Challenges and Lessons Learned on Preparing Graduate Students for GSE Work: Brazilians' Perceptions on a Multi-Site Course Experience

Josiane Kroll, Caroline Q. Santos, Letícia S. Machado, Sabrina Marczak, Rafael Prikladnicki
Computer Science School, PUCRS
Porto Alegre, Brazil
{josiane.kroll, caroline.queiroz, leticia.machado.001}@acad.pucrs.br,
{sabrina.marczak, rafaelp}@pucrs.br

***Abstract*—Global Software Engineering (GSE) has became a part of the academic curricula in Computer Science courses. However, training students for GSE inherits the challenges of teaching Software Engineering (SE) in globally distributed environments. Furthermore, the most related experience in teaching graduate students reveals difficulties in developing GSE competencies. In this paper, we report the Brazilians' perceptions in performing SE activities in a globally distributed environment. We collected data from a collaborative project developed as part of the DOSE (Distributed and Outsourced Software Engineering) project. As a result, we identified 12 challenges and 7 lessons learned on preparing graduate students for GSE. Our results are helpful for practitioners and researchers in supporting new strategies for training students in the future.**

***Index Terms*—Global Software Engineering; Software Engineering Education; Teaching; DOSE Project; Challenge.**

I. Introduction

Global Software Engineering (GSE) has become a common practice in the software industry. GSE affords new opportunities for cross-site modularization of development work, potential access to a larger and better-skilled pool of developers, and the possibility of greater innovation, learning and transferring of best practices [3]. In this context, the importance of teaching skills for geographically distributed software development becomes essential [4].

The "Distributed and Outsourced Software Engineering" (DOSE) project [1] creates an opportunity to students develop GSE competences working in software engineering (SE) projects. Students from universities involved in the project take part in a case study exploring techniques for making an offshored project succeed or recover from problems [9].

In this paper, we report the experience of teaching GSE for graduate students at PUCRS University, the largest private non profit university in the South of Brazil. We collected data about the students' perception on performing SE activities in GSE environments based on the 2014 course session of the DOSE project. Our study highlight difficulties and lessons learned by the students that are not different from what practitioners face in daily projects, reinforcing the need to train students in GSE. We present details of our study and findings next.

II. Related work

In GSE literature, there are many studies discussing SE education. Similar to the experience presented in this paper, other studies have reported challenges and lessons learned from diverse GSE projects (e.g., [4] [9] [10] [13]). Given the limited number of pages, here we discuss just some of them.

Damian et al. [4] report experiences in the design, teaching, and evaluation of a course intended to prepare graduates for GSE. The course was delivered as a collaboration among three geographically distributed universities. The course emphasized the learning of requirements management activities in a GSE environment with time zone and language differences.

Nordio et al. [10] and Nordio et al. [11] describe a practical experience in teaching GSE in the DOSE course. They discuss how their experience enabled to recreate the atmosphere of an GSE project, the challenges faced by the participants, and how they dealt with these challenges. Similarly, Zei and El-Bahe [13] also discuss challenges associated with the management and execution of GSE projects in academic settings. More specifically, they report on cultural factors that affected initiating a GSE course at the American University of Kuwait (AUK). The goal of their project was to create an innovative and highly challenging learning environment for educating the next generations of software engineers.

Some other aspects of the DOSE course have also been described in previous publications. For instance, Meyer et al. [7] present in detail the first experience in GSE teaching from the DOSE 2007 course. While Nordio et al. [12] take an example from the 2008 session to illustrate the specification risks of distributed development.

Our paper differs from the previous ones from the DOSE course for taking into consideration the current setting of the Brazilian site only: the course was run as a 'practical course', meaning that the students had to develop a project in practice without learning theory along with it. When designing it, we assumed that this format would offer the students an even closer scenario from daily industry life where people often start working with no preliminary training on GSE. We report our findings here.

2329-6313/16 $25.00 © 2016 IEEE
DOI 10.1109/ICGSEW.2016.17

III. Research Setting

In this section we describe the DOSE project and research method followed in this study. We give details of the project settings, GSE activities developed in the classroom by the students, and how we collected and analyzed the data.

