Five Years of Lessons Learned from the Software Engineering Course: Adapting Best Practices for Distributed Software Development

Crescencio Rodrigues Lima Neto[2], Eduardo Santana de Almeida[1,2]

[1]Computer Science Department - Federal University of Bahia (DCC/UFBA)
[2]Reuse in Software Engineering Labs (RiSE)
crescencio@gmail.com, esa@dcc.ufba.br

***Abstract*—Several companies around the world are using Distributed Software Development (DSD) to reduce costs and some Software Engineering courses are trying to simulate this distributed environment. This paper shows the experience faced by students during five years from the Software Engineering course performed at the Federal University of Pernambuco, Brazil, which the objective was adapting the best practices from traditional development for DSD. Course lectures and materials educate students about software engineering best practices and DSD. The students developed a project organized into a set of work assignments that could be distributed across groups. At the end they learned to communicate and collaborate with each other, and they also believed that the course was helpful to them, which justifies the low number of dropouts. Most of the students, but not all, successfully completed their projects.**

***Keywords*-Distributed Software Development; Software Engineering Education**

I. Introduction

As a result of the globalization, companies around the world have started to distribute their development environment across different sites in order to produce better software products and faster their services [1]. Students, however, rarely get a chance to learn the problems that arise, and the potential solutions to those problems, when conducting Distributed Software Development (DSD) [2].

According to [3], software engineering education emphasizes importance of practice, but courses in DSD are still rather rare. The main reason is the difficulty in establishing distributed, and yet common environment, supported by more than one site [4].

Several studies [2], [4], [5], [6], highlight the benefits of running a distributed project. Although there are many good reasons to globally distribute development activities, success is not guaranteed by just opening a development center in another region of the world [7].

This paper discusses the experience of the Software Engineering course conducted at the Center for Informatics from Federal University of Pernambuco (CIn/UFPE), Brazil. The course adapts techniques, best practices, processes, agile methodologies, and so on, for DSD. We analyze the last five years of the course figuring out what is necessary to allow the adaptation for DSD and reviewing the lessons learned.

The remainder of this paper is structured as follows: Section II presents the related work. Section III briefly describes the history of the Software Engineering course using DSD. Section IV describes the lessons learned from creating a Software Engineering course combining Distributed Software Development with other techniques, and, finally, the conclusion and future work are discussed in Section V.

II. Related Work

[8] summarizes experiences and best practices from Alcatel[1]. The projects were executed in several locations on different continents and in many cultures.

[9] reports the experiences acquired on the adaptation of a distributed development process based on SCRUM performed by an open source software factory. [10] presents an experience of adapting the Rational Unified Process for a small distributed software development project and reports on problems identified during the development and how they have been solved.

[11] presents an open source tool called FireScrum, which is a project planning and management tool designed to support distributed Scrum teams. [12] describes the experiences in adopting agile software development with DSD jointly with the aim of raising the problems and report possible solutions that have been taken and can be applied in other distributed teams.

[1] presents a Systematic Literature Review to identify which ways of collaboration are commonly used by software organizations where teams are temporal and geographically dispersed, with also different native languages and culture. [13] presents an experience of adapting Scrum agile practices in a distributed software project that involves a large team. It also presents the results, the difficulties and lessons learned.

Finally, [14] presents experiences and lessons learned from teaching a distributed course on Collaborative Innovation Networks over six years.

The majority of these related work ([1], [9], [10], [11], [12], and [13]) were derived from the CIn/UFPE Software Engineering course, and the authors were group members

