Performing a project in a Distributed Software Development Course: Lessons Learned

Federico Ciccozzi
Department of Innovation, Design and Engineering
Mälardalen University
Västerås, Sweden
Email: federico.ciccozzi@mdh.se

Ivica Crnković
Department of Innovation, Design and Engineering
Mälardalen University
Västerås, Sweden
Email: ivica.crnkovic@mdh.se

***Abstract*—Distributed software development approaches have to face with several issues like cultural differences, collaboration and communication mechanisms, which can undermine the overall development success if not handled in a proper manner. In order to provide a real environment for students placed in different countries to learn and apply the best practices in distributed software development, a course has been developed jointly by two european universities. The course aims at providing the students an insight in the complexity of distributed development and giving the possibility to work in distributed teams for actual implementations, in order to minimize the gap between theory and practice. This paper describes the course design, challenges, results and success factors, from a students perspective.**

***Keywords*-Global Software Engineering; Distributed Software Development; DSD GPXCleaner; DSD Course**

I. Introduction

Global Software Engineering (GSE) is based on Distributed Software Development (DSD) which provides several benefits such as a broader area, from which an higher number of skilled personnel and resources in general can be gathered and enrolled, as well as lower costs derived from outsourcing. In addition to the common well-known problems related to standard software engineering, in distributed software development other issues and challenges arise, especially in the establishment of communication between the distributed teams, caused by cultural, time-zone and language differences. Exchange of information and communication between distributed developing teams must be effectively and efficiently handled in order to achieve an harmonious and smooth development process and avoid costly delays in time-to-market.
GSE requires additional skills which are usually not built-up in software engineering courses. For this reason there is a need for having dedicate courses specifically addressing the issues related to GSE. Mälardalen University (MDH) in Västerås-Sweden and Faculty of Electrical Engineering (FER) in Zagreb-Croatia have jointly developed a DSD project course [1] in the academic year 2003/2004, which, due to its successful results have been run every year, and is currently at its seventh edition. The main goal of the course is to train students in a real distributed development environment by giving theoritecal notions to be applied in an actual distributed implementation project, which gives a unique experience for the students to encounter and overcome challenges coming from a real project. Students from the two institutions are expected to work together on a software system implementation in distributed teams. A certain number of lectures gives basic notions on distributed software development as such, since the DSD course is strongly focused on the actual development with submission, presentation and evaluation of deliverables by each project team. In the academic year 2008/2009, the best developed projects participated to the selections for the SCORE competition [2], organized by ICSE'09 in Vancouver, Canada. Out of 50 project teams from prestigious universities, six finalist teams were selected, and three of them, including the GPXCleaner project, were developed within the DSD course. In most of the cases educational papers report experiences in teaching from a teachers' point of view, but this paper aims at reporting such experience from a students' point of view starting obviously from the course design, but focusing on challenges, results and success factors from the DSD GPXCleaner project's perspective. In section 2, related work is presented and section 3 describes DSD course design, settings, goals and challenges. In section 4, both technical and non-technical challenges, selected approach and achieved results of the DSD GPXCleaner project are given in order to motivate lessons learned and success factors individuated from the project perspective and described in section 5. The paper is concluded by the section 6.

II. Related Work and Motivation

The spread of outsourcing and other distributed development approaches in industry generates new challenges to be addressed by evolving software engineering curricula [3]. These challenges have been widely studied in literature and a systematic review of such can be found in [4]. The need of developing new courses to form a new generation of software developers with an insight of globalization and distributed/collaborative development arises; in

[5] the Distribute Software Engineering Laboratory course and derived lessons learned are described. A similar course, but rather focused on the importance of rapid reaction to unpredictable factors in distributed settings, is named Global Studio Project and described in [6]. The purpose of this work is giving the perception of such a course from a student perspective, in order to describe the actual challenges to be faced by the main actors, i.e. the students, in developing a project in multicultural distributed settings.