A. The DOSE project

PUCRS delivers a Master of Science and a Doctor of Philosophy in Computer Science degrees. As part of these programs, graduate students take classes in SE related topics in which GSE is a peripheral topic.

From 2012 to 2014, PUCRS has participated of the DOSE project where students develop SE activities in a distributed software development setting. This project is organized in the context of a distributed software engineering course carried out in collaboration with several universities around the world. The 2014 session had the following universities as participants: ETH Zurich, Switzerland; Innopolis University, Russia; Politecnico di Milano, Italy; PUCRS University, Brazil; University of Rio Cuarto, Argentina; and Universidad Politecnica de Madrid, Spain.

The DOSE project aims to favor the collaboration between students prior to project development. It consists of making teams in different countries work collaboratively to solve a set of simple tasks [9].

ETH Zurich is the main head and coordinates the project since 2007. In each partner university, an instructor coordinated the activities of her respective students according to a prior work plan established among the universities. The project topic for the 2014 session, the case reported in this paper, was a web-based project management system to support agile distributed software development.

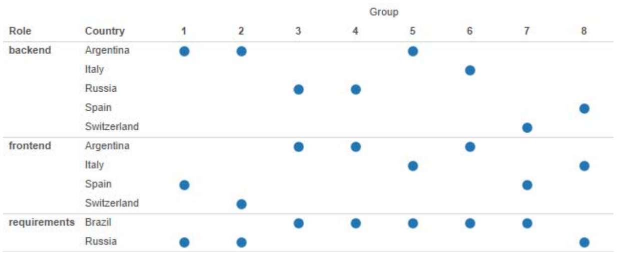
B. GSE Activities in the Classroom

In the 2014 DOSE course session, each university owned several small groups of students. Each group had one requirements team, two development teams (front-end and back-end), and one coordinator (instructor).

The project was organized into 8 groups implementing each a web-based project management system to support agile distributed software development (see Figure 1). Overall, there were 24 different sub-teams, as follows: Argentina (6 teams), Brazil (5 teams), Italy (3 teams), Russia (5 teams), Spain (3 teams), and Switzerland (2 teams). Figure 1 shows the organization of the 24 sub-teams into the 8 groups involved in the DOSE 2014 edition. Each group worked independently and distributed in three locations, and the communication among its members was self-regulated; i.e., each group decided which communication media to use, how to communicate, and so on.

The DOSE coordinator suggested an initial list of expected basic and advanced functionalities the system-to-be-delivered should have. Basic functionalities were focused on the project planning and specification as follows:

- Backlog: a collection of requirements that will be implemented by the project.



		Group							
Role	Country	1	2	3	4	5	6	7	8
backend	Argentina	●	●			●			
	Italy						●		
	Russia			●	●				
	Spain								●
	Switzerland							●	
frontend	Argentina			●	●		●		
	Italy					●			●
	Spain	●						●	
	Switzerland		●						
requirements	Brazil			●	●	●	●	●	
	Russia	●	●						●

Fig. 1. Teams' distribution at the 2014 course session of the DOSE project.

- Sprint log: a log that is used as part of a development sprint.
- User management: users need to be able to register with the system; users need to be able to assign users to projects; basic user management (delete account, change personal details). Support for different roles (for example developers and 'master'). Functionalities should be available depending on user roles.
- User authentication.
- Users can be assigned to requirements; maybe requirements need to be broken down into smaller tasks.
- Requirements should be assigned with 'points' according to their complexity. Users should also assign point to tasks. Users who implement the requirements/tasks get the 'points' of the implemented requirements.
- Developers chart: all developers are ranked according to the 'points' received from the implemented tasks/requirements.

Advanced functionalities were combined with basic functionalities in order to succeed in the project, as follows:

- A dashboard that summarizes ongoing events of a project.
- Statistics that show project performances, compared to the estimates.
- Team communication tools (e.g. a discussion board or group chat).
- Integration with issue trackers (e.g. the tracker from where the source code is hosted - Github, Bitbucket, etc) or other platforms.

The project was developed through successive iterations. Given the local school calendars, some universities joined or concluded the project in different schedules from the official project calendar–mid Sept to mid Dec 2014. PUCRS, for instance, started in mid Aug and ended its' teams activities in late Nov.

