A DSD Experience for Game Development Among Colombian Students

Carlos Mario Zapata-Jaramillo
Universidad Nacional de Colombia, Colombia
cmzapata@unal.edu.co

Bell Manrique-Losada
Universidad de Medellín, Colombia
bmanrique@udem.edu.co

Liliana González-Palacio
Universidad de Medellín, Colombia
ligonzalez@udem.edu.co

María Eugenia González
Institución Universitaria Salazar y Herrera, Colombia
maria.gonzalezp@salazaryherrera.edu.co

ABSTRACT

Teamwork skills, risk analysis, project management, requirements elicitation, and negotiation among clients and users are required competencies for implementing software development processes. Development of such competencies is hard when you follow traditional teaching methods. For this reason, an ongoing challenge educators face is related to finding new strategies to be used in the classroom for achieving this goal. Distributed Software Development (DSD) is a way to combine such competencies in a real environment. Consequently, in this paper we propose a DSD experience based on the development of a game in order to give the students a practical view about DSD. This approximation in academic environments allows the students for becoming active during their learning process. Also they learn to be careful with their challenges and risks when they start to develop software applications in industry.

INTRODUCTION

Software development process is complex and it demands a set of soft competencies to its practitioners. In fact, students need to know in depth teamwork, risk analysis, project management, requirements elicitation, negotiation among clients and users, etc., in order to be considered successful in the software engineering industry.

Lectures and toy projects are the main strategies used by software engineering educators. Even though such strategies have been used for years in software engineering, they are recognized to be unsuccessful to provide learning related to soft competencies. So, several other strategies should be used in order to improve the way students acquire soft competencies.

According to Ding and Yang (2012), DSD courses offer collaborative help for improving cross-cultural understanding among students, so they can be used as a solution for acquiring knowledge in software engineering, while encouraging the development of soft competencies. Implementation of collaborative courses should be promoted, involving academic managers, and fostering the participation of educators beyond the academic experience.

With this aim, we propose the development of a game by using a DSD experience based on the work of students coming from three Colombian Universities: Universidad Nacional de Colombia (UNAL), Universidad de Medellín (UdeM), and Institución Universitaria Salazar y Herrera (IUSH). The work described in this paper was carried out during a 16-week academic semester. The software development activities performed by students were related to requirements elicitation; functional and non-functional requirements; use case design, detailed design, prototyping, and implementation of a software application. The work teams had to meet each other in order to: (i) produce, deliver, and present specific work products; (ii) interact with work teams coming from different universities; (iii) evidence collaborative work and cooperative communication practices; (iv) assign roles and responsibilities; (v) reuse and share advances in work product design.

Fagerholm et al. (2013) recognize one of the main advantages of DSD in academic environments: the students could take ownership of their relevance in the process and—in this way—they learn to be careful with their challenges and risks when starting to develop software in industry. Even though the students are located in the same city, we discover cultural, behavioral, and even methodological differences among the students—and also among professors leading the project.

This paper is organized as follows: first, we present some background related to DSD; then, we state the problem we face; next, we propose the DSD experience; after that, we discuss the lessons learned; and finally, we summarize the conclusions of this experience.

BACKGROUND

Educators have employed several teaching strategies and

experiential learning techniques in order to improve teaching effectiveness on competencies like teamwork, risk analysis, project management, elicitation requirements, and negotiation among clients and users. These skills are essential because modern software engineering is global and often done in globally distributed teams. Also, software industry is looking for decentralizing application development, and promoting architecture, knowledge, and components reuse (Damian *et al.*, 2012). Distributed Software Development (DSD) is an ideal scenario allowing students for obtaining these and other skills. DSD allows team members to be located in several remote locations during the software development lifecycle, thus making up a network of distant sub-teams (Jiménez *et al.*, 2009). DSD allows software companies for solving global issues like geographical distance, decentralized training, and cultural differences.