[1]http://www.alcatel-lucent.com/

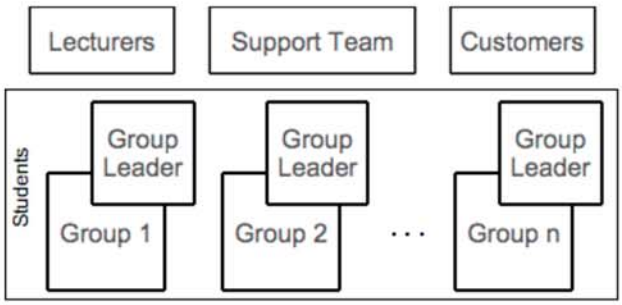

Figure 1. DSD Project Structure

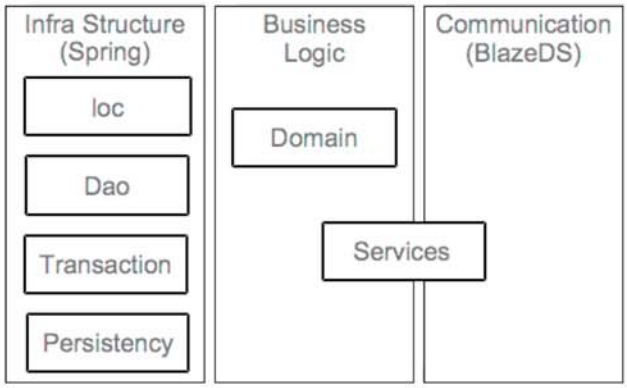

Figure 2. Back End Architecture from [11]

from the projects. In this paper, we present the information about the course in order to analyze the lessons learned from each year of experience.

III. The History of the Course

The last five years of the Software Engineering Course focused on adapting best practices from the traditional and agile development for Distributed Software Development, starting in 2007. The course was structured by the combination of lectures, guest presentations and the development of a distributed project. The lectures were presented with the presence of all the students once in a week. Since the students live in different cities, they used the rest of the day after the class for meetings and discussions.

In average, sixty students from the master course participated in the course every year, ranging from 50 to 70. They are organized into five or six groups of ten members, and each team was responsible for part of the project. At the end of the semester, the teams must gather all the parts and functionalities in only one environment creating a system that must work properly. Each team has a project manager responsible for coordinating the activities and easing the communication among the groups. The project includes all phases of software development (requirements specification, design, development and tests).

Figure 1 presents the DSD project structure. Lecturers are responsible for coordinating the DSD project and teaching the lectures. The support team is responsible for planning, following, integrating, verifying the quality, and managing the final versions. Customers present the functionalities to be produced, define the priorities of the implementation and evaluate the software produced. The group leaders manage the activities from their teams and facilitate the communication among groups. Finally, the groups implement and test the functionalities from the project.

The projects were performed as follows: the customers outlined the requirements. The group leader analyzed the problem, identified the requirements and negotiated with the customer. The groups reported the project state to the project steering group periodically for all students enrolled in the course. The group members communicated using Internet-based tools such as Skype[2] and Google Talk[3]. Team members collaborated using discussion groups and other collaborative tools for sharing information and assets. According to [4], the main underlying goal in each project was to remove the distance factor.

The course has shown to be interesting for acquiring industrial experience. It tries to simulate industrial environments by using real projects. Typically 4 to 6 guest lecturers from the industry gave presentations each year. In the five years of the course analyzed, the students were mostly from the Brazilian nationality and we could discuss some cultural issues from the Brazilian perspective.

Figure 2 presents the back end architecture from the project of the year 2009 [11]. The architecture of the distributed project was structured as follows:

- **Infra Structure** - The infra structure was managed by the Spring[4] framework. It was responsible for the dependency injection by reducing the coupling among classes in the system.
- **Business Logic** - It contains the system domain entities, application functionalities, and services.
- **Communication** - The communication was guaranteed by the BlazeDS[5]. It was responsible for the communication between Back End and Front End.

During the development some groups developed functionalities focused at the Back End, some focused at the Front End, and the others focused at combining both parts. At the end of the project, after assembling the parts, the application worked through the communication among the solutions.

Figure 3 presents the amount of actions (read transactions, write transactions, and write files) during the project development, from the start of the course (3/1/2009) until the end (7/1/2009). As it can be seen, there is a peak of

[2]http://www.skype.com/
[3]http://www.google.com/talk/
[4]http://www.springsource.org/
[5]http://opensource.adobe.com/wiki/display/blazeds/BlazeDS