III. DSD Course Design

The course has been planned and developed in the academic year 2003/2004 by a collaboration MDH and FER. During the planning phase, a time-based model (fig. 1) was created in order to describe a way to correlate instructional and course design of the course to the student projects results. The model is composed by two correlated submodels [7]:

- *Instructional Design Model* - describes the teaching perspective and focuses on the process intended to be followed in order to build an effective teaching approach, using the ADDIE model [8] including the phases (i) analysis, (ii) design, (iii) development, (iv)implementation and (v) evaluation.
- *Software Development Model* - describes the student projects development through the following phases: (i) requirements analysis and team forming, (ii) concept presentation, (iii) detailed design, (iv) development, (v) QA testing and (vi) release and maintenance.

The main idea was to individuate a correlation betweeen instructional and course design and results of the student projects which achieve the best results.

In fig.1, on both plains, the three cycles represent the related deliveries of the course: (i) course planning, (ii) first semester, (iii) second semester. Initial instructional design was carried out a semester prior to the first delivery by a graduate student from MDH, whom spent a semester in Zagreb for planning and creating the course with his Croatian counterparts at FER. When the Instructional Design Model reaches the analysis phase, the results of the students' software development projects are evaluated in order to determine how they can make a meaningful contribution to the instructional and course design for the next semester. Later, while deciding on future student projects during course delivery, Instructional Design cycle starts affecting software development cycle, setting priorities for future software development projects, team formations and inventive role-playing [7].

Further the main risky challenges, whose inefficient management may undermine the course's success have been considered:

- *Work environment:* it is not easy to reproduce the conditions of a real distributed project, mainly due to the limitations coming from the academic environment
- *Resources limitation:* another academic limitation is the availability of finances and equipment, whose usage has to be maximized
- *Time limitation:* the course lasts 11 weeks, which is obviously a short period compared to real-world distributed developments. In this short period both instruction and assistance to the projects development has to be provided, taking into account the harmonization between the two different institutions
- *Projects management:* if badly handled, may lead to wrong-directed effort by the students with consequent possible incomplete implementations as final result

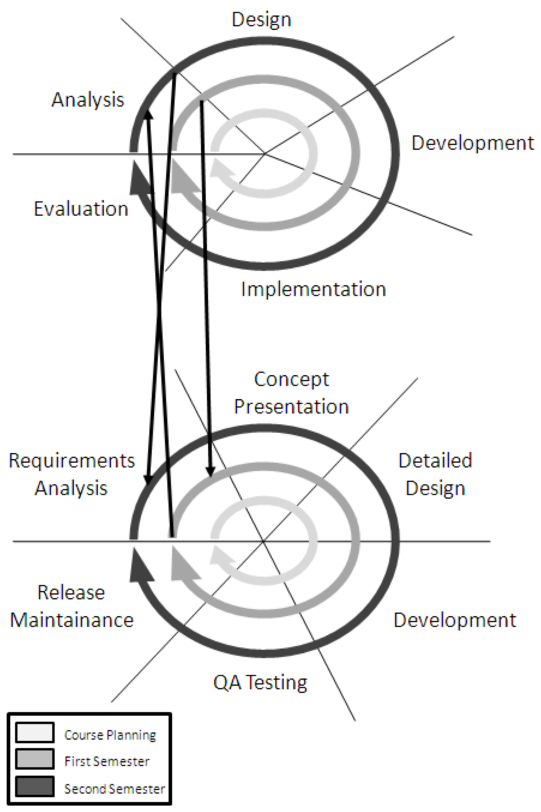

Figure 1. Instructional and Software Design Plains

The two institutions succeeded in providing a complete work environment with a well-assorted set of resources in order to allow the students to exclusively focus on the projects. Time limitation and misleading projects management could also invalidate students and teachers effort. Therefore the first phase of the course focused on students individual research of the technology intended to be used in the projects and presentation of the results; this phase brings two advantages: acquisition of basic knowledge on the technologies and increase of students' confidence in communicating in front of others in English, not mother-tongue for all the students. Afterwards, the available projects are presented during joint distributed lectures, through videoconferencing, in order to collect students' preferences about the project to develop and, considering them, build up the project teams. A project contractor is assigned to each project with the duty of defining the characteristics of the product by communicating with the project leader for defining the project demands, a project plan, monitor the development and finally provide a product evaluation. Each project team is composed by students from the two institutions in a fairly equal proportion, and two leaders are selected in order to lead each sub-team: a team leader, whose task is to coordinate the communication among the local sub-team members and between the two distributed sub-teams, and a project leader, who acts in fact as a team leader for his/her sub-team, but in addition has to coordinate the work of the whole project team, communicating with the remote team leader and with the project contractor. A core activity in a distributed students project development is monitoring what has been achieved through two mechanisms: (i) weekly reports and (ii) a fixed number of presentations of the state of the project in which the students have to defend the current state of their project in front of the other teams and eventually the project contractor.