DSD has several examples related to industrial and educational studies. In this Section we analyze some contributions. In their work, Jiménez *et al.* (2009) make a full systematic state-of-the-art review about DSD. They explore some DSD aspects researchers have focused until now. Jiménez *et al.* (2016) extract the following DSD critical success factors: (i) intervention of human resources; (ii) improvement based on the company needs; (iii) human resource training of DSD tools and processes; (iv) activity registry with information about pending issues, bugs and people; (v) knowledge management and communication among team members.

Sengupta *et al.* (2006) report a research agenda for DSD concepts with four main areas: collaborative software tools, knowledge acquisition and management, testing in a distributed setup, and process and metrics issues.

DSD contributions are focused on tools, processes, and metrics, but few authors have research on DSD educational environments. Kuhrmann and Münch (2016) propose a course unit related to an environment in which students can learn and experience the role of different communication patterns in Distributed Agile Software Development (DASD). Students understand the importance of communication by experiencing the impact and constraints of communication channels and the effects on collaboration and team performance. The authors provide a detailed design of the course unit to be implemented in further courses (Kuhrmann & Münch, 2016).

Fortaleza *et al.* (2013) discuss the importance of developing communication skills in students. They present an observational study aiming to improve communication and collaboration skills by using DSD. They demonstrate the use of DSD in classroom is accelerated by the adoption of communication and collaboration practices.

Damian *et al.* (2012) describe the goals, design, and initial challenges found in teaching a global DSD course in collaboration between the University of Victoria—Canada—and Aalto University—Finland. The collaborative development is based on the Scrum methodology. They conclude the main challenges they face are differences cultures, time, courses, and curricula, as well as technical and time-zone issues.

PROBLEM STATEMENT

Software engineering industry must be considered in modern curricula in order to respond to its demands by adding or improving graduate student skills. Commonly, software development companies express they are dissatisfied with the lack of some competencies exhibited by graduates/alumni from Higher Education. "Soft competencies" are the typically the most argued missing competencies—e.g. interpersonal relations, personal attributes, communication, and leadership, among others. Considering software engineering as an inexact science—prone to be directly affected by perceptions, subjectivities, and dependence on expertise of those responsible for its execution at software projects—we can argue traditional teaching methods like lectures are ineffective for developing the soft competencies relevant to this knowledge area. For this reason, an ongoing challenge educators face is related to find new strategies to be used in the classroom for achieving this goal.

Fortaleza *et al.* (2013) reinforce the importance of simulating real environments in academic environments, including the ability to unexpectedly change the requirements. We can develop socio-technical skills, soft competencies, and effective work in a global context by following classroom strategies like DSD, in opposition to traditional strategies commonly used for developing such competencies. DSD has been difficult to disseminate/sustain due to high adoption costs and difficulties to find teaching partners (Faulk & Young, 2012). So, we need to explore the application of DSD on educational environments.

Most practices in software development classrooms remain hidebound in course content and student experience, since most of them are introduced in projects where teams, conditions, and needs are simulated. In addition, such simulated projects are made by fostering interactions among classmates in simulated environments—with the same time zones and cultures. In actual industrial practices, the culture, geography, and time zone differences, as well as the coordination and control problems, are common problems to be faced (Bosnić *et al.*, 2011; Cramton & Hinds, 2005). The aforementioned reasons lead us to propose a DSD experience based on software development.

PROPOSAL: DSD EXPERIENCE

EXPERIENCE OVERVIEW

The DSD experience is based on the development of a game in order to give the students a practical view about DSD. The students coming from three Colombian Universities should design and implement a game as a learning strategy for a particular scenario. The experience framework is based on four different software engineering courses belonging to Systems Engineering programs: information systems, requirements engineering, software engineering I, and software engineering II. Such courses have related learning objectives, which was relevant for implementing the experience. The general goals of the experience are:

1) Developing—in a real environment—competencies related to teamwork skills, risk analysis, project management, requirements elicitation, and negotiation among clients and users;
2) Teaching software engineering skills and strategies by emulating a real-world environment by using DSD practices;
3) Practicing several collaborative tools and technologies for providing effective support to communication, software design, software construction, and testing among global teams;
4) Exposing students to cultural differences in a DSD context for improving cross-cultural understanding among them.

