course assessment.

1. Introduction

Most of the Technical Universities offer a course 'Team project' for the under graduate students but rather rarely such a course is a part of final exam to earn bachelor's degree.

Polish Ministry of Education published 'Standards of education for Computer Science faculty (Bachelor studies)' in 2007 [1]. The document not only forced the Universities to offer the course mentioned above, but also gave the opportunity to treat it as a part of bachelor's thesis. The Institute of Informatics at Wroclaw University of Technology seized this chance and included 'Software Engineering Team Project' in the set of obligatory courses. The course was issued first time in the 2010/11 academic year in the winter semester. However, the course was elaborated based on experiences gained from others, previously offered, courses.

The aim of the paper is to present the lessons learned from the first course edition as well as more detailed qualitative and quantitative course assessment. The gained experiences could be helpful in organizing similar courses, and the description could serve for comparison purposes.

The paper is organized as follows. Section 2 presents the course assumptions, while Section 3 – the process of course preparation. Section 4 gives a detailed description of the course implementation. The course evaluation is the main part of Section 5. It presents valuable statistics, interesting observations and lessons learned. The last Section 6 summarizes the paper.

2. Course design

The main assumption about the course was that it should simulate the real environment (a virtual IT company) for its participants. The course was aimed to introduce issues regarding programming in the medium scale, project planning and management. It was a real challenge as the course was offered for more than 140 students. The number of staff members involved in its preparation and performance was more than 15.

The aim of the course was to develop something useful for a customer (usually a software product), and possibly to deploy it in a target environment. The course was thought as a final checking of the effects of previously offered courses, as it demanded the broad knowledge of software engineering.

According to [2] the Software Engineering Team Project was placed at the 3rd level of maturity:

Defined. Class projects are undertaken based on a defined, documented, and standardized process. The instructor relies on an established process which serves as a foundation for further process improvement. Course practices can be consistently implemented, regardless of the presence or absence of certain key instructors.

We have also implemented some practices from the 4rd maturity level [2]:

Quantitative Managed. Course projects are quantitatively measured using statistical methods. Measurements on student projects are gathered and analyzed for further improvement.

The course schedule was planned for only 10 weeks of very intensive work. A student was given 15 ECTS points for the course, what was translated into 40 hours per week of work for each course participant (9 hours at the Institute, and 31 hours for the private study). No other forms than project were involved in the course. Course design was prepared and is further presented according to the suggestions given in [3]. The description considers following issues:

- team formation,
- team supervision,
- problem statement and assignment,
- team communications,
- team assessment,
- development process.

2.1. Team formation

There are two possibilities for team formation [3]:

1. students create teams by themselves,
2. students are assigned to the teams.

We followed the first scenario, as it was easier from managerial point of view. The students were expected to organize themselves into four people teams (exceptionally, 3 or 5 people were allowed). The team was organized based on student's previous experiences. The general assumption was that the team is self-organizing with inter-changeable roles.

2.2. Team supervision

To simulate the reality, students should have a considerable amount of freedom. On the other hand, since students usually had no project experience, some amount of supervision, monitoring and guidance was needed to ensure sufficient progress and a successful result [3]. It was the responsibility of a team supervisor. Each supervisor looked after 1–4 teams (in one case 8 teams).

Project's supervisor played many roles. First of all his/hers responsibility was to formulate a project subject and its requirements (both functional and non-functional). In that way the person became a product owner. He or she was also responsible for steering the project complexity. The supervisor was allowed to reduce the scope (if it occurred unrealistic) or to extend it (if the capability of the team was greater). He/she also decided about final product assessment. Additional responsibility of the person was to give continuous feedback to the teams.

2.3. Problem statement and assignment

Each team selected a subject of the project from the list of themes, proposed by academic teachers playing the role of supervisors. Teams were also allowed to propose something interesting by themselves (the proposals were assessed by the supervisors, and after acceptance were realized). As the course was a part of the final exam, all themes had to be accepted by Department Authorities before the start of classes.

