Teaching Software Architecture in LATAM Universities through a Distributed CSCL Activity

Fáber D. Giraldo
System and Computer Engineering
Universidad de Quindío, Armenia, Colombia

Sergio F. Ochoa
Computer Science Department
Universidad de Chile, Santiago, Chile

Myriam Herrera
Informatics Institute, Universidad Nacional de San Juan, Argentina

Sergio Zapata
Informatics Institute, Universidad Nacional de San Juan, Argentina

Clifton Clunie
Computer Systems Engineering, Universidad Tecnológica de Panamá

Andrés Neyem
Computer Science Department
Pontificia Universidad Católica de Chile, Santiago,Chile

José Luis Arciniegas
IDIS, Universidad del Cauca, Popayán, Colombia

Fulvio Lizano
Informatics School,Universidad Nacional de Costa Rica

Abstract

Software engineering education is a challenging task in computer science undergraduate programs. These challenges become stronger if we consider the distributed software development scenario that is gaining space everyday into the software industry. Today most higher education institutions are not addressing the challenge of transferring practical knowledge about distributed software development to their students. Such knowledge will be vital for the next generation of software engineers. This article presents a Collaborative and Distributed Learning Activity (CODILA) that helps higher education institutions to deal with the stated problem. The use of CODILA helps students to reach a set of skills that will be required to deal with their professional activities in the distributed work scenario. Moreover, this activity acts as a guideline that helps instructors to transfer this knowledge to the students. This article also present and discuss the results obtained of applying a CODILA in six Latin American universities. Such activity involved the distributed design of a software architecture of a communication infrastructure able to deal with a particular communication problem. The obtained results were satisfactory and indicate that the CODILAs can be used to address the stated challenges.

1. Introduction

An important trend in software engineering is the distributed development of software project. This is a direct consequence of the globalization and today it represents an important and growing work scenario. Recent advances in telecommunications networks, Internet technologies and collaboration tools have provided sufficient infrastructure to allow software companies to work in a distributed way and thus to optimize their investments. These companies take advantage of remote human resources with high technical skills and low cost, in order to increase their competitiveness in software development and maintenance. Globalization in software engineering can reduce the time to market, increase the teams productivity, improve the product quality and reduce the development cost for software organizations [19].

Globalization is characterized by cultural and organizational commitments in different geographical locations and time zones. These commitments may have limitations to achieve a common understanding, build trust and perform a real teamwork [20]. These limitations typically involve the following aspects: communication, coordination, control, supporting infrastructure, and conflict expectations. Therefore, software engineers working in distributed scenario use specific techniques to mitigate these problems, for example, the division of work in modules sufficiently independent to minimize communication between team members [21]. However, in practice, software engineers have still major problems and they continue searching for more effective mechanisms to mitigate the problems and risks inherent to the globalization [22].

Many of software engineering sub-areas present several challenges in their teaching process. Among the factors that influence these challenges are the software scalability and complexity, the software design and construction under cost and quality

constraints, and substantial human aspects that affect all areas/activities of software practices. The complexity of software construction is well known and documented. This complexity includes technical aspects (e.g., the use of development tools and programming languages), cognitive aspects (e.g., understanding key practices) and social aspects (e.g. issues related to teamwork). As result of this multifaceted activity, universities delivering computer science and software engineering programs must put special attention to the tasks performed by students, in order to: (1) improve their capability to analyze and solve problems, (2) expose them to a real world scenario of software development, and (3) teach them to work geographical distributed.

Those requirements generate particular challenges to the software engineering teaching/learning processes, since it is difficult that students of a regular academic program can work on real situations, in order to get the skills required for their professional lives [1].

Today it is required to define a new set of best practices for teaching/learning software engineering. The globalization of the software development has made that educational institution must focus on some particular practices, such as the problem-based learning, the teamwork, and the tasks monitoring and control. These practices typically contribute to develop the soft skills required by the future software engineers to work into a global software development market.

The teamwork involved in software projects can be co-located or distributed, and both are relevant for the future engineers. However, the growing outsourcing tendency makes the distributed teamwork every day more and more required.

This paper presents the model of a Collaborative and Distributed Learning Activity (CODILA) [2], and the results obtained when applying such activity in the design of a software architecture. In this academic experience participated advanced students from six Latin American universities, whom collaborated on a work scenario simulating a distributed software development setting. The obtained results show that the CODILA can be used by educational institutions to deal with the stated challenges.

Next section briefly introduces principles supporting the Computer-Supported Collaborative Learning. Section 3 introduces the distributed CSCL activities and its application to experimental software engineering. Section 4 describes the design of the CODILA model and introduces the template to specify this type of activities. Section 5 presents collaborative distributed experience performed on software architecture and the obtained results. Finally, section 6 concludes and presents the further work.