SCENARIO FOR THE GAME DESIGN

A concept of operations was defined comprising the primary events for designing a simulation game. Such a concept was edited based on previous approaches with other universities, as well as some rounds of play with similar scenarios which served as the basis for this scenario. The concept of operations is named "software boulevard," a simulation of a software company which develops software projects. Each company includes four roles: project leader, analyst, developer, and tester, who are intended to bid and develop projects in order to increase the bidding capacity of the company (see the full concept of operations in the Appendix). A list of secondary events was also specified in a round of play to facilitate the development of the assignments for each team work. Professors agree on and define a set of assignments aligned to the learning objectives of each course and institution, and considering those which expectedly have a positive influence upon the previous goals of the DSD experience.

GAME SETUP

Students belong to three Colombian Universities located in the same city (Medellín, Antioquia): Universidad Nacional de Colombia (UNAL), Universidad de Medellín (UdeM), and Institución Universitaria Salazar y Herrera (IUSH). The experience was carried out during a 16-week academic semester in the following three stages:

Phase 1: Preparation

During the first eight weeks, work teams were formed with two students belonging to the software engineering course of UdeM and one student of the software engineering course of the IUSH. Such work teams designed the requirements document with functional and non-functional requirements, and use cases with their respective contracts.

During the last eight weeks, work teams were formed by students from UNAL and UdeM. These work teams were in charge of the detailed design, prototyping, and implementation of the software boulevard simulation.

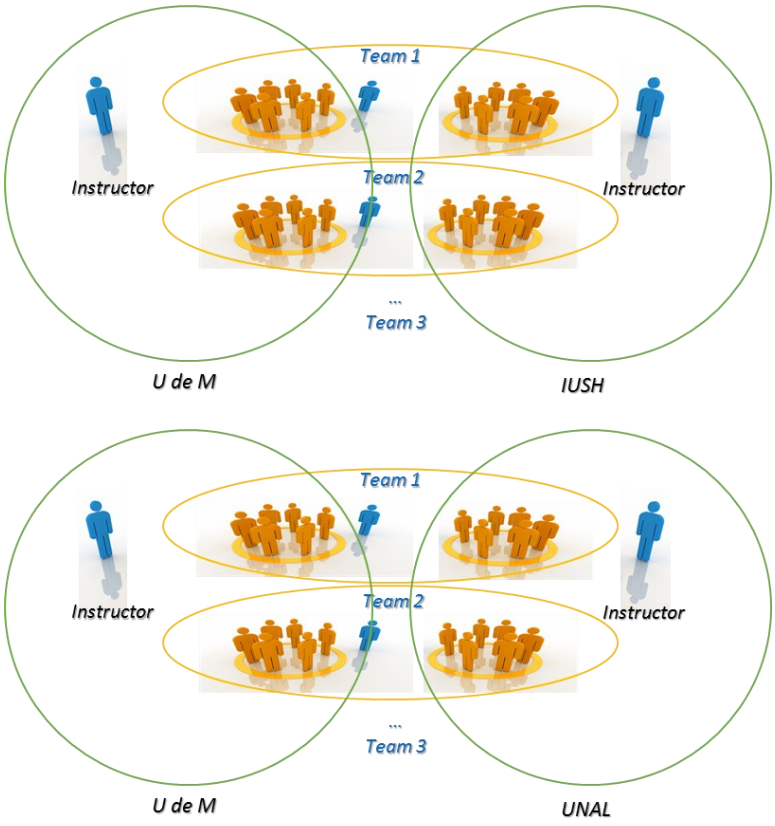
Phase 2: Interaction

- *Group 1: Interaction UdeM-IUSH***.** Based on the common concept of operations, students started the interaction UdeM-IUSH. As a starting point of this interaction, the students review and analyze the concept of operations, the specifications, and the game rules.
- *Group 2: Interaction UdeM-UNAL***.**

Student interaction should comply a set of rules in the DSD interaction. The most relevant rules are the following:

(i) producing, delivering, and presenting specific work products;
(ii) interacting with work teams coming from different universities;
(iii) evidencing collaborative work and cooperative communication practices;
(iv) assigning roles and responsibilities;
(v) reusing and sharing advances of work product design.