Google Code was used for storing source code and documents (using a SVN repository) and for providing basic information (using a Wiki page). English was the elected language for communication among the distributed teams and for the hand out of deliverables.

As shown in Figure 1, requirements teams were from Brazil and Russia. Both teams started their activities before the others students. These teams were instructed to prepare a requirements document (SRS) based on the functionalities

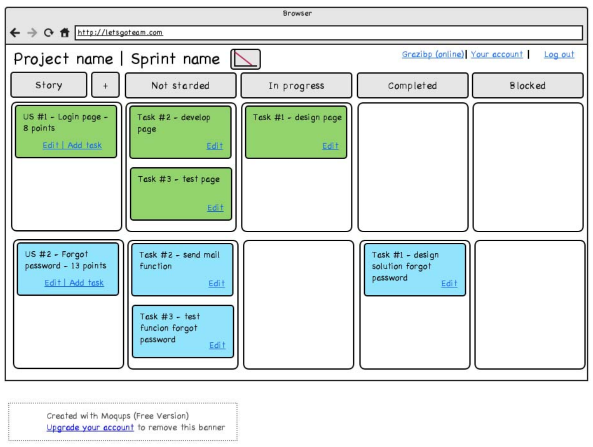

Browser
http://letsgoteam.com
Project name | Sprint name
Grazibp (online) | Your account | Log out
Story
+
Not starded
In progress
Completed
Blocked
US #1 - Login page - 8 points
Edit | Add task
Task #2 - develop page
Edit
Task #3 - test page
Edit
Task #1 - design page
Edit
US #2 - Forgot password - 13 points
Edit | Add task
Task #2 - send mail function
Edit
Task #3 - test funcion forgot password
Edit
Task #1 - design solution forgot password
Edit
Created with Moqups (Free Version)
Upgrade your account to remove this banner

Fig. 2. A sample of Taskboard screen prototype by Group 7.

listed by the customer (the DOSE project coordinator). In addition to writing the requirements specification, the Brazilian teams also prototyped the system screens (UI) in order to better assist the development teams, as illustrated in Figure 2.

Each member of a group (excluding the requirements teams) had to review the SRS document. During the review, the member had to assess the quality of the document and fill-in an online questionnaire about it. Reviews occurred until the consolidation phase. The goal of the consolidation phase was to agree on the requirements to be implemented. Groups should discuss whether the requirements were feasible to be tested, whether there were missing requirements, and also identify needed clarifications or change requests. Any changes to the SRS document had to be discussed and agreed with the requirements team.

C. Data Collection and Analysis

At PUCRS, the students were evaluated based on the quality of the SRS document and of their participation during the checking-points meetings scheduled every other week during the course period. Although it was not part of the course grade, the students knew from day 1 they had to write a group project report describing their perceptions about working in a distributed fashion. We asked them to report on the difficulties faced during the project development and the lessons they learned throughout the project. A class discussion was conducted in the last week once grades have been published so the students could debate about their reports and speak freely. Students gave consent to the course instructor to have their notes consolidated and published. We performed a qualitative analysis on the information provide by the students. Next we present our results.

IV. Results

Our analysis of the data revealed 12 challenges and 7 lessons learned on preparing graduate students for GSE work. We briefly describe each of them in this section.

A. Challenges

Challenges experienced by students were categorized in three categories: Communication, Coordination, and Control. Each challenge is also relating to a particular source characteristic: temporal, geographical, or socio-cultural distance [2]. These challenges are summarized in Table I.

- Communication Challenges: Temporal, geographical, and socio-cultural distance can impact communication processes by creating challenges. In our study, students reported four challenges related to communication. They felt difficulties in understanding the scope of the project, scheduling meetings with team members from other sites, getting the customer involved, and getting to understand other team members leading to cultural misunderstandings due to the inconsistent communication. Following we describe these challenges.

