Advanced Hands-on Training for Distributed and Outsourced Software Engineering

Martin Nordio
ETH Zurich
Clausiusstrasse 59
8092 Zurich, Switzerland
+41/44-632-0297
Martin.Nordio@inf.ethz.ch

Roman Mitin
ETH Zurich
Clausiusstrasse 59
8092 Zurich, Switzerland
+41/44-632-7290
Roman.Mitin@inf.ethz.ch

Bertrand Meyer
ETH Zurich
Clausiusstrasse 59
8092 Zurich, Switzerland
+41/44-632-0410
Bertrand.Meyer@inf.ethz.ch

ABSTRACT
Today's software projects are often distributed across multiple locations. This distribution poses new challenges produced by the cooperation across different countries, times zones, and cultures. Software engineering courses have to prepare students accordingly. This paper reports an experience on teaching a distributed software engineering course. In this course, students develop software in collaboration with five universities located in Italy, Hungary, Russia, Switzerland, and Ukraine. The projects allow students to face the difficulties of developing software in a globalized context, and provide a practical experience on distributed software engineering. We describe the major obstacles on organizing such a course, and we suggest best practices to achieve successful outcome.

Categories and Subject Descriptors
D.2.9 [**Software Engineering**]: Management – *life cycle, productivity, programming teams, cost estimation, software quality assurance.*

General Terms
Management, Documentation, Performance, Human Factors.

Keywords
Distributed software engineering, multinational project, teaching.

1. INTRODUCTION
Developing software engineering projects is difficult. Distributing the team across different countries, continents, time zones and cultures does not help to solve the problem; it makes it harder. Due to the restriction in communication, software requirement specifications, interface specifications, and project management become a key to develop successful distributed projects. Failures reported in outsourced and distributed projects happened not due to lack of technical expertise, but due to lack of proper management. One of the roles of software engineering courses is to prepare students to face the new challenges of distributed software engineering.

ICSE'10, May 2–8, 2010, Cape Town, South Africa.

Most of today's software engineering courses include a student project as key component. The project provides a practical experience for the students. Distributed software engineering courses should also include a project, in this case a distributed project. To provide a real experience the teams in the distributed projects should be located in different countries with different cultures, native languages and if possible, different time zones. How can one organize these projects?

Some of the difficulties of organizing these projects, besides the time schedules of the courses, which can differ from one country to another, is the coupling between the teams: the success of the project does not only depend on the success of a local team but also the success of the team located in another country. This coupling does not exist in typical software engineering projects where a project is developed by one team. Another problem is the integration of the projects in a final system: even if the local teams perform a great job, the project might still fail.

This paper describes a practical experience teaching distributed software engineering. This experience recreates the atmosphere of an international project, and faces some of the severe obstacles that may come up in a real project. Some aspects of the course have been described in previous publications[7][5]; the novel elements in this paper are: (1) a description of the major obstacles to organizing distributed projects, (2) an assessment of the results, and (3) best practices to achieve successful results. The paper is organized as follows: in Section 2 we describe our software engineering course. Section 3 presents the assessment. The paper concludes with lessons learnt.

2. THE COURSE
The Chair of Software Engineering at ETH Zurich has taught a *"Distributed and Outsourced Software Engineering"* course (DOSE) for several years. The course targets master students with good experience in programming and some prior knowledge in software engineering. In the last two years, the course has incorporated a distributed project in collaboration with Politecnico di Milano (Italy), Odessa National Polytechnic University (Ukraine), the State University of Nizhny Novgorod (Russia), and University of Debrecen (Hungary). While the organization of the courses is local to each university, the project is shared.

As a result of this scheme, the students get experience in the development of true distributed projects. Although the size of the

projects is small as compared to an industrial project, students face the same difficulties as in a distributed project in industry.

2.1 Lectures

While each participating university was teaching its own software engineering course, the course material was shared. Furthermore, the lectures at ETH Zurich were recorded and were available for all students. The course had two hours of lectures and one hour of exercise sessions per week.

The topics covered in the course are requirements elicitation and writing software requirement specification, the CMMI model, quality assurance, cost models for outsourcing, supplier agreements, and risk management in distributed projects.