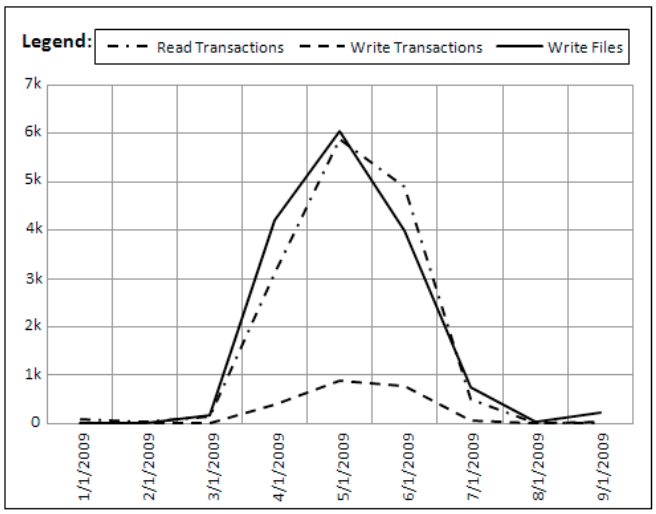

Figure 3. Amount of Actions per Month

actions in 05/1/2009. A possible reason for this fact is the knowledge acquired during the course by the users and the familiarization with the adaptations of the traditional development for DSD.

The adaptations of the best practices enabled the productivity growth in a short learning period. We identified that the combination of some motivating factors raised by [15], such as: learning new things, challenging project, and team atmosphere contributed for the success of the projects.

At the end of the course, the application final version was released for the software engineering community. Figure 4 shows the amount of download of the project distributed by the years. Until 2012, more than 19.250 downloads were made by 115 different countries. More details about it can be seen in: *http://sourceforge.net/projects/firescrum/*.

IV. Lessons Learned of Adapting DSD

In order to implement efficiently the DSD, some conventional practices of software development needed to be adapted. After five years of experience, a set of good practices from the traditional way of development was adjusted for Distributed Software Development. The main findings identified are described next:

A. Online Daily Meetings

Since the participants were not at the same city, online daily meeting of fifteen minutes were managed in order to discuss the problems and define the activities from each day. Web-based communication tools were also used to handle and record the meetings, and the group leader was responsible for coordinating this activity.

At the beginning of the project, the teams exceeded the meetings time limit (fifteen minutes). The solution to mitigate this problem of non-objectivity was the use of prior communication by mail among team members and the group leader. Possible solutions were raised before the meetings and the decision-making were held during the meetings.

B. Group Leader for Managing the Difficulties and Facilitating the Communication

With the team geographically distributed, the group leaders accumulated more responsibilities. They managed the group impediments, implemented functionalities as developers, tested the application as testers, and facilitated the communication among members and groups.

According to [7], team members must communicate whenever necessary to make the team efficient. For this reason, the group leader was the first person of the team to know about the communication problems in order to solve them.

C. Iterative and Incremental Development

One narrow scope was defined at the beginning of the project because the groups needed to understand the context and the technologies. The customers selected new functionalities during the development, which were implemented and presented by the groups. At the end of the iteration, the customers evaluated part of the working system. After the evaluation, they required new functionalities or demanded changes at the existing ones. Liasons were also defined to handle the communication with the customers.

Another changes we introduced to achieve real incremental development were to analyze requirements from the beginning in view of how they could be clustered to related functionality, analyze the context impacts of all increments up front before starting development, and develop each increment within one dedicated team, although a team might be assigned to several increments [7].

D. Activity Redistribution During Development

During the development some tasks have been redistributed among the members of the group to ensure the project evolution. Problems with services provided by the customers needed to be resolved in advance and some others were postponed according to the prioritization order.

All the team-building efforts had to take place in a shared environment [8], so the participants from different sites had the opportunity to exchange information. Therefore, it was possible to know the difficulties faced by the teams and allow the cooperation in order to solve the problems.

E. Version Control Closely Monitored

In order to provide the necessary information for the developers to configure the development environment of the Project, the configuration plan was elaborated. As the project progressed and the configuration problems appeared, measures were taken to eliminate and minimize these problems. Some were only recommendations on what to do if problems