2. Computer Supported Collaborative Learning

Computer-Supported Collaborative Learning (CSCL) is a pedagogical approach that is supported on the principles of cognitive theories; for example the psychogenetic approach of Jean Piaget [24], Vygotsky's socio-cognitive theory [25], the theory of activity of Leontiev and Galperin [26], and the theory of Cognitive Modifiability of Feuerstein [27]. These theories share the cognitive and meta-cognitive development of students in real, symbolic and dynamic contexts that allow them to recreate their knowledge and potentiate them.

This type of learning is based on practical activities performed by the students in a co-located or distributed way. Therefore it becomes an interesting approach to deal with the stated challenges. According to the CSCL foundations, there is a list of key elements that must be considered in the design of any CSCL activity. Next we briefly present such elements.

- *Positive interdependence*: This is the central element that covers the organizational and operational conditions that must occur within the team to promote and enhance the teamwork. Team members should be supported by each other, and they have to trust in the understanding and success of the teammates. Interdependence must be considered when assigning goals, tasks, resources and roles.
- *Interaction*: The quality and quantity of interactions between team members, driven by positive interdependence, affect directly the learning results. These interactions can be face-to-face and computer-mediated. Such instances allow team members to trace the teammates' activities and realize how close or far is the target of the team. Typically the students are able learn from each partner who interact with them. The team can enrich their knowledge and skills if it has different interaction modes, enhancing thus the reinforcements and feedback.
- *Individual contribution*: Each member must assume the assigned functions. They must also have the space for sharing with the teammates and receive their contributions.
- *Personal and team skills*: The experience of the team must allow each member to develop or enhance his/er personal skills. Such skills must also allow the team's growth and skills acquisition, such as listening, leadership, evaluation, participation, coordination of activities and monitoring.

Studies reported in [12] [13] argue collaborative learning increases participant satisfaction and

motivation, and also prepares them to conduct research activities. Research in primary and secondary education highlights the importance of collaborative learning, since it has been proven that students learn best in non-competitive and collaborative scenarios [15]. Authors in [16] expose that collaborative learning helps to develop critical thinking in students, and also helps to improve interpersonal relationships. It implies that each team member learn to listen, discern and communicate their ideas or opinions to another members with a positive and constructive focus.

In [17], by comparing the results of collaborative learning with traditional learning models, authors have found that students learn more when using collaborative learning. They also develop higher reasoning skills and critical thinking, and feel more confident and accepted by themselves and others. These authors also state that virtual environments can help implement more participatory models of education and expand opportunities to gather, communicate and distribute the knowledge.

In [18] the authors say that collaborative learning seeks to develop personal and social skills in the student, making feel each member of the group responsible not only for his/her learning, but also of other group members' learning.

Students need to be continually questioned about their personal and group performance. It helps to reach some advantages of collaborative learning, such as:

1. *Concerning the execution of group tasks*:
   - Promoting the achievement of goals and the improvement of the work quality, since the team elaborates its proposal based on the members' proposals and solutions.
   - Helping to disseminate and validate the knowledge of group members.
   - Promoting the commitment of each team member.
2. *Concerning the interpersonal relations*:
   - Encouraging the feeling of solidarity and mutual respect, based on the results of teamwork.
   - Enhancing the social skills, and the communication and coordination inside the team.
3. *Concerning the cognitive skills*:
   - Promoting the knowledge generation due to each student is involved in the development of a particular research.
   - Promoting the development of critical thinking and meta-cognitive development.
4. *Concerning the self-development*:
   - Empowering the team members and enhance their self-confidence.
   - Motivation both, the individual and group work.
   - Contributing to decrease the feelings of isolation.
   - Decreasing the level of frustration to criticism and fear of failure.

The main obstacles for collaborative learning are usually three: (1) the resistance to the paradigm change required by students and instructors, (2) counting on an appropriate design of the collaborative activity to apply, and (3) having a good technological solution to support the activity (especially in distributed environments). Therefore, when we design a CODILA, we should clarify the educational principles, stakeholders and the teaching/learning processes that allow students to achieve particular knowledge and skills.

Considering the research findings reported in the literature and the authors experience designing CSCL activities, we recommend to design CSCL activities to support the instructional process of software engineering in a distributed way, by considering the following issues: promoting interaction among teammates, assessing and sharing the individual contributions, and developing personal and group skills (e.g. negotiation, debate, question, make decisions and coordinate). These issues have systematically contributed to promote and enhance the teamwork. However the human aspects involved in the software development process go beyond teamwork.