- **To understand the scope of the project:** some participants did not understand well the project scope. They felt that important aspects could be made more specific. A few participants had a clear vision of what the customer wanted and they were failing to understand the project as a whole. Many client-directed questions raised by the requirements team were answered simply with a ”go ahead and define this yourselves”. Participants believe that the lack of explanations about the clarification requests might have contributed to controversial opinions about the requirements.
- **To schedule meetings:** some teams did not attended group calls due to the lack of agenda. It was difficult to find a common time for those involved in the project. This difficulty was mainly because of time zone differences and availability of the team members due to class schedules and program dedication (some were part-time students). Students mainly adopted e-mail as a communication mechanism to discuss the requirements.
- **To get the customer involved:** some students faced difficulties interacting with the customer. Only one asynchronous clarification session with the client was made during the project time. The students concluded that one single session was not enough to debate the initial list of requirements. Many pending clarifications led to decisions to be made by the requirements teams, which turned out to reflect on the downstream decisions from the development teams.
- **Inconsistent communication (miscommunication):** some team members felt difficulties in taking decisions. When a decision was taken, the information was not be spread out the other group members.

- Coordination Challenges: Temporal, geographical, and socio-cultural distance can also impact GSE coordination. The need for coordination increases when there is a substantial dependency among various activities and team members [5]. In this category, students faced difficulties in defining tools for communication, clarifying responsibilities of the requirement teams and overall project roles, understanding the software development process in order to define consistent work practices,

TABLE I
SUMMARY OF THE IDENTIFIED CHALLENGES

Category	Challenge	Source Category
Communication	To understand the scope of the project	Socio-cultural distance
	To schedule meetings	Temporal distance
	To get the customer involved	Geographical distance
	Inconsistent communication (miscommunication)	Socio-cultural distance
Coordination	To define tools for communication	Socio-cultural distance
	To clarify responsibilities of requirement teams and project roles	Socio-cultural distance
	To define consistent work practices	Socio-cultural distance
	To establish collaboration among team members/sites	Socio-cultural distance
Control	To be proactive	Socio-cultural distance
	To adopt new tools and programming languages	Socio-cultural distance
	To build teams with different work dedication (full-time vs. part-time)	Temporal distance
	To motivate the team	Socio-cultural distance

and establishing collaboration among members from the same or different sites.

- **To define tools for communication**: students reported the lack of clear rules to be followed by all teams about which tool to use and how to use them. They reported that they felt like there were 'weak ties' to the team spirit, which would have helped improve the overall perception about how to solve issues, including this one. They also reported difficulties in getting access to the development team project repository (SVN).
- **To clarify responsibilities of requirement teams and project roles**: project roles and hierarchy were not well defined, for instance, the development teams had to rely solely on the requirements team.
- **To define consistent work practices**: students were spread in different countries, where differences in national culture, language, motivation, and work ethics may impeded effective project coordination. They faced difficulties in planning kick-off meetings, understand interdependencies between systems, teams, and projects.
- **To establish collaboration among team members/ sites**: students reported difficulties in collaborating over geographic, temporal, cultural, and linguistic distances. English is not a language fluently spoken by Brazilians although kids are taught it at school. Thus, the lack of competencies concerned with cultural and language barriers made collaboration even more difficulty.

- Control Challenges: Control challenges are recognized to arise due to geographic, temporal, and socio-cultural distances encountered in GSE environments [6]. Students faced control difficulties related to get proactive, to adopt new tools and programming languages, to build teams with different work hours dedication, and to motivate team members to work together. Following we describe theses challenges.

- **To be proactive:** during the project, students faced delays in receiving answers to clarification requests about the defined requirements. Students from Brazil considered themselves as not proactive in answer questions and solve conflicts. Comments such as *"we were not good at providing feedback"* were reported by the students.
- **To adopt new tools and programming languages:** the development teams adopted a new programming language and tools. These tools and language were not experienced by all students. Thus, several problems occurred due to the lack of skills in the technologies used in the project.
- **To build teams with different work hours dedication (full-time vs. part-time):** some students were available full-time and others part-time. These differences in work hours resulted in several problems such as communication misunderstandings, missed deadlines, lack of teamness, and others.
- **To motivate the team:** not all Brazilian students were feeling motivated and willing to make the project a success. Thus, some activities were not completed during the project for several reasons such as the course load of the term in relation to their time dedication to the Master/PhD program.