2.2 Project

The key component of the DOSE course is its distributed project. Since distributed projects pose new challenges, the topic of the project should not be a complex system; however, it should be interesting enough for the students. In DOSE2008 [1], the project was chosen from the topics offered in the SCORE [9] competition: "BTW" [10], a system to provide advice to someone planning a trip to a city. We divided the project into three clusters[1] to be handled by different teams within a group[2]; the BTW clusters were:

- SYST: GUI and overall organization of the system
- GEO: Interface with GIS information
- PLAN: Route planning and advice

Each project group includes teams from different universities. Specifically each group in DOSE 2008 was made of three teams, each including two students from a given university. Typical group configurations were:

- Zurich – Nizhny Novgorod – Milano
- Debrecen – Milano – Zurich
- Milano – Zurich – Odessa

This approach has a pedagogical benefit: it forces the teams, in their work and especially their interactions with other teams, to focus on interfaces (in the sense of program interfaces, also known as APIs). This is a key software engineering concept; the best way to teach it is by experience, as students discover the essential role of high-quality interface descriptions, in particular contracts, and realize the extreme degree of precision required to avoid mishaps. The pedagogical value is higher and the lessons deeper than what can be learned from the experience of implementing someone else's blueprint in the process-based approach (although that experience is also useful).

2.3 Schedule

The course had the duration of 14 weeks (from September to December 2008), devoting 13 weeks for the project. The nature of the project makes it desirable to have other participating universities adhere as closely as possible to this schedule. This is difficult to achieve as other universities have their own calendars. This is one of the reasons of the collaboration with Milano, Nizhny Novgorod, Odessa, and Debrecen: their academic calendar is similar to ETH calendar.

The project consisted of the following three phases:

- Phase 1: Elaborate requirements document per component (4 weeks)
- Phase 2: Revise requirements documents and create consolidated document and develop interface specification (5 weeks)
- Phase 3: Implementation (4 weeks)

In the first phase, besides elaborating the first version of the requirements document, the students have prepared a communication plan. The task consisted of making the first communication with all members of the group, and agreeing on a time slot for weekly meetings. The requirement for the weekly meetings was that at least one member of each team had to participate in the meeting. This task forced students to start the communication from the beginning of the project, and it helped to avoid possible misunderstandings.

The second phase was devoted to revise the requirements documents. The documents were reviewed by all members of the group, and feedback was integrated. Furthermore, students developed interface specifications for their clusters. To avoid misunderstandings and ambiguities of the requirements document, the interface specifications were written in Eiffel using contracts (our previous work [7] shows a study of the use of contracts in distributed projects). The last phase was the implementation and testing of the system.

In 2007 (DOSE2007 [2]), we ran a similar experience with distributed projects. We found out that students did not start the assignments after the first week. This late starting of the assignments produced integration problems: the assignments were finished closed to the deadline leaving no time to solve integration problems. To solve this problem, last year, each phase was implemented in two cycles; thus students had to submit an assignment every second week.

2.4 Technology

The analysis, design and implementation of the projects are done in Eiffel, using the open-source EiffelStudio environment. As Students in several of the universities involved did not know Eiffel in advance, thus ample teaching material was available from the teaching pages at the Chair of Software Engineering's site se.ethz.ch and also from Eiffel Software. Learning Eiffel was not a difficulty for the students, however, we require knowledge of an object-oriented language. One of the reasons of the use Eiffel is its integrated Design by Contract mechanism. Last year projects have shown that contracts help the development of distributed projects [1].

To host the student projects, we use the Origo platform [8] as a general-purpose open-source hosting framework. Origo provides forums for discussions, Wiki pages, configuration management support (Subversion) etc.

[1] A *cluster* is a part of the project implemented by one team.

[2] A *group* does the full project and is made of teams, each doing a part of the project; A *team* is made of students from one university, but a group involves teams from different universities

3. ASSESSMENT

3.1 Project outcome

The results of the project were good: the teams integrated the subsystems and a final project was working by the end of the semester. Although some functionalities of the project were missing, the requirements with high priority were implemented.