EXHIBIT 1.
STRUCTURE OF THE TEAMS



Phase 3: Analysis and result presentation

The analysis of the experience and the generated results—including the inquiry of the perceptions from the students and professors—was the last phase of the DSD experience. Students should generate work products and results defined by the work team, and present a report comprising the evidences of the process, DSD interaction with the other university, and the results achieved.

IMPLEMENTATION

Teams and Roles

We split the students into four distributed teams, each consisting of 7-8 members. Each team comprises of 3-4 students from UdeM and 4-5 students from IUSH or UNAL (see Exhibit 1). The product owner, was a member of UNAL. As the Group 1 students had worked on the project for two months before the Group 2 students joined, one of them with the professor were selected to function as the joint for the following teams.

Project Initiation

The UNAL professor visited the other two institutions before the first interaction started (Group 1), to give a 'face' behind the idea and the whole project by using lessons learned in previous experiences. When the Group 1 started, such professor and a subset of each team kept in touch interchanging extensive technical knowledge of the project and suggesting DSD practices.

Development Process

The development process was an implementation of a particular method adopted by each team work looking for distributed projects, along two months each interaction.

At the beginning of each phase, the teams do synchronous planning and task assignment, according to the requirements specified. Work teams participated in a joint videoconference session and similar sessions for tracking and progress control as an initial interaction with stakeholders.

Each team work had a module of 'software boulevard' on your responsibility, as follows:

MODULE 1: ESTIMATION

This is the module for the Project manager who has the following main processes:

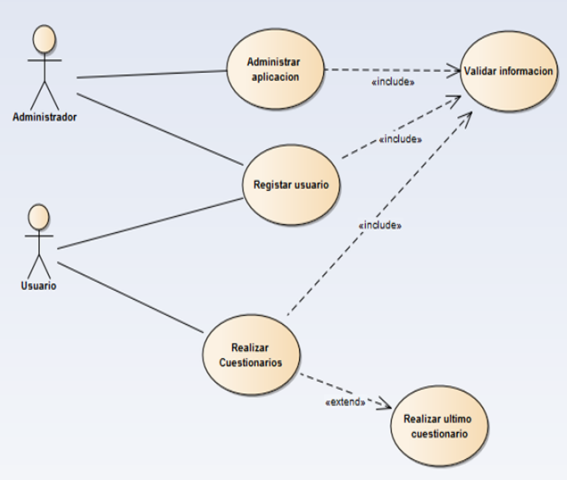
- Estimating cost and time
- Puzzles solving for obtaining resources
- Recruiting team members

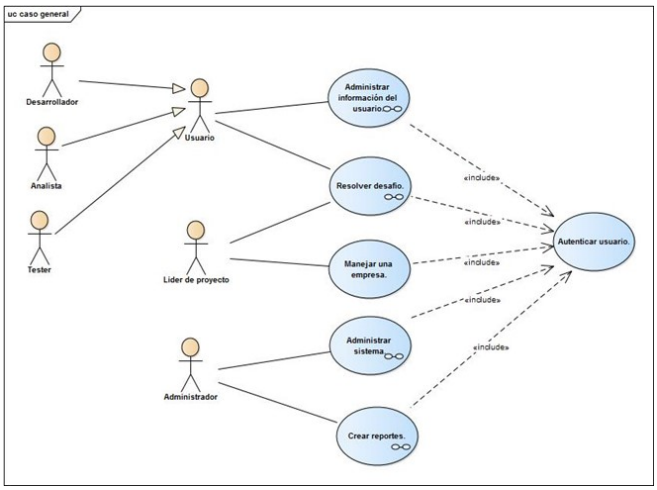
MODULE 2: ADMINISTRATION

This module belongs to the administrator and it has the following main processes:

- Registering and updating companies, roles, and users
- Registering and updating question bank

EXHIBIT 2.
EXAMPLES OF WORK PRODUCTS DESIGNED BY TEAMS BELONGING TO GROUP 1





Software Boulevard Pág. 1

Juan fernando Sepúlveda.
Stiven Sánchez rico.