B. Lessons Learned

Taking into account that the concept of GSE is based on the idea of performing software development with distributed teams, project success and the satisfaction of stakeholders must be ensured through the use of a process that facilitates interaction among those involved, the use of best practices in project management, and a good survey and specification of system requirements.

Since the project involved students from different universities in different countries, and in addition, the class schedules were different, communication was identified as a key factor for the success of the DOSE project. We realized that multicultural aspects are relevant to specify requirements systems (and the other development cycle phases). Thus, we present next the lessons learned by the PUCRS students in the 2014 DOSE project, along with a brief description of each one.

Lesson learned 1: Obtain prior knowledge on the problem-area domain. Prior knowledge on the problem-area domain would be essential for the system requirements specification. Participants realized that after the release of the SRS document not all members of their group knew the subject they were working on (i.e., agile development). They believe that if a leveling had been done before the beginning of the course their performance would have been better.

Lesson learned 2: Define the standardization of terms and expressions. Some teams realized belatedly that their members used different terms to refer to the same subject. For example, Sprint and Iteration were used to refer to ”release cycles”. Perhaps this happened because it was done by many people at the same time and there was not a careful review before being released. An agreement was later negotiated, which significantly contributed to the understanding of the requirements.

Lesson learned 3: Adoption of best practices for communication. Communication among group members was done by e-mail, Google Code, Google Docs, and Skype. For some groups, meetings were unproductive because the lack of a good meeting structure, e.g., purpose, agenda, required decisions that need to be made, requests for make decision during the weekend etc. For others teams, conversations via e-mail flowed well. However, meetings by Skype were positive and allowed the clarification of points that were not clear in the SRS. A team highlighted that *”it would be interesting if a mandatory weekly meeting was to be set”*. Teams liked to interact with the distributed members using Google docs. After the meeting, all members received an e-mail containing a summary of what was discussed at the meeting and delivery dates. Other issue was the communication between instructors. Information about the project was sent by the coordinator only to the students and not shared with the local instructors.

Lesson learned 4: Do not expect feedback. Solicit responses in time to meet goals. There was a lack of feedback from participants at the beginning of work. For instance, one team has not sent its observations and questions about the requirements. The requirements team waited for this feedback to consolidate the SRS, which generated a warning from the DOSE coordinator. There was a clear fail of communication that resulted in unnecessary wear and tear in the group.

Lesson learned 5: Give clear instructions about work. Guidelines for the requirements teams about the project scope and purpose should have been shared at the beginning of the PUCRS semester (not when the DOSE coordinator course started its activities, a month later). The begin of work was delayed and the group wasted precious work time. Project scope was too large for the deadline and some features system were not developed.

Lesson learned 6: Establish a prior testing plan. Requirements teams should view and monitor development and deliveries. However, until the eve of finalizing the course date in Brazil, the development teams had not passed to the requirements teams the system's on-line address so the system could be tested. So, the requirements teams could not verify the developed systems. They believe that it would be interesting if the requirements teams had been engaged in formal user testing phase. That would have probably promoted a higher engagement of the teams and offered them the feeling that ”all dots tied together”.

Lesson learned 7: Allocate a project manager to the project. The students lacked a project manager to introduce the team, centralize the project information, mediate conflicts, organize the communication, plan work processes and define a project calendar (considering the different locations), distribute roles and responsibilities, define collaborative tools, and explain the working model. The project manager's role was absent and this would have made a difference.

These learned lessons are intended for software developers working in distributed teams and also for students. Thus, one can take the ”good” lessons and aim to follow them, and use the ”negative” lessons as an eye opener and try to avoid them in similar new projects.

Based on the above, we suggest to include a profile of the stakeholders in a shared course page as an simple action to facilitate communication and awareness. Overall, the students indeed agreed that it is more beneficial to learn about GSE in theory before starting a project in practice. In addition, we suggest establishing a Project Manager role to assist the teams and mediate work among the different sites.

We highlight that UI prototypes helped in understanding requirements documentation, mainly in this case in which people of different locations worked together. Other positive point was allowing an experience working with the distributed software development and real life problems (industry) in an academic environment.