2.4. Team communications

Team members needed to communicate quite often. The responsibility of the University was to prepare the conditions for that. Depending on the type of project and the decision of the supervisor the team met at the University (orga-

nized lessons): once per week for 9 hours, twice a week for 4,5 hours or three times a week for 3 hours. In one classroom at most 4 teams were working in the same time. Each had access to computers, and a blackboard. In [3] there was a suggestion, that each team should create and regularly maintain a web page, and have documents repository.

2.5. Assessment

The supervisor was responsible for determining student's final grade. There are two common approaches to students' assessment [3]:

1. to give each team member the same grade and,
2. to give each team member a grade based on his individual contribution.

We combined both of them. In general the whole team obtained the same grade. It was offered basing on product quality, documentation quality, and the product presentation. The main focus was put on the final product itself. Its presence was a necessary condition to pass the course. The second element taken into account was the student engagement. To motivate the students to better work, the supervisor was allowed to give a student so called yellow card to indicate that his/her engagement in the project was too low, and the quality of his/hers artifacts were bad. The first yellow card decreased the final grade by 1, while the second yellow card became a red card and meant that the student failed the course.

We pondered over including peer to peer assessment component to the student's assessment recommended by many authors [3, 4], but finally we rejected this idea. We had bad experiences in using it in past projects. The students did not want to fairly judge each other. Similar results of formally conducted experiments were reported in [5].

2.6. Development process

The course assumption was that a development process should involve all necessary stages of software development from requirements to implementation and testing. A stage of project deployment was welcomed. The selected development process should fit to different projects' characteristics.

3. Course preparation

Preparation for the course was rather a long process which took about 6 months. The authorities of Institute of Informatics appointed 8 people team for that purpose. The staff members team involved experts from different domains: software engineering, data bases, computer networks, programming languages, and net administrators. The team consisted of people with experiences in providing team projects before, what resulted in some publications, e.g. [6].

The main task of the team was to develop course's recommendations including methodologies (both development and managerial), best practices, and supporting tools. It was expected that the team will also prepare educational materials for both: the faculty staff members, and students attending the course.

The task was challenging of many reasons: (1) very short time of project realization (only 10 weeks), (2) diversity of projects' subject (the recommended tools, techniques and practices must have been adjusted to all of them).

The outcomes of the course preparation process included a report, published as an internal document by the Institute of Informatics, covering all the topics mentioned above as well as some publications presented on the National Conference in Software Engineering [7], and on the International Conference ISAT [8].

The main recommendations were as follows:

– methodologies: Scrum methodology was selected as a basic one, as it fited to different kinds of projects which could be done in an iterative manner. The team considered several methodologies, both agile (Scrum, XP) and heavier ones e.g. UP, RUP. The UP methodology was recommended only for typical software development projects, for which the strong need for documentation existed. The

assumption was that each project should use at least: Product backlog, and Sprint backlog [9]. The Sprint backlog should contain tasks together with time evaluation (for at least 90% of sprint capability, about 30 h a week for each team member). The other artifacts (e.g. specifications, documentations etc.) were under supervisor jurisdiction,
- best practices: The staff team analyzed the best practices for all above mentioned software development processes, trying to define a minimal set of the most useful ones (obligatory practices). The selected practices included: iterative development (sprints), and self-organizing team with daily meetings [9]. Since each development process includes requirement specification, the requirements were presented with the use of product backlog. Requirement analysis, if it is possible, should be based on use-cases specification. The coding standards should be used at implementation stage. The other recommended (but not obligatory) practices included: source code sharing, TDD development, and continuous integration,
- tools: Team Foundation Server (TFS) [10] was selected as a primary supporting tool, mainly because the University obtained professional support from the Microsoft Polska Company; the tool allows to plan releases, sprints, and tasks according to Scrum methodology; it offers also other useful capabilities, as source code management system, or continuous integration support. It integrates well with Microsoft Visual Studio and good enough with Eclipse platform. Moreover, it offers a web site for each projects, and is a good mean for communication purposes (e.g. Wiki). The functionality is available by the Internet. Students were also allowed to use virtual machines to deploy their solutions.