Requirements Catalogue & Use Cases Modeling

Project: Software Boulevard
Version 01

Software Development Framework
Programa de Ingeniería de sistemas
UdeM- IUSH - UNAL

- Defining diagrams for solving puzzles
- Creating projects
- Monitoring process

MODULE 3: SOLVING QUESTIONS

This module belongs to all team members (analyst, developer, tester) and it has the following main processes:

- Solving questions for role certification
- Solving questions for developing projects
- Obtaining skill level

MODULE 4: COMMUNICATIONS

This module belongs to all team members (analyst, developer, tester) and it has the following main processes:

- Communication among team members by using internal e-mail
- Communication among companies by using internal e-mail
- Communication administrator-companies by using internal e-mail

Group 1: Interaction UdeM-IUSH.

We show some evidences of the work products designed by the work teams in this interaction in Exhibit 2. Work teams designed the requirements document comprising functional and non-functional requirements, and use cases with their respective contracts.

Group 2: Interaction UdeM-UNAL.

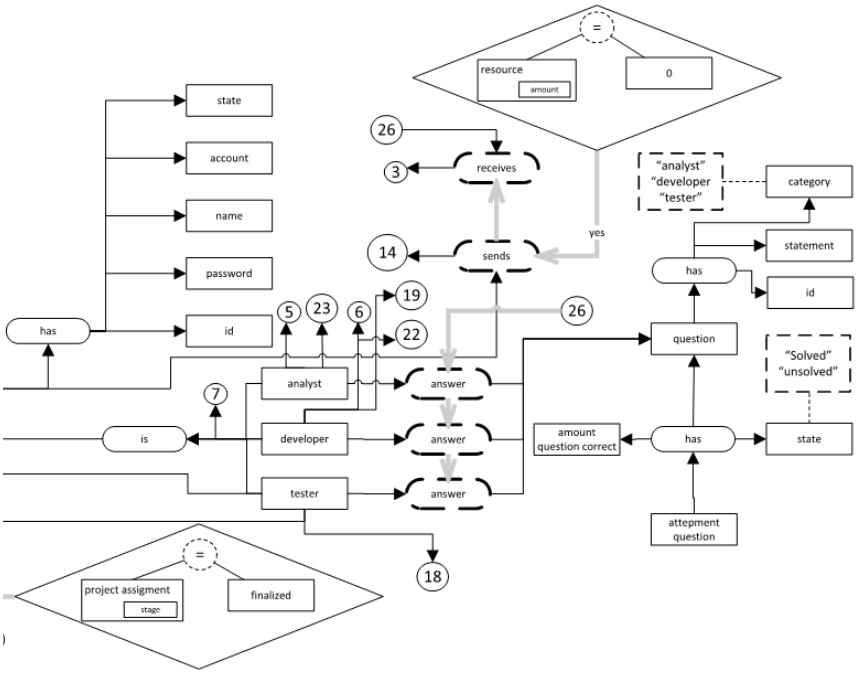
In this group, the work teams from UdeM and UNAL are in charge of generating the work products related to the detailed design, prototyping, and implementation of the 'software boulevard' simulation. Some of such work product are shown in Exhibit 3.

LESSONS LEARNED

One of the situations described in this work was evident during the development of the experience made by the first group of work teams (UdeM & IUSH): the ambiguity of the concept of operations. In fact, professors should review together several times the concept of operations. Such review involved reprocesses and disagreement among several teams.