V. Discussion

At PUCRS, a GSE course is offered as an optional topic in the academic curricula in the Computer Science undergraduate program. Thus, a few students take the course and have the opportunity to develop their competencies in GSE before start working in the software industry or aim for a graduate degree.

While a variety of studies have been published in GSE field, the most knowledge produced is not about teaching GSE. We have observed that challenges faced by Brazilians students are frequently discussed in the GSE literature. Most of them are related to the distributed nature of GSE projects. For example, the scope of a project and scheduling meetings. Scope changes and maintaining the quality can be extremely challenging in a distributed project team. The inherent lack of visibility into the project progress can increase the project risks [8]. Scheduling meetings is related to communication. Many studies describe communication issues in GSE (e.g., [2] [5] [6]).

Regards to the lessons learned, these are associated with challenges faced by students as show in Figure 3. Through the students' own perceptions of experiences and outcomes associated with being part of a GSE team, we observed from the lessons learned that students experienced complexities of working in a GSE setting.

Figure 3 shows that for each lesson learned there is at least two challenges associated with it. Furthermore, it is possible to observe that lessons learned are distributed in more than one challenge category. For example, ”Lesson learned 1: Getting prior knowledge of the problem-area domain” deals with communication and coordination challenges.

VI. Conclusions

In this study, we present the Brazilian students perceptions in developing activities in a GSE project. As part of the DOSE

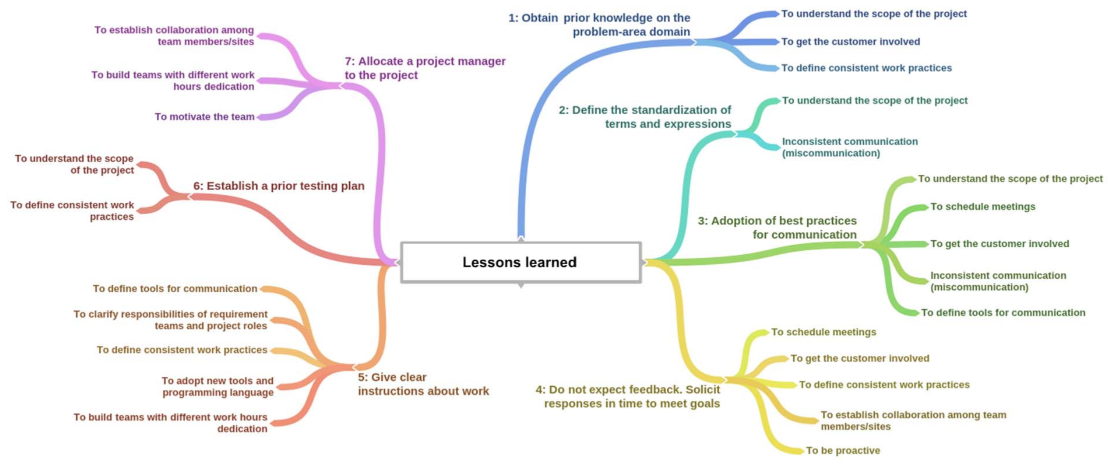

Lessons learned
1: Obtain prior knowledge on the problem-area domain
To understand the scope of the project
To get the customer involved
To define consistent work practices
2: Define the standardization of terms and expressions
To understand the scope of the project
Inconsistent communication (miscommunication)
3: Adoption of best practices for communication
To understand the scope of the project
To schedule meetings
To get the customer involved
Inconsistent communication (miscommunication)
To define tools for communication
4: Do not expect feedback. Solicit responses in time to meet goals
To schedule meetings
To get the customer involved
To define consistent work practices
To establish collaboration among team members/sites
To be proactive
5: Give clear instructions about work
To define tools for communication
To clarify responsibilities of requirement teams and project roles
To define consistent work practices
To adopt new tools and programming language
To build teams with different work hours dedication
6: Establish a prior testing plan
To understand the scope of the project
To define consistent work practices
7: Allocate a project manager to the project
To establish collaboration among team members/sites
To build teams with different work hours dedication
To motivate the team

Fig. 3. Lessons Learned Map.

course, Brazilian students experimented GSE activities with students from other universities. As a result of this study, we identified a set of challenges and lessons learned on preparing graduate students for GSE work.