IV. A Students Project Perspective: DSD GPXCleaner

A. The Project

The DSD GPXCleaner application was intended to provide several features to allow users to easily manage GPX data files received from GPS devices. The application is supposed to be used to handle personal excursion recordings and to set up GPX data files before sharing them with others, since most often GPS records are poorly suited to be directly used for sharing activities due to their heavy rendering; the DSD GPXCleaner application's main feature is the possibility to accurately render the GPS record in a much lighter manner by sensitively reducing the record's number of points composing the track [9].

As a result of DSD GPXCleaner being a university project, worked on by students without experience in distributed collaborative software development, no formal development process was used; rather a mixed process incorporating elements of waterfall, iterative, and evolutionary prototyping models emerged (fig.2).

Evolutionary prototyping, defined as building robust prototypes in a structured manner, which could be constantly and continuously refined, was successfully used in designing a data manipulator class, which started out having only a few basic functions, and was expanded gradually as needed (the expansion was generally triggered by the gradual completion of the graphical user interface module, or by modifications at the requirements specification by the customer) [10]. Commercial off-the-shelf software was used for the embedded map component and GPX files reading and writing functionalities; those were implemented by pre-existing code from the NASA World Wind Java code base. Our development process can be said to have had three major iterations over its course.

B. Challenges

The project faced challenges which derived mostly from the distributed nature of the development and that could be summarized as follows:

- *Cultural differences:* a distributed development, especially in the case of the DSD GPXCleaner in which students from two universities placed respectively in Croatia and Sweden and moreover coming from four different countries (Italy, Sweden, Croatia, Netherlands) composed the project team, may have to face with cultural differences. They can comprise language differences as well as a different approach to the work such as heterogeneous workload schedule by each sub-team or even each member. Working in such development other differences can be noticed, for instance in the willingness to overwork for specific reasons or to work during holidays.
- *Communication and Coordination:* they can be considered important issues in any sort of software development, but they become core issues in the particular case of a distributed development, especially when, as for the DSD GPXCleaner project, no face-to-face meetings could be run between the two sub-teams [11]. Moreover, the team was composed by students that were not used to work in a distributed manner and had to establish appropriate communication and coordination channels by themselves. Since the parties involved were three (project teams, supervisors, customers), we had to face a three-fold communication challenge. A lack in either of these issues could have severely increased the probability of failure.
- *Technical skills:* the DSD GPXCleaner project was based on two GPX file formats which were unknown to the whole project team. Moreover, the tool was developed as a Java desktop application, and not all the team members had knowledge of the Java programming language as well as the GPL graphics handling which was needed for rendering the GPS tracks on the Earth's surface. The NASA World Wind application was adapted and embedded in the DSD GPXCleaner tool for the tracks rendering on the globe, and none of the team members had knowledge of such application. Mathematical algorithms had also to be implemented in order to provide path reduction and wild points detection functionalities.
- *Customer satisfaction:* the requirements specification was changing throughout the development according

to the customer requests and these changes could have undermined a successful development if not accurately and promptly managed. Having external customers and internal supervisors made it even more complicated, since actions had to be taken in order to satisfy both at the same time.

- *Timeliness:* the DSD course was run within an eleven-weeks study period in which the involved students had to meet severe deadlines while following and working for other courses as well. In order to achieve the prefixed goals, a detailed schedule about project actions and milestones had to be defined and respected even though none of the team members had previous experience on such activities.

C. Approach and Results

In this section the solutions, which we adopted to face the challenges listed in the previous section, will be described with a special focus on the DSD GPXCleaner project perspective.