Bosnic *et al.* (2013) provide a list of the risks associated

EXHIBIT 3.
EXAMPLES OF WORK PRODUCTS DESIGNED BY TEAMS BELONGING TO GROUP 2



code	name	description	formule	related business rules
BR001	send message to recruit	when no members of the project manager sends a message to members to recruit	lack member	
BR002	receive message to recruit	receives only available members	members = available	BR001
BR003	send message to answer recruit	the member must send message when receive de message		
BR004	send message to resource request	the member must send message when resource is zero	resource amount = 0	
BR005	send message to project notification	the project manager sends a message when stage of project assignment is in analysis	stage project assignment = analysis	BR008

with the orientation of a DSD course. Our work led us to add another item to the list: plagiarism in geographically distributed environments. Some students can take advantage of presenting other team work as proper, since they believe professors involved in the process are unable to detect fraud. So, another "soft skill" we can argue for DSD projects in the classroom is related to honesty and ethics.

In the first phase of the work, some inconveniences arose, since some information was not transmitted to the groups of the two universities in a synchronous way. As a consequence, some students partially advanced the project with minimum interaction of the other group. So, they supposed they already prepared much of the work and rejected the contributions they received from members of the other University. This situation accentuated one of the reasons presented by Brindley *et al.* (2009) related to the resistance exhibited by some students to work in a group, considering their peers are unproductive. So, another lesson learned is that both the planning of the subject matter on which the students are going to do a collective construction, and the supporting resources should be synchronously transmitted to the study groups.

Professors in charge of the first part of the project defined the evaluation metrics for that part, and also jointly reviewed and qualified the works in order to assess the same items for the three work team members. However, neither the means nor the format for a unified assessment of the students was allowed, avoiding the assessment of their joint work until the end of the project.

The evaluation metrics included the student personal perception about the DSD model, but it should have been more precise and rigorous for the assessment of the collaborative work. Therefore, the metrics should not be limited to validating the technical consistency of the work products made by each student, but to showing the joint work during the whole process.

Prior to the beginning of the interactions, we need to make each work team aware about the importance of DSD and the benefits it brings to their training process. Filipovikj *et al.* (2013) suggest some tips: remember students we are different in formation, position, and status. Such differences can generate some attitudes like "if they do not ask me, I do nothing," "I will not respond to them in time," "I will not waste my time meeting with them," and "what they have done does not work."

Another lesson learned was the need to define common deliverables, despite the differences among courses. In the second part of the project, we tried to continue with the deliverables defined in each university, but the differences among them and the way to assess them generate great concern to some students and anger to others.

IUSH students did not know the result of the work they started, leading to demotivating some students. Also, lack of responsibility was generated in others, as they "disconnected" the result and the consequences of their actions. In this case, the lessons learned are related to a final feedback was required with all the working groups involved in the project life cycle.

Stakeholders should not only belong to the students of a single university. This allows the orientation of the work to be biased by the interests of a group. To this concern, Lima and Almeida (2012) make some suggestions for group leaders. From the experience, we believe collaboration and good relations among faculty of the groups is perhaps more relevant than the collaboration among students. In the case of this project, collaborative faculty work towards was not enough by the end of the project and some students belonging to the first group did not know the result of the software project.

We would expect students coming from the same city to be similar in culture and knowledge. However, this DSD experience led us to detect several culture differences among students. For example, UdeM students were part-time workers while UNC students were full-time students, leading to difficulties for arranging meetings among them. Also, professors had different ways of teaching and pedagogical methods. One of the main lessons learned in this project is related to the acceptance of differences among participants of the DSD experience.

CONCLUSIONS

In this paper, we summarized the experiences acquired during the development and implementation of a DSD project between UNal, UdeM and IUSH, in their respective software engineering courses of the Systems Engineering program, which involved four different leading professors and the same number of courses at different levels of training. During the first eight weeks, work teams were formed with two students belonging to the software engineering course of UdeM and one student from the software engineering course of the IUSH. Such work teams designed the requirements document with functional and non-functional requirements, and use cases with their respective contracts. The rest of the semester, some other work teams were formed with students from UNAL and UdeM. These work teams were in charge of the detailed design, prototyping, and implementation of the application. A common concept of operations was used as a starting point of this experience.