The combination of technical foundations, practice case studies and CSCL activities makes the teaching of software engineering particularly challenging. It implies that the instructors' profile has to include not only instructional skills, but also professional experience in this area. Since the pool of candidates with academic and professional experience is relatively limited, establishing academic cooperation in this area (with other universities and/or with the industry) can help address this challenge.

3. Distributed CSCL and experimental software engineering

Software industry becomes globally distributed, mainly driven by the development of trends, such Open Source software and developments outsourcing. Therefore software practitioners require significant changes into their formative stages; i.e. in the knowledge and mainly the skills considered by the educational institutions in undergraduate programs.

A model of distributed multi-faceted software development is emerging and demands the

application of engineering principles in new and unfamiliar contexts. For ensuring the successful development of software using distributed resources, software engineers need to count on a solid foundation of project and product management, engineering of organizational processes, and multi-cultural communication skills [3].

The adoption of collaborative technologies in a software development organization involves the introduction of new technologies and work practices within the comprehensive and highly variable activity of software development. Productivity and deliverables of software projects are factors significantly variable. Therefore, it is difficult to assess the impact of the introduction of new technologies in a project [4].

Provided the distributed collaborative work in software development processes and the geographical distribution of software team members, it is mandatory to formulate experiments to validate the application of collaborative techniques in current software engineering teaching/learning scenarios [5]. Research and experimentation in collaborative work applied to software development should focus on factors such as negotiation among participants and stakeholders, distributed communication and coordination, and use of new engineering processes and strategies.

Experimentation scenarios should consider the distance among members as a critical factor influencing the collaborative learning/work [5]. Differences in the physical contexts (i.e. distributed and co-localized), time zones, culture and language will persist despite the use of new technologies. Remote work is enhanced when it is supported by high speed networks, such as RENATA [6] academic network in Colombia and RedCLARA [7] in Latin America. However, some distances aspects will remain present explicitly without possibility of support it even in the future.

It is well-known that software development processes require the participation, expertise and ability of many people working together to achieve the project goals. Therefore, it is necessary to establish new mechanisms that let students to acquire skills of teamwork and effective communication to address these new working scenarios [8]. It is relevant not only from a technical perspective, but also to develop competencies and skills required by students to act as members of a distributed software development team [9]. Trying to deal with this challenge, the consortium of Latin American universities involved in the initiative LACXER (Latin American Colaboratory of eXperimental Software Engineering Research) [23] developed the CODILA model, which is explained in the next section.

4. The CODILA Design

The CODILA involves an instructional process that promotes the distributed learning through collaborative work [2]. Typically student teams work in a collaborative and distributed way to solve a problem (i.e. the team goal). In the case reported in this article the problem to address by each team was the design of software architecture for a secure communication system. The activity execution is typically monitored by instructors in charge of the experience, in order to understand the individuals' and teams' performance.

The CODILA model involves five stages (see Figure 1): Preparation, Theoretical Class (lecture), Local Practice, Distributed Practice and Evaluation. However the local practice is optional.

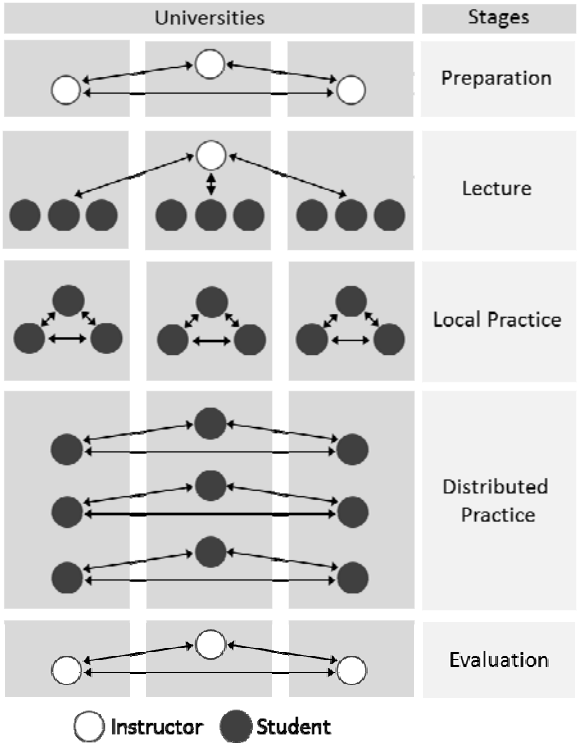

Figure 1.CODILA Model

During *preparation* stage the instructors will collaborate to define the parameters of the activity, such as: the topic to teach, the activity main goal, the instructor who will deliver the lecture, the group size and composition, the technologies that will support the collaboration process, and schedule of the activities.

During the *lecture* the instructor playing the role of lecturer delivers the knowledge about the select topic to all students participating in the activity. It is a synchronous activity in which participate all the students involved in the experience.