At the beginning of the semester, there were six projects (about 50 students distributed in the five universities): five projects were implemented and successfully integrated; and one project failed. On average, the implemented projects had 44 classes, and 4789 lines of code (the projects were developed by 6-7 students).

3.2 The problem of using a distributed project

There are several problems when one tries to develop distributed projects: lack of communication, inconsistent shared vision, cultural differences, different interest in the project, weak commitment of certain teams in a group etc. However, one of the most important problems to solve is the coupling of the subsystems: if a subsystem fails, the whole project will fail.

In 2007, we experienced difficulties with teams leaving the project. Some of the reasons are lack of motivation or inability to cope with the language barrier. For example, in Russia one student was excited about the project in the beginning but, due to poor language skills, decided to leave the project. As a result one group lost a team and the critical component this team was working on. To solve this problem, we had to reorganize the groups.

Students need a sufficient level in English to participate in the project. In Russia and Ukraine, not all students speak English. To avoid communication problems, students in Russian and Ukrainian universities had to pass a language test before joining the course.

At Politecnico di Milano, the DOSE project is a selective part of a large software engineering course (with more than 100 students). Since the project is taught, students had the option to implement a distributed project or a local project. Many students in Milano were interested in the distributed project, and the best applications were selected to participate. This selection produced an excellent result from the students at Politecnico di Milano: no student left the project, and the quality of requirements documents and the implementation was very good. In the other universities, there was at least one student leaving the course either due to language problems or personal interest.

3.3 Students' feedback

During the projects, we have collected students' feedback. This empirical data is not statistically significant; it only characterizes our experience. In the following sections, we present the results.

3.3.1 Motivation

Keeping the students motivated and excited about the project is important for all software engineering projects. Before starting each phase (1-requirements, 2-interface specification, and 3-implementation), students filled in a questionnaire about how motivated they were. They were asked to give a number from 1 to 10 where 1 is no motivation and 10 very motivated. The questionnaire was filled in by 100% of the groups (the questionnaire was part of the assignments, so all groups reported their motivation). In average, the motivation was very high (8.2) at the beginning of phase 1. Before phase 2 and phase 3, motivation was still very high: 8.1 and 8.0 respectively. We think challenging projects help to enhance the motivation of students.

3.3.2 Expended Time

Some interesting questions to ask are:

- How much time did the teams expend in each phase?
- How much time did they expend in communication?

Table 2 shows the expended time in the whole phase 1 and phase 2, and the time expended only in communication. This data represents the 78% of the projects; unfortunately, the collected data for phase 3 is not representative and it is not presented in the table.

The average of time expected in phase 1 is 30 hours; the average time used for communication was 14 hours, which represents 47% of the time used in phase 1. This high percentage is because in phase 1, students start the collaboration; they exchange contact information, they discuss how to organize the project, how to split the tasks, they schedule the meetings, etc.

The percentage of the average of communication time was reduced in phase 2: 40%. The average of hours expended in communication is 43 hours, and the average of communication time is 18 hours.

We think there are two main reasons why the time expended in communication is high. The first one is that phase 1 and 2 are the critical phases of the project: a misunderstanding in these phases might result in ultimate failure of the project. In phase 2, students discuss the interface specification. This interface specification is important for the integration of the project.

The second reason is that projects are distributed in three locations. All discussions have to involve three different countries (in some cases with different time zones). We do not have concrete evidence of the overhead in communication comparing distributed projects in two and three locations; we plan to make a study during the development of next DOSE course.

3.3.3 Tools for communication

Another important part of the students' feedback was to collect information about the tools they used to communicate. Most of the students used Skype, e-mail, wiki, and Google docs. There was a small group using Google groups and Basecamp. The average of the tools is: 63% Skype; 26% e-mail; 8.5% e-mail; 1.5% Google docs; 1% other. This information was collected three times during the project, and it represents 95% of the projects.

Students reported that most of the Skype communication was done using only chat. About 60% of the teams did not used voice for communication. The reason was that the Internet connection, especially in Russia, Hungary, and Ukraine, was slow, and they were not able to use voice communication.

3.3.4 Other problems

The main problems during the project were about communication. Some teams reported that the level of English of the other teams was poor, making the communication harder.