Although the DSD experience was new for most of the students and professors, the initiative was adopted without any reticence. Consequently, the participant professors were aware about the importance of adapting and adopting new trends of globally distributed working groups. We could also evidence the pedagogical commitment of the leading professors of each group when assuming the challenge of incorporating new trends in their work plans.

From the lessons learned, we can summarize that—prior to the beginning of the courses—we need to carefully plan the activities, meetings and agreements to be carried out during the whole life cycle of the project to be developed. The planning should be rigorous and at the same time flexible in order to adapt to possible unexpected changes arising when the activities are developed, in a similar way to projects in business environments.

The implementation of the DSD project among four groups of three universities made possible to highlight the importance of developing the "soft skills" of all those involved in the project (professors and students), especially: ethics, sociability, empathy, active listening, time optimization, work under pressure, and respect the opinions of others. By the end of the experience, a verbal survey was carried out among the participants and the most common concern among students was the difficulty for arranging meeting places with their colleagues from other universities.

Professors of the first phase detected some differences among theories and examples provided by each group, and leading to apparent theoretical contradictions in the way professors teach. However, software engineering is not an exact science, and opinions should be different among professors, since problems lack a single solution in this science. Acceptance of differences among participants is one of the most remarkable lesson learned in this project.

REFERENCES

Bosnić, I., Čavrak, I., Orlić, M., Žagar, M. & Crnković, I. (2011). Avoiding Scylla and Charybdis in Distributed Software Development Course. In *Proceedings of CTGDSD'11*, Honolulu, USA 2011.

Bosnic, I., Ciccozzit, F., Cavrak, I., Mirandola, R., & Orlic, M. (2013). Multi-dimensional assessment of risks in a distributed software development course. In *3rd International Workshop on Collaborative Teaching of Globally Distributed Software Development (CTGDSD),* San Francisco, USA, 2013. 6-10.

Brindley, J. E., Walti, C., & Blaschke, L. M. (2009). Creating Effective Collaborative Learning Groups in an Online Environment. The International Review of Research in Open and Distributed Learning, 10(3).

Cramton, C.D. & Hinds, P.J. (2005). Subgroup Dynamics in Internationally Distributed Teams: Ethnocentrism or Cross-National Learning. In B.M. Staw and R.M. Kramer, eds., *Research In Organizational Behavior*. 231-263. Oxford: Elsevier Ltd.

Damian, D., Lassenius, C., Paasivaara, M., Borici, A., & Schröter, A. (2012). Teaching a globally distributed project course using Scrum practices. In *Collaborative Teaching of Globally Distributed Software Development Workshop (CTGDSD), 2012*. 30-34. Zurich, Switzerland.

Ding, J., & Yang, B. (2012). Teaching Software Engineering with Global Understanding. In *Collaborative Teaching of Globally Distributed Software Development Workshop (CTGDSD), 2012*. 11-15. Zurich, Switzerland.

Fagerholm, F., Oza, N., & Münch, J. (2013). A platform for teaching applied distributed software development: The ongoing journey of the Helsinki software factory. In *3rd International Workshop on Collaborative Teaching of Globally Distributed Software Development (CTGDSD)*, San Francisco, USA, 2013. 1-5.

Faulk, S. & Young, M. (2012). Chapter 6.7: *Teaching Globally Distributed Software Development (DSD): A Distributed Team Model*. University of Oregon, USA.

Filipovikj, P., Feljan, J., & Crnković, I. (2013). Ten Tips to Succeed in Global Software Engineering Education: What Do the Students Say? In: Proceedings of the Collaborative Teaching of Globally Distributed Software Development - Community Building Workshop (CTGDSD3), San Francisco, US.

Fortaleza, L., Vieira, S., Junior, O., Prikladnicki, R., & Conte, T. (2013). Using Distributed Software Development in the improvement of communication and collaboration skills in SE courses: An observational study. In *2013 26th International Conference on Software Engineering Education and Training (CSEE&T).* San Francisco, USA, 2013. 139-148.

Jiménez, M., Piattini, M., & Vizcaíno, A. (2009). *Challenges and improvements in distributed software development: a systematic review*. Advances in Software Engineering, 1-16.