The *local practice* is performed during the session after the lecture and its goal is to help students to

assimilate the knowledge delivered by the expert during the lecture. Typically it is a collaborative solving problem activity that involves 90-120 minutes and it is related to the topic tough by the lecturer.

The *distributed practice* involves at least one week and also team of distributed students with a common goal. This practice has three sequential phases: (1) individual work, (2) collaborative session and (3) integrative work. During the first phase the work is divided in as much parts as members has the team. Each member is responsible to complete his/her assignment. During the second phase the team performs a collaborative session where each student explains his/her work to the teammates. Thus the knowledge flows inside the team. Finally, the individual works are integrated and adjusted to form a single product that will represent the outcome for each team. Such outcome is delivered to the instructors.

During the *evaluation* stage such outcome is evaluated and also the collaboration process and supporting technologies used into the experience.

The authors have designed a template to specify a CODILA, which has been useful as mechanism for synchronization, feedback and managing the activity design (see Table 1). The template has also helped to socialize, validate and adjust the experience. Instructors from the participating universities are free to contribute with their suggestions and comments. Table I shows the structure of the template to specify these distributed collaborative experiences.

5. Teaching software architecture using a CODILA

In 2010, we have applied the CODILA model in a collaborative experience for teaching the software architecture topic. In this experience participated six Latin American universities from four countries. These universities were grouped in two subsets as follow:

Track 1 (50 students)

- *Pontifical Catholic University of Chile, Chile (PUC).*
- *Technological University of Panama, Panama (UTP).*
- *University of Cauca, Colombia (UCauca).*

Track 2 (52 students)

- *University of Quindío, Colombia (UQ).*
- *University of Chile, Chile (UChile).*
- *National Autonomous University of Costa Rica, Costa Rica (UNA).*

All students were in the 8th or 9th semester of Computer Science or Informatics undergraduate programs. They grouped in distributed teams with three members each. The track 1 involved 18 teams and track 2 involved and 17 teams. Teams of track 1 were randomly formed, whereas in track 2 we used the students' psychological profile to form the teams.

The challenge to address by the team was the architectural design of a communication infrastructure for the government, were privacy and security are critical attributes. We also ask for software performance and maintainability which are contradictory requirements. The idea was to push the students to negotiate their individual solutions (that considered just one quality attribute) in order to obtain a balanced, integrated and robust design. Usually software quality attributes coexist in states of mutual tension. It demonstrates the need for high level of interaction, collaboration and negotiation between students designing the architectural specification. In the case of globalized projects, architectural decisions are largely made by distributed development teams.

The main goal of this collaborative and distributed architectural design was to simulate a real software distributed design process. The quality attributes addressed by the students were *Security*, *Performance* and *Maintainability.* Each attribute is approached by one student, i.e., each team has a student who is expert in *Security*, other student in *Performance*, and so on. During the collaborative exercise students realize the need to negotiate with their colleagues, to define a unified architecture that considers the three quality attributes mentioned above. Designs negotiation is not a trivial task due to each quality attribute possess a set of architectural tactics and drivers that may generate conflicts between quality attributes.

The lecture about software architecture design was delivered synchronously from the University of Chile to the participating universities, by using Microsoft LiveMeeting as a communication platform. Later, the students began working in a distributed way in order to specify the architecture for the assigned system.

Table 1. Template to Specify a CODILA

SECTION	PURPOSE
Topic	It refers to the topic that will be used in the distributed collaborative activity.
Short description of collaborative experimental activity	It briefly describes the learning activity to be performed. Using this description an instructor must determine whether the activity is adequate for your course.