Before running the course, the staff team presented the assumptions about it to all members of the Institute, e.g. organization details. Moreover, a training covering the usage of the Team Foundation Server in the context of Scrum methodology was conducted.

4. Course implementation

The course was attended by 37 groups (146 students) supervised by 17 people. Each supervisor, as a product owner, set a number of releases, a number of sprints, and a length of sprints. Mostly, the projects had one release (66%), or two (32%). One project had three releases (2%). Sprints lasted 7 days (89%) or 14 days (11%). The capacity for 7 days iterations was calculated usually for 30h/iteration/person (10 teams) or 40h/iteration/person (27 teams) – it was under supervisor jurisdiction.

The subjects were very diverse, e.g. Conference management system, Team competition management system, System for collection and analyzing samples of handwriting, Virtual campus of Wroclaw University of Technology, Collision-free motion of a group of autonomous vehicles – an algorithm using the Webots environment. A few subjects came from industry, e.g. System for support decision making based on data collected in a data warehouse. The results of some projects are visible in the Internet: [11–13].

5. Course assessment

5.1. Quantitative assessment

To compare different teams, and to find out existing shortcomings a quantitative assessment of the course was done. The measure values were taken from the TFS (product backlog, and sprint backlog) which was the obligatory managerial tool for all teams, and their supervisors. However, one team of unknown reasons did not use the TFS, so it was excluded from further consideration. The research questions were formulated as follows:
- What was the number of product backlog items (PBI), and sprint backlog items (tasks)?
- How many hours did the teams spend on tasks realizing during the whole project?
- What was the coverage (in percent) of the declared capacity?

Statistic	The number of PBI	The number of tasks	The number of work hours	Capacity coverage [%]
Average	22,68	128,33	792,50	0,77
Minimum	4,00	32,00	40,00	0,04
Maximum	89,00	313,00	1698,00	1,77
Standard deviation	19,53	68,76	375,76	0,36

Table 1. The basic measures for projects obtained from TFS tool

I wanted to establish the parameters of a 'typical' project which could serve as a reference for other projects. The answers for the questions are presented in Table 1.

The number of PBI fluctuated from 4 to 89, with the average 22,68. Lower numbers of PBI usually meant that the granularity of them was bigger. In many cases where the PBI number was below 10, a development process was rather a waterfall than iterative one with PBI elements representing requirement specification, analysis, design etc., and the tasks associated with them were done in many sprints.

The number of tasks for a project ranged from 32 to 313 (the average was equal to 128,33). Lower number of tasks usually meant that they were very long (sometimes they were estimated for more than 30h) or that most of the tasks were not documented in the TFS (for a team with 32 tasks, the coverage of the declared capacity was 23%, and for another team with 37 tasks, the coverage was only 0,04%). A team with the highest number of tasks (313) defined – during the whole project – a lot of small (2–4 hours) tasks. The coverage of the declared capacity fluctuated from 0,04% (for a team that defined only 37 tasks for 40 hours) to 170% (for a team with the tasks estimated for 1698 hours). Probably, in the former case, the team did not use the TFS tool, while in the later case, the duration of tasks were overestimated; half of them lasted more than 8 hours, many – 20 hours. The average of the capacity coverage was equal to 0,77%. The number of teams with the capacity in the range [80%;120%] was equal to 17, only 2 teams defined the tasks for more than 120%.

The additional research questions were about other good practices the teams could use, mainly, how many teams used the source control version tool, integrated with the TFS? (20 teams, 54%), how many teams registered bugs in the TFS? (8 teams, 21%, while only 5 had more than 3 bugs), how many teams used automatic tests? (5 teams) It should be mentioned that the data were collected only from the TFS tool. From the qualitative assessment of the course I know that some teams used another system of version control, e.g. Git.

Because in [3] there is a suggestion to make available some projects as good examples, I tried to classify projects (product bakclog, sprints, strpint backlog) stored into the TFS into three groups of quality: (1) target (2) minimal (3) non-accepted.