Since the project was developed as part of the DSD course with defined and mandated milestones, the early stages of development followed a waterfall model. Several project stages along with pertaining documentation were required to be completed at set dates. These were: the project description, the project requirement definition and the design description, along with three written feature descriptions, in that order. When most of the document milestones were met, we switched to an iterative model in order to meet possible modifications at the requirements specification by the customer (fig.2).

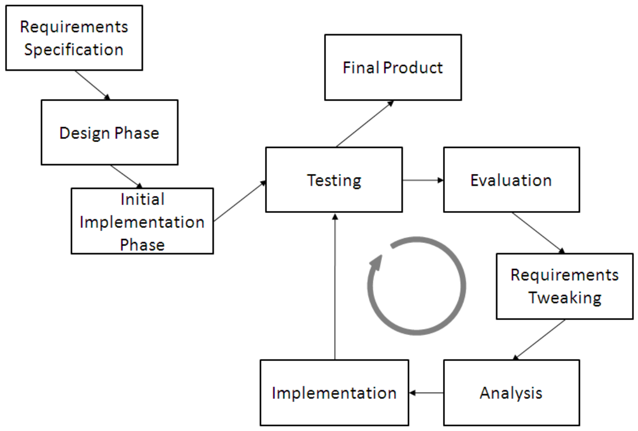

Figure 2. DSD GPXCleaner Development Process

These iterations helped in decreasing the risk of developing incomplete functionalities; generally, the mixed approach helped in satisfying both the supervisors, through the first steps inherited from the waterfall model, and the customer, through the ensuing iterative steps.

In order to solve communication and coordination issues, multiple channels have been built-up.

Communal web pages had been made available to each DSD project. These web pages were used for communication between local supervisors and the teams; in particular, for sharing files required by the university-assigned milestones (mostly documents and presentations), for sharing team summary weekly reports, and for sharing minutes of meeting of internal team meetings.

Moreover, each DSD project had the possibility to use a dedicated server running a software versioning system. This server was used for all collaborative code development.

The two universities made their teleconferencing facilities available to the staff and students of the DSD course once a week. Usually, that was used by the supervisors to communicate with the students, but it was made also available for the students to communicate amongst themselves.

Concerning the DSD GPXCleaner project, early in the development process, the team members agreed to run weekly teleconference meetings in order to discuss the current project status and to distribute the workload among the team members in a more flexible and effective manner. Detailed meeting notes were then published on the project web pages in order to make them reachable for everyone involved in the course.

A special Google Group was created in order to provide a forum-like venue of communication. The group was used to discuss development topics in depth, as an addition to the weekly Skype meetings, and for collaborative creation of documentation.

Email and instant messaging were used for communication between individual team members, and especially for communication between the project manager and both local and SCORE supervisors. This form of communication was mostly used for person-to-person communication, while the other methods were used for group communication.

In order to avoid that lack of technical skills could lead to delays or overwork, we divided the workload tasks according to each single team member's knowledge and interests when possible. Nevertheless every component of the team group had the possibility to gain in terms of improved skills in both implementing and documenting a software product.

V. Lessons Learned and Success Factors

The road to a successful completion of the project was not straightforward. For the whole project team, this was the first distributed project with members of four different nationalities with different cultures and located in two countries without the possibility to meet in person. Besides the concrete outcome, the DSD GPXCleaner application, we believe that the *non-visible* outcomes are more important; this project resulted in a very valuable experience on how to work in a team and how to cope with distance with members that are not always around to help each other. Furthermore, this gave us *real world* experience since we learned how to communicate with supervisors and a customer, work out problems and come to solutions together despite the particular distributed nature of the project.

A multicultural composition of the project team may be frightening and may undermine the successful completion of the project if taken as an obstacle. Our experience led to the conclusion that it is, on the contrary, a very positive characteristic for a project team since *cultural diversity* may mean also different approaches to the same problems that may generate unexpected and valuable solutions. Obviously, these differences have to be handled in the right manner in order to lead to positive results.

The *team workload* has to be strictly organized and scheduled, and every team member has to know exactly what to do and when to deliver it. This however includes the success of the software architecture design, not only adjusted to the run-time system, but also to the development structure.