University leader / name of instructor(s) responsible(s)	University proposing the experience will guide the implementation of the activity. It should be specified the name and contact information of the expert instructor that will be delivering the lecture.
Participating universities	It is the list of participating universities that will be performing the activity. The list also includes instructors and students.
Activity goal	Pedagogical purpose pursued by the distributed collaborative activity. This activity should try to verify a Software Engineering and/or instructional hypothesis.
Experience expectations	It refers to the potential benefits that academy and industry will obtain with the regular execution of this activity.
Hypothesis	The designer of the collaborative experience can define assumptions associated with the activity. It can be of two types: • Interesting for software industry. Hypotheses that attempt to produce results (indications, findings, etc.) that are useful for software companies. • Interesting for software engineering teaching. These are hypotheses that intend to identify how to produce useful results for participants in the teaching-learning process. If the designer defines a hypothesis the following section must be completed.
Experimental design activity	The activity should be adjusted within a rigorous experimental design, since it intends to obtain scientific findings in addition to the educational results.
Instructional model	For each stage of the collaborative teaching-learning model, it must be included: the duration, the responsible, the documentation to be distributed to the participating universities, the documentation that participating universities will deliver to the leading university, deliverables for /of students, etc.
Final evaluation	Define the method used to perform the final evaluation of the collaborative experience.
Analysis of results	Define the procedure that will be used for analyzing the results of the activity. These procedures may involve statistical analysis, observations, etc.
Requirements for students	This section presents the pre-requirements that the students must have to participate in the activity. Usually these pre-requirements are related to previous knowledge or academic level.
Requirements for communication	These are the requirements that must be addressed by the technological infrastructure supporting the distributed experience (e.g., communications platform, tools for supporting collaboration, software development tools and specific software).
Material support	It represents the papers, slides, assignments of the local and distributed practices, and any other supporting document required to perform the activity. This material should be provided by the leading university.
Performing date and time	The most probable date and time to perform the collaborative activities must be established. Such instance must be agreed between all participating universities. The time schedule for the activity should considering the time zone of every university.
Universities and institutional responsible	It specifies the acronyms of participating universities and the responsible of each one.
Expected results	The CODILA specification should include a list of products the team wants to obtain when completing the activity.
Annexes	The CODILA designer must specify the documents related to the experiment, such as: notes, practical exercises, charts and slides. He/she must also include references, in which s/he based the experience, or any bibliographic reference that can be appropriate for students and other instructors.

Distributed practices included three recordings in which the students shown their interactions. Each record was done for a specific deliverable and it had a specific purpose:

- *Recording 1*: First meeting of students. During this meeting the team members were presented and a software quality attribute was chosen by each of them. Thus, each student became responsible to design such aspect of the communication infrastructure.
- *Recording 2*: Each student explained the implication that his/her software quality attribute has on the design of the whole product. The students also exposed his/her proposal for the architectural design, based on his/her software quality attribute.
- *Recording 3*: In this session team members integrate their individual solutions and negotiate to reach an integral design of the communication infrastructure. They also prioritized the features of the final architecture and discussed the technical characteristics of the solution (i.e. the advantages and disadvantages).

Fig. 2 shows a general representation of the collaboration scenario during the lecture stage. It corresponds to the software architecture lecture delivered by the University of Chile through videoconference.

5.1. Obtained Results

The design created by the student teams was evaluated and qualified with a score between 1 and 7. The minimal score for approval was 4.

Analyzing the students' scores (Table 2) we could hypothesize the use of an instrument (based on the psychological profile of the students) to form the teams had a positive impact on the teams' performance, since the track 2 performed better than track 1. However if we consider the teams average scores for each track, we can see the difference between them seems to be minimal.

In order to clarify this point we analyze the qualifications given by students to their joint work as a team; i.e. the level of communication and coordination among team members. The analysis results show us that teams formed randomly (i.e. Track 1) had more interactions than Track 2 teams.

Table 2. Students' Scores

Track 1	***Score***	***Track 2***	***Score***
PUC	***3.9***	*UQ*	***4.9***
UTP	***4.0***	*UChile*	***5.3***
UniCauca	***4.7***	*UNA*	***3.6***
Average:	**4.1**	**Average:**	**4.3**

Initially we hypothesized that this occurred as a consequence of the lack of conflicts between students enrolled in the groups belonging to the Track 2, since the heuristic used to form the teams avoided to assign conflicting people to the same team.

In the practice the conflicts inside a team acted as triggered of discussions and negotiations among teammates, which usually improved their collaboration level.

A more detailed analysis of the collaboration process inside each team, and several interviews done to students after this experience, allows us to identify some key elements to understand the process results and the obtained scores. The first key element identified was a limitation in the heuristic used to form the team based on the students' psychological profile.

Figure 2. Distributed Scenario Involved in the Software Architecture Lecture

Although the use of the students' psychological profile in that heuristic is correct, it does not consider the students' responsibility as part of the criteria used to form the team. In the teams of the track 2, the low scores obtained by some of them were a direct consequence of the lack of responsibility shown by one or more team members.

Some particular situations were identified in the track 1, where students from UCauca reported that their peers in PUC and UTP showed some difficulties in asking for help when they were

problems. In the track 2, UChile students said that students from UNA did not answer emails on time and were engaged with the project. Since we identified the responsibility dimension as relevant for the teamwork, we can understand the low score shown by UNA students belonging to the track 2 (see Table 2).

On the other hand, the students from UQ belonging to the track 2 felt that UChile students did not work as team members, although they accomplished the assigned work. This situation allows us to identify a second key element was present in the experience: the time required to consolidate team.