To do that a checklists with disqualifying questions were defined. Below the checklist to disqualify a project to have a target quality is presented:

1. PBI not associated with iterations,
2. Tasks not associated with PBI items, or some sprints without the tasks at all,
3. Tasks without time evaluated,
4. Very long tasks – lasted more than 30h,
5. Mistakes in PBI and/or tasks descriptions, e.g. many repetitions of the same PBI definition,
6. PBI realized in waterfall manner (tasks associated with PBI were realized during many sprints),
7. Capacity coverage outside the range [70%; 130%],
8. States of sprints and tasks not changed properly.

The elements disqualifying a project to have a minimal quality are as follows:

1. 1–5 as for target quality,
2. Capacity coverage outside the range [50%; 150%].

According to the rules defined above we had 6 projects with target quality, 17 – with mini-

Statistic	Course usefulness	Quality of supporting materials	Course organization
Average	4,66	3,37	4,32
Minimum	2,00	1,00	1,00
Maximum	5,00	5,00	5,00
Standard deviation	0,65	0,96	0,82

Table 2. The assessment of course components steaming from closed questions

mal quality, and 14 – with non-accepted quality. The number of bad projects seems to be too big, but I hope that in the next edition this number will decrease as there are some good examples.

The conclusion from the facts presented above is that the supervisors need to better monitor the usage of the TFS tool. The limits (min, and max) for the number of hours for a person/week must be defined and obeyed – the recommended value is 30. The maximal duration for task estimation (8 hours) must be strongly obeyed.

5.2. Qualitative assessment

To perform qualitative assessment of team project a questionnaire for the students were prepared. The questionnaire questions were as follows:

1. (Closed question): Assess the following elements in the scale from 1 (the worst mark) to 5 (the best mark):
   - Course usefulness: ...
   - Quality of supporting materials: ...
   - Course organization: ...
2. (Open question) Which elements of the course were the most valuable?
3. (Open question) Which elements of the course need to be improved/changed in the next issues?

The questionnaires were filled after the final exams, so telling the truth, we asked bachelors about their opinion. The questionnaires were available in two forms: paper and digital one (published in the Internet). We gained the answers from 93 students from 146 who managed to finish the course (what gives 64% response rate). The answers for the closed questions are presented in Table 2.

The grades for all components were rather high – the highest for course usefulness, the smallest for course supporting materials. It is something that should be improved in the next edition.

The questionnaire consisted mainly of open questions. It was a challenge to analyze them. I had defined some synonyms that were present in different answers, and next counted the numbers of people, who used these synonyms in their answers. The results are presented below. The threshold, below which the answers are not presented, was set to 10.

Figure 1 presents how many people pointed out a specific element to be the most valuable.

As it is easily to observe, the team work (39 answers) and agile methodology (38) were perceived as the best choices. An opportunity to acquaintance the new technologies (17) and to put different pieces together (to build the whole project from the beginning to the end – deployment, 17 answers) was inspiring experience. The students appreciated also regular meetings as the means that motivated them to hard work (14) – they needed to answer 3 questions at the beginning of each meeting (what was done, what problems were encountered and what is going to be done to the next meeting). The team members also thanked for interesting themes that allowed them to deeply involve to the project (11).

Of course not all elements were the full success. Figure 2 presents the elements that need to be improved.

First of all the students complained about the short time of the project (24 answers). The Project Management Course was run in parallel with the team project. The students reasonably suggested to move Project Management Course one semester back. They also found TFS environment quite difficult and suggested to do a