There were some difficulties about project management: some teams had problems deciding how to split the work in different cluster, and agreeing the scope of the clusters. In the first phase of the projects students were asked to prepare a communication plan,

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	**Av**
Total time in Phase 1	17h	24h	30h	56h	58h	25h	52h	21h	26h	9h	50h	19h	14h	17h	**30h**
Communication in Phase 1	12h	15h	12h	30h	18h	8h	30h	6h	10h	5h	30h	7h	8h	7h	**14h**
Perc. in communication	*70%*	*63%*	*40%*	*53%*	*31%*	*32%*	*57%*	*28%*	*38%*	*55%*	*60%*	*36%*	*57%*	*41%*	**47%**
Total time in Phase 2	38h	55h	-	80h	70h	22h	60h	14h	20h	30h	-	100h	21h	13h	**43h**
Communication in Phase 2	15h	16h	-	30h	45h	4h	15h	6h	12h	12h	-	50h	12h	3h	**18h**
Perc. in communication	*39%*	*29%*	-	*37%*	*64%*	*18%*	*25%*	*42%*	*60%*	*40%*	-	*50%*	*57%*	*23%*	**40%**

Table 1. Expended time in phase 1 and phase 2

but absence of scope management plan was an additional obstacle for the SRS phase.

4. CONCLUSIONS

One of the lessons learnt is how to organize distributed projects in an academic environment. If it is the first time the course includes a distributed project, it is strongly recommended that the teaching stuff at the different universities meet to discuss the organization of the course. Important items to discuss are the project topic, the programming language to use, communication tools, set up for *svn* or *cvs* repositories, and student assignments and deadlines. We find useful to provide a scope document and a basic project implementation. The scope document should describe the overall project structure, and the interaction between the groups.

Organizers in each university should ensure that students commit to the project. In many universities students can leave the course at any time; this is strongly undesirable for a distributed project. We have addressed this problem working with small groups of volunteer students. In some universities, students are selected. If students leave the course during the project, we reorganize the groups.

To implement the project, we recommend that a common programming language is chosen. This language can be also used to define a common interface specification between the subsystems. If the students are not familiar with the programming language to use, it is useful to organize an introductory training before the project starts.

Communication in distributed project is difficult. However, it is not enough only to describe the problem of communication in a lecture. Before starting the projects, we run several communication exercises, and we discuss the results in an exercise session. For example one good game can be found in student books for the official Microsoft MSF course. Furthermore, we require students write a communication plan.

5. ACKNOWLEDGMENTS

We would like to thank all the people involved in DOSE 2008 especially Dr. Peter Kolb, Prof. Carlo Ghezzi, Prof. Elisabetta Di Nitto, Giordano Tamburrelli, Prof. Viktor Gergel, Andrey Zaychikov, Lajos Kollar, Prof. Juhasz Istvan, and Prof. Victor Krissilov; to the students who worked hard and gave us useful feedback.

6. REFERENCES

[1] DOSE 2008. http://se.ethz.ch/teaching/2008-H/dose-0273/index.html

[2] DOSE 2007. http://se.ethz.ch/teaching/2007-F/outsourcing-0273/index.html

[3] M.J. Hawthorne and D.E. Perry. Software engineering education in the era of outsourcing, distributed development, and open source software: challenges and opportunities. In International Conference on Software Engineering. 2005.

[4] J.D. Herbsleb and D. Moitra. Global software development. Software, IEEE, 2001.

[5] Meyer B., Piccioni, M.: The allure and Risks of a deployable software engineering project. In proceedings of the 21st IEEE-CS Conference on Software Engineering Education and Training. 2008.

[6] Richardson, I., Milewski, A., Mullick, N., and Keil, P. Distributed development: an education perspective on the global studio pro ject. In ICSE '06. 2006. ACM.

[7] Nordio, M., Mitin, R., Meyer, B., Ghezzi, C., Di Nitto, E., and Tamburelli, G.: The Role of Contracts in Distributed Development in Proceedings of Software Engineering Advances For Offshore and Outsourced Development. 2009.

[8] Origo. http://www.origo.ethz.ch/

[9] SCORE, http://score.elet.polimi.it/

[10] BTW Project, http://score.elet.polimi.it/projects.html