Communication is a core aspect of a distributed development, especially for the DSD GPXCleaner case in which no face-to-face meetings have been run. Tools and schedules for distance communication had to be set and strictly followed in order to reach the prexifed milestones.

Members from different cultures may also have different ways to express their opinions and reach an agreement; this does not mean that *individualism* is worse than *collectivism* or vice versa, but rather that everyone's opinion has to be taken into serious consideration, trying not to impose one's own. If a member looks at the team's objectives as his/her own, the gap between individualism and collectivism may be reduced, and both the member and the whole team would benefit from that. Even if distributely located, most of the team members became not only colleagues but rather friends during the development, and that increased the willingness to help each other.

The successful completion of several projects within the DSD course highlights the fact that the *supervising activities* had a key role in such a success.

Going back to the *success factors*, we perceived, from a DSD GPXCleaner developer's perspective, that the most important were the following: (i) DSD course design, (ii) communication & coordination management, (iii) positive attitude towards the cultural differences and (iv) team motivation coming from the double goal given by the course completion and the participation at the SCORE competition. This perception is partially supported by anonymous students' surveys evaluating the DSD course [12], which highlight the course design and communication & coordination management as particularly positive factors. Besides these factors, at the end of the development some issues, which could have been treated in a different and perhaps better way, can be identified. The versioning system offered by the common repository has not been completely trusted, due to some initial technical problems of the repository itself. Therefore we kept on using both the versioning system and a direct manual exchange of files that could have led to version conflicts resulting in loss of precious time. Furthermore, the testing phases have not been very well planned and carried out since we did not set up any automation in the procedure; this may be considered the weakest spot of the whole development process.

VI. Conclusion

In this paper we have presented challenges of working in a distributed project in a distributed software development course from a student's perspective.
A provident design and structuring of the course itself has been a core success factor. From a project perspective, the management of communication and coordination issues, the team motivation coming from the given double goal (the course itself and the SCORE competition) and a very positive attitude towards the challenging cultural differences are undoubtly to be highlighted as essential success factors.

References

[1] I. Crnković, R. Land, I. Bosnić, I. Čavrak, and M. Žagar, "Customers' Role in Teaching Distributed Software Development," in *Proocedings of CSEET '10*. IEEE Computer Society, 2010, to appear.

[2] The SCORE 2009 Contest. (2009) Official Website. [Online]. Available: http://score.elet.polimi.it/

[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 *Proceedings of ICSE '05*. ACM, 2005, pp. 643–644.

[4] M. Jimnez, M. Piattini, and A. Vizcano, "Challenges and Improvements in Distributed Software Development: A Systematic Review," *Advances in Software Engineering*, 2009.

[5] J. Favela and F. Peña Mora, "An Experience in Collaborative Software Engineering Education," *IEEE Software*, pp. 47–53, 2001.

[6] A. E. Milewski, N. Mullick, P. Keil, and I. Richardson, "Distributed development: an education perspective on the global studio project," *Software Engineering, International Conference on*, pp. 679–684, 2006.

[7] I. Crnković, R. Land, I. Bosnić, I. Čavrak, and M. Žagar. (2004) DSD Course Design. [Online]. Available: http://www.fer.hr/rasip/projekti/rpi/en/design

[8] C. Peterson, "Bringing ADDIE to Life: Instructional Design at Its Best," *Journal of Educational Multimedia and Hypermedia*, pp. 227–241, 2003.

[9] M. Young. (2008) SCORE Project Specification. GPXCleaner: GPS Path Editing and Simplification. [Online]. Available: http://score.elet.polimi.it/projects/young.pdf

[10] F. Ciccozzi, T. Tvrtković, T. Bregović, J. Labor, C. Tempelaars, P. Santibañez Jara, and J. Jutterström. (2009) DSD GPXCleaner SCORE Report. [Online]. Available: http://www.fer.hr/_download/repository/DSD_GPXCleaner_SCORE_report.pdf

[11] D. Woit and K. Bell, "Student communication challenges in distributed software engineering environments," in *Proceedings of ITiCSE '05*. ACM, 2005, pp. 286–290.

[12] DSD Course Team. (2004) DSD Course Evaluation Students' Surveys. [Online]. Available: http://www.fer.hr/rasip/projekti/rpi/en/surveys