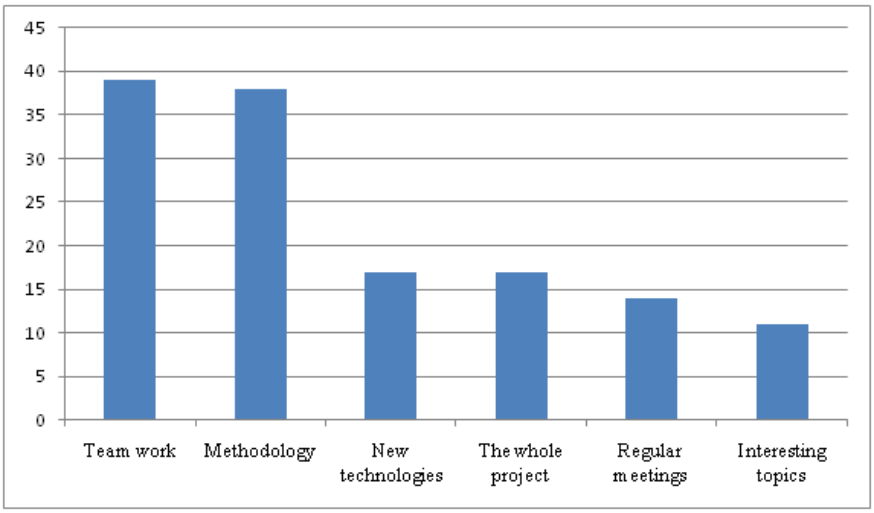

Figure 1. Number of answers for the questionnaire question nr 2

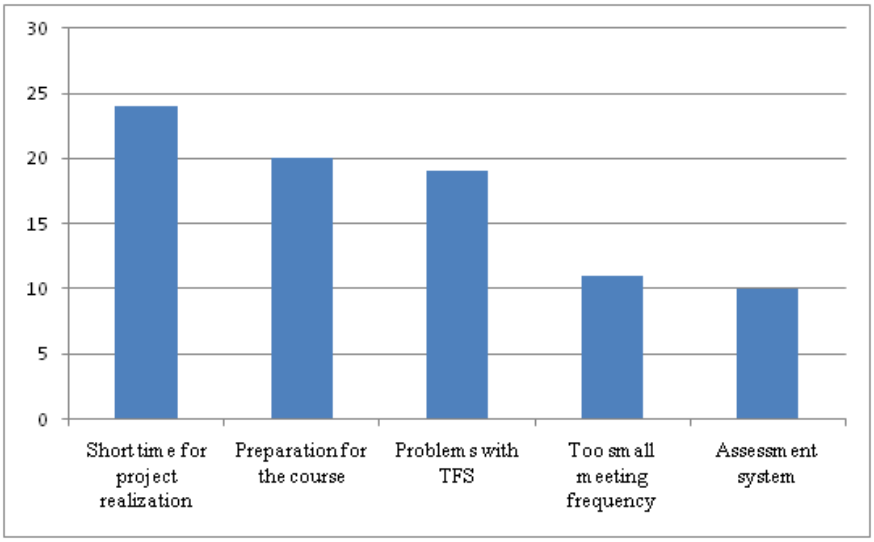

Figure 2. Number of answers for the questionnaire question nr 3

training before vacations (20). The access to the TFS was not such efficient the students expected (19). The students wanted also to meet more often (this opinion was given by the team members who had organized meeting only 1 per week). The students suffered a little from not clear enough assessment system (10).

The same questions were asked to the academic teachers playing the role of projects' supervisors. But the results were very similar to those obtained from the students.

Below a list of possible improvements is presented.

Course preparation:

- Project Management course should be moved to the 6th semester,
- Training of the tools used within a course should be provided in the 6th semester of during vacations,
- Supporting materials should be improved.

Supporting tools:

- Performance parameters (throughput, response time) of the TFS should be better,
- Opportunity of usage of another popular supporting tool should be considered.

Course organization:

– Give the students a longer time for their own topics formulation,
– Attract more topics from the industry,
– Make an assessment system more transparent.

6. Conclusions

The Software Engineering Team Project starting from 2010/11 academic year is a compulsory subject to all students of Informatics at Wroclaw University of Technology. Many universities offer a team engineering course as a part of the studies, e.g. [3, 14–17], but there are some important differences between them and our proposal:

– Course outcome: system prototype versus a working release of a system, often with its installation version and a user manual,
– Focus: educational aims versus put things together (part of the final exam),
– Methodology: due to documentation reasons rather heavier methodologies versus as light methodology as possible,
– Accompanying courses: lecture or seminars versus none.

