A Training Tool for Global Software Development

Miguel J. Monasor
University of Castilla-La Mancha
Campus Universitario s/n, 02071, Albacete, Spain
MiguelJ.Monasor@gmail.com

Aurora Vizcaíno, Mario Piattini
Alarcos Research Group, Institute of Information Technologies & Systems
Escuela Superior de Informática
University of Castilla-La Mancha
Paseo de la Universidad 4, 13071, Ciudad Real, Spain
{Aurora.Vizcaino, Mario.Piattini}@uclm.es

Abstract— **In recent years, the application of Global Software Development (GSD) has grown significantly and its efficiency has been proven. However the collaboration between distant members is not always easy and some challenging issues must be confronted. The main problems are related with the communication with distant members, especially when cultural and language differences appear.**

Members involved in GSD must develop specific skills in order to confront these problems. On the other hand, the training to develop these skills is not easy as the practice in real projects is too risky, and simulating the complexity of real environments is not always possible and requires the coordination with distant institutions. In this work we present a training tool that will help universities and companies to achieve this training by simulating GSD scenarios that will place learners in realistic projects in which they will interact with Virtual Agents of different cultures that will simulate real members.

Keywords-Global Software Development, Simulators, Educational Environment, Technology for Training, Software Engineering Education

I. INTRODUCTION

Global Software Development (GSD) is an emerging paradigm of the Software Engineering whose main objective is to optimize resources, to increase the market area and to decrease development costs by distributing the software development across zones in which a skilled workforce is more readily available [1]. However, apart from these advantages, a number of problems, caused mainly by distance, must be confronted, such as those relating to communications, group awareness, knowledge management, coordination, collaboration and project management [2].

In distributed environments, communications are often based on the use of groupware tools such as: e-mail, video conference, wiki and instant messaging. And typical collaborative problems are related to poor communication, high response delays, lack of face-to-face contact, lack of understanding and response delays [3], which can cause frustration and a lack of motivation and interest.

Time and cultural differences may also cause problems in the attainment of a common understanding between participants, a lack of trust and inefficiency when attempting to resolve conflicts [4]. Stakeholders should become accustomed to dealing with people who have different customs, languages, beliefs, skills and even ways of interacting [5].

Because of this engineers should be trained to deal with the new challenges caused by globalization. However, recently graduated professionals generally lack the skills needed since they have seldom been involved in real projects [6], in which they must be able to communicate effectively with globally distributed members using a common terminology and language [7].

Some global team working skills can be acquired while working on real projects. Others, however, may need to be explicitly taught in educational environments, where the impact of mistakes is not critical [8]. Training these skills is not easy, since it necessitates providing learners with real experiences that will allows them to develop both technical and non-technical skills [9]. It is therefore essential to develop teaching strategies that will permit an active and collaborative learning, with which students can learn by doing [10] and develop skills required in the use of communication tools in order to confront language and cultural diversity problems [11].

Many strategies found in literature propose involving students in projects developed in various universities [12], in which students play a role in the team in order to solve a GSD scenario. However, replaying the complexity of real teams in educational environments is difficult and requires a great deal of coordination between teams from different universities [6]. Companies also deal with similar problems, as customers are not always willing to confront the risks of involving students in real projects.

In this work, we present a simulator based on software agents that will place learners in virtual GSD scenarios specifically designed to improve cultural understanding capabilities and personal relationships, along with improving the performance and quality of communication between the stakeholders.

Initially, we focus on the requirements elicitation stage, by placing learners in virtual elicitation meetings in which students play the role of interviewer. Customers are represented with Virtual Agents which will textually answer the learner's questions, simulating that they are from a specific culture.

The paper is organized as follows: Section 2 describes the strategies found in literature dealing with the teaching and training of GSD. Section 3 presents the aim and characteristics of our teaching simulator. The main applications of this tool are described in Section 4. In Section 5 we describe the manner in which our simulator addresses language and cultural differences. The fundamental aspects of the training scenarios are explained in Section 6, and Section 7 presents a training scenario oriented towards the requirements elicitation phase. We close with some conclusions and remarks concerning future improvements to our approach.