The experiences described in this study shows the importance on preparing students for GSE work. Students had the opportunity of developing their GSE competences in a 'practical course'. The GSE complexities was discovery by the students throughout the project where they realized the importance of developing competences to mitigate the faced challenges.

In future work, we plan to introduce GSE practices to prepare students for GSE work in our regular undergraduate courses, as well as create new approaches to teach GSE competences. Further research of our lessons learned can help in creating recommendations for non experienced students in GSE. It is interesting to observe that students can also face different challenges in new GSE projects. Thus, future studies can also further investigate challenges reported in this paper.

Acknowledgments

The authors would like to thank the participants of the 2014 DOSE project course session from all partner universities, specially the PUCRS graduate students. The authors acknowledge financial support in this research from CNPq (projects 312127/2015-4 and 406692/2013-0), FAPERGS (project 2062-2551/13-7) and the PDTI Program, financed by Dell Computers of Brazil Ltd. (Law 8.248/91).

References

[1] Dose project. http://se.inf.ethz.ch/research/dose/.

[2] P. J. Agerfalk, B. Fitzgerald, H. Holmström Olsson, and E. Ó Conchúir. *Making Globally Distributed Software Development a Success Story: Proc. International Conference on Software Process, Leipzig, Germany*, chapter Benefits of Global Software Development: The Known and Unknown, pages 1–9. Springer, 2008.

[3] E. O. Conchúir, P. J. , H. H. Olsson, and B. Fitzgerald. Global software development: Where are the benefits? *Communication of the ACM*, 52(8):127–131, Aug. 2009.

[4] D. Damian, A. Hadwin, and B. Al-Ani. Instructional design and assessment strategies for teaching global software development: A framework. In *Proceedings of the International Conference on Software Engineering*, pages 685–690, Shanghai, China, 2006. IEEE.

[5] S. Deshpande, S. Beecham, and I. Richardson. *New Studies in Global IT and Business Service Outsourcing: Global Sourcing Workshop, Courchevel, France*, chapter Global Software Development Coordination Strategies - A Vendor Perspective. Springer, 2011.

[6] 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 *Proc. International Conference on Global Software Engineering, Florianopolis, Brazil*, pages 3–11, 2006.

[7] B. Meyer and M. Piccioni. The allure and risks of a deployable software engineering project: Experiences with both local and distributed development. In *Proc. Int'l Conference on Software Engineering Education and Training, Charleston, USA*, pages 3–16, 2008.

[8] S. Mohan and J. Fernandez. Distributed software development projects: Work breakdown approaches to overcome key coordination challenges. In *Proceedings of the India Software Engineering Conference*, pages 173–182, Mysore, India, 2010. ACM.

[9] M. Nordio, H. C. Estler, B. Meyer, N. Aguirre, R. Prikladnicki, E. D. Nitto, and A. Savidis. An experiment on teaching coordination in a globally distributed software engineering class. In *Conference on Software Engineering Education and Training, Klagenfurt, Austria*, pages 109–118, 2014.

[10] M. Nordio, C. Ghezzi, B. Meyer, E. D. Nitto, G. Tamburrelli, J. Tschannen, N. Aguirre, and V. Kulkarni. Teaching software engineering using globally distributed projects: the dose course. In *Collaborative Teaching of Globally Distributed Software Development - Community Building Workshop*. ACM, 2011.

[11] M. Nordio, R. Mitin, and B. Meyer. Advanced hands-on training for distributed and outsourced software engineering. In *International Conference on Software Engineering, Zurich, Switzerland*, pages 555–558, 2010.

[12] M. Nordio, R. Mitin, B. Meyer, C. Ghezzi, E. D. Nitto, and G. Tamburrelli. The role of contracts in distributed development. In *International Conference on Software Engineering Approaches for Offshore and Outsourced Development, Zurich, Switzerland*, pages 117–129, 2009.

[13] A. Zeid and R. El-Bahey. Establishing a global software development course: A cultural perspective. In *Frontiers in Education Conference, Oklahoma, USA*, pages 1695–1701, 2013.