We found the first course edition as a success, however some elements could be improved. The weakest part of the course was the assessment. More strict rules for it should be defined, especially taking into account the number of working hours. We are going to introduce an obligatory presentation of the project before the members of faculty staff (the last time it was under a supervisor jurisdiction). We are also going to reward the best project with the highest grade (5.5). At that moment we are preparing to the second edition of the course for more than 160 students, e.g. a new version of training materials is under development. The number of topics coming from industry is a little bit higher than the last year. We hope that we manage to eliminate shortcomings and not to lose good points.

References

[1] Rozporządzenie ministra nauki i szkolnictwa wyższego z dnia 12 lipca 2007. (2007). [Online]. http://www.bip.nauka.gov.pl/_gAllery/21/97/2197/20070712_rozporzadzenie_standardy_ksztalcenia.pdf

[2] J. Collofello and C. H. Ng, "Assessing the process maturity utilized in software engineering team project," *Journal of Engineering Education*, Vol. 90, No. 1, 2001, pp. 75–78.

[3] M. Bielikova and P. Navrat, "Experiences with designing a team project module for teaching teamwork to students," *Journal of Computing and Information Technology*, Vol. 13, No. 1, 2005.

[4] N. Clark, P. Davies, and R. Skeers, "Self and peer assessment in software engineering projects," in *Proc. of 7th Australasian Computing Education Conference (ACE 2005)*, D. T. A. Young, Ed. CRPIT, 2005, pp. 91–100.

[5] N. Herbert, "Quantitative peer assessment: Can students be objective?" in *Proc. of 9th Australasian Computing Education Conference (ACE 2009)*, S. Mann and S. Mann, Eds. CRPIT, 2009, pp. 63–71.

[6] I. Dubielewicz and B. Hnatkowska, "Improving software development process implemented in team project course," in *Proc. of the 8th International Conference on Computational Science, Part II.* Berlin: Springer-Verlag, 2008, pp. 687 – 696.

[7] ——, "Praktyki w inzynierii oprogramowania – perspektywa pracy zespolowej," in *Inzynieria oprogramowania w procesach integracji systemow informatycznych*, C. O. Janusz Gorski, Ed. Gdansk: Pomorskie Wydawnictwo Naukowo-Techniczne PWNT, 2010, pp. 121–228.

[8] ——, "Best practices in students team projects," in *Information systems architecture and technology: IT models in management process*, Z. Wilimowska, Ed. Wroclaw: Oficyna Wydawnicza Politechniki Wrocławskiej, 2010, pp. 487–497.

[9] R. Levin. Understanding scrum – best practices guide. (2010). [Online]. http://www.brighthub.com/office/project-management/articles/68791.aspx

[10] J. Meier, J. Taylor, P. Bansode, A. Mackman, and K. Jones. Team development with visual studio team foundation server. (2007, Sep). [Online]. http://msdn.microsoft.com/en-us/library/bb668991.aspx

[11] Collision-free motion of a group of autonomous vehicles – an algorithm using the webots environment. (2011). [Online]. http://www.youtube.com/watch?v=tzm2uU8_nQA

[12] Laboratory of the distributed computer networks portal. (2011). [Online]. http:

//156.17.130.12/Main.aspx

[13] Virtual campus of wroclaw university of technology. (2011). [Online]. http://www.ii.pwr.wroc.pl/WirtualnyKampusPWr/index.html

[14] D. Delaney and G. Mitchell. Tutoring project-based learning: a case study of a third year software engineering module at nui. (2005). [Online]. http://www.aishe.org/readings/2005-2/contents.html

[15] G. Dobbie and G. Bartfai, "Teaching software engineering in a computer science department," in *Proc. of International Conference Software Engineering: Education and Pracitce*, Dunedin, New Zealand, 1996, pp. 58–63.

[16] Unit of study engg1805 professional engineering & it. (2011). [Online]. http://sydney.edu.au/engineering/it/~engg1805/Documents/ENGG1805CourseOutline.pdf

[17] M. Zaigham, "A framework for software engineering education: A group projects approach," *International Journal of Education and Information Technologies*, Vol. 1, 2007.