Some teams of the track 2 had not problems with their members' responsibility; however they did not work as a team. The interviews to these students allow us to realize that any team needs a time to know and gain trust among the team members. Just after that they start to act as a team. Since this experience was quite short (three weeks approximately) none team belonging to the track 2 was able to become a real team. Therefore it does not make sense to design cohesive teams if they are going to work together for a short time period. In other words, in this experience the teams belonging to the track 1 are comparable to those of the track 2.

Analyzing the students' opinions concerning their teamwork in both tracks, it is evident that most students have a positive perception of the work done by their peers. Table 3 shows average values of the scores assigned by the teammates to each student participating in this experience (i.e. the co-evaluation). Such scores range from 1 to 5.

Table 3. Results of Students Co-evaluations

Track 1	*Score*	*Track 2*	*Score*
PUC	***4.1***	*UQ*	***3.5***
UTP	***4.7***	*UChile*	***4.2***
UniCauca	***4.6***	*UNA*	***4.2***
Average:	**4.5**	**Average:**	**4.0**

For measuring the satisfaction of participating students, we built a survey which was filled by the students. It survey consisted of 35 questions grouped in several dimensions related to social skills (e.g. teamwork, leadership, negotiation, technology and empathy) and experience (e.g. attitude and aptitude of the coordinators/contributors). Fig. 3 and Fig. 4 present the general satisfaction level of the teams belonging to each track. These results also support the hypothesis that teams of both tracks are comparable.

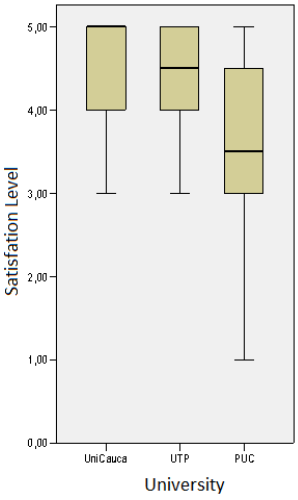

Figure 3. Satisfaction level in Track 1

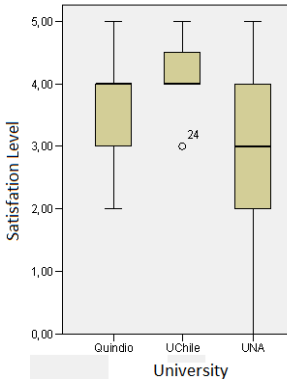

Figure 4. Satisfaction level in Track 2

The Fig. 5 and Fig. 6 present the satisfaction level of the students about the use of the technology that supported their collaboration process. In other words, it measures the appropriateness of the used technologies to support CODILAs. Most students assigned a good score to such activity aspect except those form the UNA, who had serious bandwidth limitations to access Internet. These results show that a regular Internet access is enough to support the several stages involved in a CODILA.

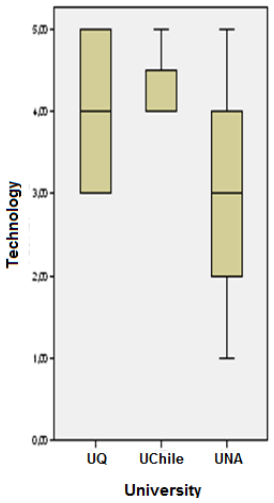

Figure 5. Students' satisfaction in Track 1

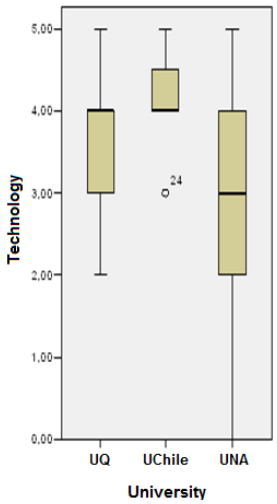

Figure 6. Students' satisfaction in Track 2

Summarizing, students from four of six universities have expressed their satisfaction with the performed experience (UChile, UQ, UTP and UniCauca). In the other two cases (PUC and UNA), the researchers team found a lack of responsibility and motivation in students, influenced by external factors to this CODILA; for example the existence of parallel projects and a low representativeness of the experience score into the course's score.

The participation of students and instructors from six Latin American universities promoted a flexible and dynamic interaction, which contrasted their prior knowledge, technical experiences, life experiences and real needs in each university and country. Students demonstrated interaction, negotiation and leadership skills during the experience. However the detailed analysis of the CODILA results allows the authors to identify several additional goals reached by the participants:

- *Identification and comparison.* The students were able to recognize the essential quality attributes in the software architecture, identifying the features of each attribute (i.e. drivers, tactics and conflicts with other attributes). This topic was initially confronted with the students' prior knowledge and expectations; therefore they established relationships between each quality attribute and the strategies required to obtain it.
- *Analysis.* The students established relationships between the software architecture and their previous learned subjects, and were able to infer the which quality attributes (defined as software requirements) required some special considerations for its achievement.
- *Synthesis.* The students inquired and integrated the relationships between each quality attribute and the context of the proposed problem.
- *Classification.* The students were able to identify, classify and analyze features and essential requirements of each quality attribute involves in the CODILA.