Jiménez, M., Piattini, M., & Vizcaíno, A. (2016). challenges and improvements in Distributed Software Development: a Systematic review. *Data Structure and Software Engineering: Challenges and Improvements*, 225.

Kuhrmann, M., & Münch, J. (2016). Distributed software development with one hand tied behind the back: A course unit to experience the role of communication in gsd. In: *Proceedings of the IEEE 11th International Conference on Global Software Engineering Workshops (ICGSEW)*, Orange County, US.

Lima, C., & Almeida, E. (2012). Five Years of Lessons Learned from the Software Engineering Course Adapting Best Practices for Distributed Software Development. In:; *Proceedings of the Collaborative Teaching of Globally Distributed Software Development CTGDSD*, Zurich, Switzerland.

Sengupta, B., Chandra, S., & Sinha, V. (2006). A research agenda for distributed software development. In: *Proceedings of the 28th international conference on Software engineering*, Shanghai, China.

APPENDIX

CONCEPT OF OPERATIONS: SOFTWARE BOULEVARD

Software Boulevard is a simulation of a software company including four roles: project leader, analyst, developer, and tester. Companies are intended to bid and develop projects in order to increase the bidding capacity K of the company. Bigger numbers of the K unit represent bigger capacity for developing projects. A company starts with zero K and it needs to develop the so-called "instant projects" for increasing the K unit and reaching the K unit demanded by "bidding projects."

Projects are developed by using a waterfall approach. Once the project is assigned to the company, the analyst start to model the project; then, the developer codes the project; finally, the tester tests the project. The way these roles make such activities is simulated by asking several multi-choice questions related to their competencies. Questions can have 1–3 out of 4 answers, but the role doesn't know how many answers a question have. A question is overcoming when all the related answers are correctly selected. The questions can be solved by the roles when the project manager acquire enough resources for the project. One example of analyst question is the following:

QUESTION: Some kinds of UML diagrams are:

a. Goal Diagram
b. Class Diagram
c. Process Diagram
d. State Machine Diagram

The answers to this questions are options b) and d), and the analyst must select such options for passing the question. Similarly, the developer and the tester must answer their questions for completing the project. The waterfall approach demands the analyst must answer all his/her questions before the developer starts to answer his/her questions. Similarly, the developer must finish before the tester starts to answer his/her questions.

Instant projects deliver K units to the company after all the questions defined are answered by the team members.

Bidding projects are different to instant projects because they need the project manager to estimate both the cost and the time of the project. Project managers need to generate resources for estimating projects. Depending on the amount of resources gained by the project manager, the remaining team members can perform their functions, either to develop the project or to improve their competencies. The way a Project Manager generates resources is by solving sliding puzzles of diagrams similar to the Exhibit 4, but with parts of the diagram instead of numbers. Once the Project Manager completes the puzzle, n resources are generated and he/she can start to estimate cost and time of a bidding project. When both values are in the threshold—e.g., 10% above/below the value—of the defined values for the project, such a project is assigned to the company and the team members can proceed to complete the project in a similar way to instant projects. Each team member can perform his/her functions by using the amount of project resources available (resources earned by the project manager). When a team member identifies that there are not enough resources for using, he/she should inform to the manager who must generate more resources.

EXHIBIT 4
EXAMPLE OF A SLIDDING PUZZLE.

Another difference between bidding and instant projects is related to the competencies the roles need for developing the project. Competencies are defined by levels, so instant projects require level zero for all of the roles. However, bidding projects require the specified level for every role (from one to five). Roles can reach their levels by the certification process. Certification is simulated in Software Boulevard by asking training questions, in a similar way to developing projects. A certain level of competency requires an amount of questions to be answered. Also, the turns for estimating project time and cost can be used for answering questions for either certification or project development.

All this process is constantly monitored and reported to all of the participants of the simulation. Several charts can be used for monitoring, for example companies vs. current K units, questions answered by project/role, efficiency of the turns used, bidding/instant projects assigned/completed by company, etc.