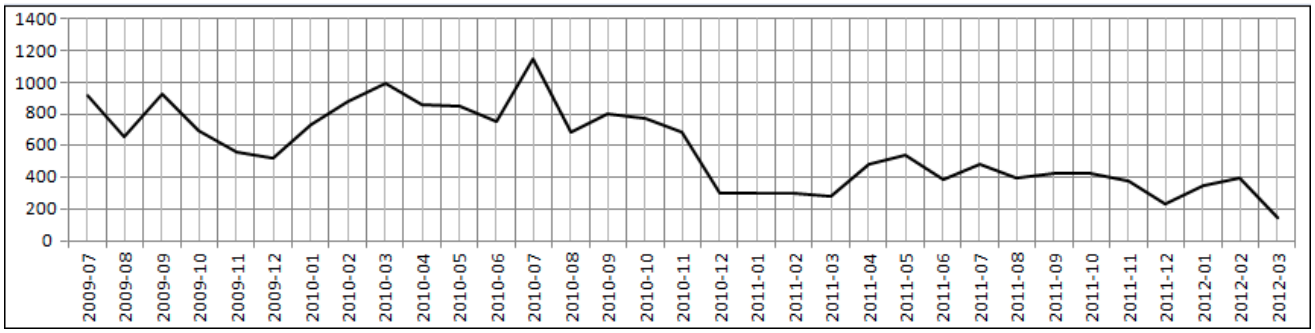

Figure 4. Amount of Downloads per Year

occur in the setup steps. One member was responsible for managing the version control of the project.

The synchronization of deliveries in a distributed project added complexity to the development process [7]. The developers did not easily copy corrections from one code branch to the other because they impacted ongoing development. Therefore, we used the alert mechanism of a bug tracking tool (Mantis[6]) in order to identify every project changes.

According to [7], it is necessary to rigorously enforce the configuration management and build management rules (such as branching, merging, and synchronization frequency) and provide the necessary tool support.

F. Self-manageable Groups

The groups worked as self-managing teams, considering the motivation as the main aspect for the project success [16]. These same features were noticed in distributed development, which teams were formed by any member of the open source community who wanted to be part of the project. Each member chosen which activity they wanted to develop.

In order to optimize the productivity, the groups were balanced [14] according to the profile of each individual. Thus, the teams were composed by students with and without experience in software development. It is worthwhile to mention that one of the course objectives was to allow the exchange of knowledge among students. They shared thier experiences by solving a real problem using a DSD environment.

G. Online Pair Programming

Initially, the groups generally believed that this method would not be efficient compared with pair programming with physical presence, but, during the Implementation phase, this mode of collaboration has proved very effective and was supported by tools such as emails and instant messages with or/and without audio.

Developers and testers with experience were responsible for teaching and sharing their knowledge with developers and testers without experience. As they were learning in practice, they assumed more complex tasks.

[6]http://www.mantisbt.org/

H. Internal and External Groups Training

Training were given among the group members in order to balance the knowledge. The members who had more experience in some activity were overwhelmed at the beginning of the project because in addition to the tasks they also coordinated the training. As a solution to balance the roles, after the training the other members assumed activities while gradually acquired experience in practice. When a good internal group practice benefited the whole project, the training was taken for all groups.

Finally, we used focus groups and interviews to get feedback from the instructors and colleagues. While focus groups encouraged reflection on the teamwork and address how well team members collaborated, interviews brought the experience from the individual to the fore [8].

V. Conclusion and Future Work

Find new ways of adapting software development methodologies to the reality of teams and projects is an interesting way to bring new processes for different approaches. This work presented an experience of adapting best practices in different contexts of the project from traditional development for distributed software development.

Thus, we analyzed the lessons learned identified during five years of the software engineering course. It is essential that practical experience, as presented here, are available for other students and professors, seeking results more relevant to the software community. Finally, the personal commitment of the students (group members) was important for the success of the distributed development projects. Another key factor was the constant communication among groups and group members in order to avoid rework and time loss.

As future work, we identified the necessity to define a Software Product Lines for the course. Each year, a new product of the product line could be created or each group could create a product of the line. Moreover, with

the partnership of local companies, we intend to apply the lessons learned in industrial projects.

ACKNOWLEDGMENT

This work was partially supported by the National Institute of Science and Technology for Software Engineering (INES), funded by CNPq and FACEPE, grants 573964/2008-4 and APQ-1037-1.03/08 and CNPq grants 305968/2010-6, 559997/2010-8, 474766/2010-1.

REFERENCES

[1] R. G. C. Rocha, C. Costa, C. Rodrigues, R. R. Azevedo, I. Junior, S. Meira, and R. Prikladnicki, "Collaboration Models in Distributed Software Development: a Systematic Review," in *CLEI Electronic Journal*, vol. 14, no. 2, 2011.