5.2. Learned lessons

Based on this architectural design experience and taking into account the collaborative results of previous experiments carried out since 2008 [10][2][11] by the authors, several lessons learned have been identified:

- *Team size*: The suggested number of members for a team is three. That number allows solving in case of conflicts and it also avoids free ridding attitudes because they are too visible in a small team.
- *Supporting material for the problem solving*: The instructors must provide appropriate supporting material that helps the students during the activity. Supporting material, particularly examples, ensures a better preparation of the students to conduct the experience. This material includes: guides, related examples, practical activities and case studies.
  - *Motivation*: In order to achieve better students' engagement and successful results of the experiment, instructors should motivate the students. Some of the strategies that can be used are the following ones: The participating institutions must assign a significant relevance to the experiments' score.
  - The participation of the entire course is mandatory these experiences. Moreover, such participation cannot be optional for the students.
  - Disseminate and clarify the importance of participating in these experiences. UQ students suggested the preparation of videos in which students that participated in CODILAs can tell to other students about their experiences.
- *Subjects to teach/learn*: The topics selected to teach/learn in a CODILA must have the following characteristics: i) students must have prior knowledge in the subject, ii) participating institutions must count on infrastructure to perform distributed practical activities, iii) the subject should be self-contained (i.e. as independent as possible from other subjects). Instructors must ensure that the activity will be relevant for the professional lives of the students.
- *Interaction:* The communication platform used by the students to interact and share their work should selected by the students. It is not recommended the instructors push the student to use a particular software.
- *Monitoring:* Based on previous experiences, the authors recommend performing evaluations that allow instructors monitoring the teaching and learning process.
- *Social interactions:* In this type of activity social interactions and cultural exchange help student to generate trust with the teammates. It contributes to improve the results of the collaboration process. This finding is also stated by the socio-constructivism.

6. Conclusions and further work

This article presents the results of applying a collaborative and distributed learning activity (CODILA) to teach software engineering practices in several Latin American universities. These collaborative activities are framed within a Latin American initiative that tries to establish a co-laboratory to teach, experiment and research on Software Engineering.

A CODILA involving the architectural design of a software communication infrastructure was presented and discussed. The obtained results show complex interactions between students, due to the

experimental scenario proposed by researches. It reflects that designing software architectures implies an important challenge for software developers.

The results obtained in the experience were encouraging. Since the topic is complex to address by undergraduate students, the score were not high. However most students felt comfortable with the activity and they think that CODILAs can help them to address the challenges of a distributed software development scenarios. Instructors participating in the experience also felt highly comfortable with the activity.

Evidence exposed in this paper must be corroborated with the application of new experiences in this topic, and evaluating results obtained in the anonymous surveys. For further works we will measure the maturity level of previous experiences and we will propose a distributed course in globalized Software Engineering.

Acknowledgements

This work is partially funded by the project entitled “Fortalecimiento de la Red de Investigación Aplicada en Ingeniería de Software Experimental", in the II Call of “Proyectos de Fortalecimiento a Redes Interuniversitarias”, Ministry of Education, Argentina. Authors also thank to CINTEL Colciencias RENATA (Colombia) for partial funding of this work through the Project entitled "Red Latinoamericana de Investigación Aplicada en Ingeniería de Software Experimental", Grant IF-007-09 (Call Colciencias 487 - RENATA 2009), LACCIR through the grant R1209LAC003 and RedCLARA through ComCLARA 2010 call.

References

[1] Ellis, J.C., Demurjian, A.S., Naveda, J.F. (2009). *Software Engineering: Effective Teaching and Learning Approaches and Practices*. IGI Global.

[2] Collazos, C., Ochoa, S.F., Zapata, S., Lund, M.I., Aballay, L., Giraldo, F., Torres de Clunie, G. (2010). *CODILA: A Collaborative and Distributed Learning Activity Applied to Software Engineering Courses in Latin American Universities*. The 6th Intl. Conf. on Collaborative Computing: Networking, Applications and Worksharing.Chicago, USA.

[3] Hawthorne, M., Perry, D. (2005). *Software Engineering Education in the Era of Outsourcing, Distributed Development, and Open Source Software: Challenges and Opportunities*. In Proc. of the 27th Intl. Conf. on Software Engineering (ICSE'05). ACM Press, pp. 643-644.

[4] Whitehead, J. (2007). *Collaboration in Software Engineering: A Roadmap*. Proc. of Future of Software Engineering (FOSE'07). IEEE Press, pp. 214-225.

[5] Olson, G., Olson, J. (2000). Distance Matters. *Human-Computer Interaction* 15 (2), pp. 139-178.

[6] *National Academic Network of Advanced Technologies (RENATA)*. URL: www.renata.edu.co. Last visit: August 2011.

[7] *Latin American Cooperation of Advanced Networks (CLARA)*. URL: www.redclara.net. Last visit: August 2011.

[8] Hawthorne, M., Dewayne, E. (2005). *Software Engineering Education in the Era of Outsourcing, Distributed Development, and Open Source Software: Challenges and Opportunities*. Proc. of the 27th Int. Conf. on Software Engineering (ICSE), pp. 643 - 644. St. Louis, USA.

[9] Bareiša, E., Karčiauskas, E., Mačikėnas, E., Motiejūnas, K. (2007). *Research and Development of Teaching Software Engineering Processes*. Proc. of the International Conference on Computer Systems and Technologies. Bulgaria.

[10] Giraldo, F., Collazos, C., Ochoa, S.F., Zapata, S., Torres de Clunie, G. (2010). *Teaching Software Engineering from a Collaborative Perspective: Some Latin-American Experiences*. Proc. of Workshops on Database and Expert Systems Applications (DEXA), pp. 97-101.

[11] Collazos, C., Zapata, S., Lund, M.I., Aballay, L., Ochoa, S.F., Giraldo, F., Clunie, C., Torres de Clunie, G., Anaya, R. (2010). *A Collaborative Model to Teach and Learn Software Engineering* (In Spanish). Latin American Conference on Higher Education. Asunción, Paraguay.

[12] Sheridan, J. (1989). Rethinking Andragogy: The Case for Collaborative Learning in Continuing Higher Education. *Journal of Continuing Higher Education* 37 (2), pp. 2-6.

[13] Warmkessel, M., Carothers, F. (1993). Collaborative Learning and Bibliographic Instruction. *Journal of Academic Librarianship* 19, pp. 4-7.

[14] Bruffee, K. A. (1987). The Art of Collaborative Learning: Making the Most of Knowledgeable Peers. *Change* 19 (2), pp. 42-47.

[15] Leidner, D., Jarvenpaa, S. (1995). The use of information technology to enhance management school education: a theoretical view. *MIS Quarterly* 19 (3), pp. 265- 291.

[16] Millis, B. J. (1996). *Cooperative Learning*. The University of Tennessee at Chattanooga Instructional Excellence Retreat. Available at: http://www.utc.edu/Teaching-Resource-Center/CoopLear.html. Last visit, July 2011.

[17] Barab, T., Merrill, H. (2001). Online Learning: From Information Dissemination to Fostering Collaboration. J*ournal of Interactive Learning Research* 12 (1), pp. 105-143.

[18] Lucero, M., Chiarini, M., Pianucci, I. (2003). *A Collaborative Learning Environment for the ACI* Environment (in Spanish). White paper. Computer Science Department, San Luis National University URL: www.dirinfo.unsl.edu.ar/~profeso/PagProy/articulos/Lucero%20Cacic%202003.pdf.

[19] Jimenez, M., Piattini, M., Vizcaino, A. (2009). Challenges and improvements in distributed software development: A systematic review. *Advances in Software Engineering* 2009, pp. 1-14, Article ID 710971.

[20] Holmstrom, H., Fitzgerald, B., Agerfalk, P.J., Conchuir, E.O. (2006). Agile Practices Reduce Distance in Global Software Development. *Information Systems Management*, pp. 7–26.

[21] Herbsleb, J., Grinter, R. (1999). Architectures, Coordination, and Distance: Conway's Law and Beyond. *IEEE Software* 16 (5), pp. 63–70.

[22] Hossain, E., Babar, A.M., Paik, H., Verner, J. (2009). *Risk identification and mitigation processes for using Scrum in global software development: A conceptual framework*. In: Proceeding of the Asia Pacific Software Engineering Conference, APSEC'09, pp. 457–464.

[23] Latin American Colaboratory of eXperimental Software Engineering Research (LACXER). URL: www.lacxser.org. Last visit: August 2011.

[24] Briones, G. (1995). Prepareness and Evaluation of Educational Projects (in Spanish). Andrés Bello, Chile.

[25] Vygotsky, L.S. (1978). Mind and Society: The Development of Higher Mental Processes. Cambridge, MA: Harvard University Press.

[26] Galperin, P.I. (1992). The Problem of Activity in Soviet Psychology. Journal of Russian and East European Psychology, 30(4), pp. 37–59.

[27] Tebar B.L. (2009). The Instructor as Mediator of the Learning Process (in Spanish). Magisterio Press, Bogotá, Colombia.