[2] D. Weiss, E. S. Almeida, L. Dali, S. Faulk, C. R. L. Neto, Z. Rui, J. Ying, M. Young, and L. Yu, "Teaching Globally Distributed Software Development An Experience Report," in *25th Conference on Software Engineering Education and Training*. IEEE Computer Society, 2012.

[3] M. Shaw, "Software Engineering Education: a Roadmap," in *Proceedings of the Conference on The Future of Software Engineering*. ACM Press, 2000.

[4] I. Bosnić, I. Čavrak, M. Orlić, M. Žagar, and I. Crnković, "Avoiding Scylla and Charybdis in Distributed Software Development Course," in *Proceedings of the 2011 Community Building Workshop on Collaborative Teaching of Globally Distributed Software Development*. New York, NY, USA: ACM, 2011, pp. 26–30.

[5] H. Holmstrom, E. O. Conchuir, P. J. Agerfalk, and B. Fitzgerald, "Global Software Development Challenges: A Case Study on Temporal, Geographical and Socio-Cultural Distance," in *International Conference on Global Software Engineering, 2006. ICGSE '06.*, oct. 2006, pp. 3 –11.

[6] M. Nordio, C. Ghezzi, B. Meyer, E. Di Nitto, G. Tamburrelli, J. Tschannen, N. Aguirre, and V. Kulkarni, "Teaching Software Engineering using Globally Distributed Projects: The DOSE Course," in *Proceedings of the 2011 Community Building Workshop on Collaborative Teaching of Globally Distributed Software Development*, ser. CTGDSD '11. New York, NY, USA: ACM, 2011, pp. 36–40.

[7] C. Ebert and P. De Neve, "Surviving Global Software Development," *Software, IEEE*, vol. 18, no. 2, pp. 62 –69, mar/apr 2001.

[8] J. Favela and F. Pena-Mora, "An Experience in Collaborative Software Engineering Education," *Software, IEEE*, vol. 18, no. 2, pp. 47 –53, mar/apr 2001.

[9] F. Soares, L. Mariz, Y. C. Cavalcanti, J. P. Rodrigues, M. G. Neto, P. Bastos, A. C. Almeida, D. Pereira, R. Seabra, and J. Albuqueruqe, "SCRUM adoption in a Factory of Distributed Software Development," in *I Distributed Software Development Workshop*, 2007.

[10] R. Rocha, D. Arcoverde, R. Brito, B. Aroxa, C. Costa, F. Q. B. Silva, J. Albuquerque, and S. R. L. Meira, "An Experience on Adapting RUP for Small Teams of Distributed Development," in *II Distributed Software Development Workshop*, 2008.

[11] E. Cavalcanti, T. M. Maciel, and J. Albuquerque, "Open Source Tool for Support Distributed Teams Using SCRUM," in *III Distributed Software Development Workshop*, 2009.

[12] V. C. Chalegre, W. B. Santos, L. D. Souza, H. J. Muñoz, and S. R. L. Meira, "Case Study on the Use of SCRUM with Distributed Software Development," in *Brazilian Workshop of Agile Methods*, 2010.

[13] A. B. Junior, F. Kenji, P. Alves, R. Rocha, R. R. de Azevedo, and S. Meira, "Using SCRUM Practices at the Software Development With Distributed Teams," in *V Distributed Software Development Workshop*, 2011.

[14] P. Gloor, M. Paasivaara, C. Lassenius, D. Schoder, K. Fischbach, and C. Miller, "Teaching a Global Project Course: Experiences and Lessons Learned," in *Proceedings of the 2011 Community Building Workshop on Collaborative Teaching of Globally Distributed Software Development*, ser. CTGDSD '11. New York, NY, USA: ACM, 2011, pp. 1–5.

[15] I. Bosnić, I. Čavrak, M. Orlić, M. Žagar, and I. Crnković, "Student Motivation in Distributed Software Development Projects," in *Proceedings of the 2011 Community Building Workshop on Collaborative Teaching of Globally Distributed Software Development*, ser. CTGDSD '11. New York, NY, USA: ACM, 2011, pp. 31–35.

[16] K. Schwaber and M. Beedle, *Agile Software Development with Scrum*. Prentice Hall, 2001.