II. Related Work

We carried out a systematic literature review which has led to the discovery of various strategies that address the teaching and training of students and software engineers in GSD.

At a higher level we also found **academic courses** [13], [14], some of which replicated GSD environments in virtual settings [15]. [16] proposes a framework oriented towards offshoring for defining and evaluating learning objectives in three key areas: remote communication, knowledge management, and project and process management, showing a case-study of an off-shore requirements engineering class experience.

With regard to university courses, we also found several efforts concerning collaboration between different universities to perform certain activities or collaboratively develop software [17], [18], [19], [20], [21]. This type of environments facilitates the student's familiarization with cultural and time differences and allows them to train in the use of communication tools and coordination and collaboration activities [9].

However, these approaches usually involve schedule issues, since universities do not always share the same timetable or holidays. As a consequence it is difficult to attain an appropriate level of coordination and collaboration with the different institutions [22]. Communication and interaction problems between students and a lack of motivation also appear [23].

On the other hand, several **learning environment** approaches also exist [3]. A representative example is the collaborative learning platform CURE [23], which is based on the metaphor of virtual rooms (virtual places for collaboration). Those users who are in the same room can communicate by using the communication channels available. Each department of the university can administrate its own courses in public virtual rooms. Students' work is logged and teachers are provided with daily reports to monitor their activities.

iBistro [24] is a proposal based on the 'learning by doing' approach to address issues related to informal communications in distributed project courses. iBistro is an augmented meeting space that enables virtual teams to collaborate in distributed projects, and focuses on improving the students' project management and programming and social skills by allowing them to capture, structure and retrieve knowledge from informal conversations. iBistro also enables participants to improve the content and the structure of meeting minutes by using an asynchronous collaborative tool, and furthermore captures the audio, video, notes and drawings from the meetings that will assist in the elaboration of the minutes.

We have also found tools oriented towards training skills, such as those related to communications and content management, as is the case of Jazz [25], a synchronous, collaborative development platform that integrates support for team development, providing version control, chat, and work item features.

Moreover, e-Learning approaches also appear, such as OAS!S [26], which allows discussion boards, mail systems, chat and content management, which can be very helpful, mainly in improving communicative skills.

The application of these approaches present common problems such as the problem of working with distant members that can miss some meetings or fail to comply their tasks or milestones, affecting the whole team [27]. Therefore their application is not always easy. Furthermore, it is difficult to adequately cover any phase of GSD.

Finally, the **teaching of GSD in enterprises** has not been commonly reported. The most remarkable example was found in [28], which presents the experience of a multinational organization that trains its engineers in GSD activities. The study presents an evaluation of the company's global training program, which offers the same training courses to all employees, and proposes training strategies after analyzing the results with the aim of reducing cultural differences. On the other hand [29] describes an experience of training a virtual team of developers and testers in testing activities, in which cultural challenges appeared.

However, putting these proposals into practice is difficult, since companies are not always willing to invest time and resources in these training programs, which might prove to be extremely risky in real scenarios.

We have not yet found any relevant studies dealing with efforts made in the creation of an environment to simulate GSD scenarios such as that introduced in this paper.

A. Virtual Agents in Education

Since our training tool uses Virtual Agents (VAs), in following paragraphs we also describe other related works which, despite not being framed in GSD, nevertheless deal with language and cultural differences as in our case.

VECTOR [30] is a virtual training environment oriented towards the teaching of cultural differences by means of VAs that represent the members of the other culture. Cultural rules are encoded and mapped onto the specific scenario for each virtual agent, along with its behavior and its emotional model. VAs will react to trainee dialog and actions in order to show and measure cultural rule violations.

[31] describes a training environment based on VAs in which the student, taking on the role of a soldier, learns elements of the Iraqi dialect, along with the nonverbal language, culture, and customs that would be most essential to military personnel.

We have also found studies focused on language learning [32] that can be used to improve students' communicative

skills and cultural awareness. One of these approaches is the SPELL system [33], which uses embodied VAs to create scenarios in which learners can converse in the target language by employing speech recognition.

However, these tools are not oriented to tackle the problems and situations that can appear in GSD scenarios, in which software engineers interact with multicultural and multidisciplinary members for performing a wide set of activities during the software life cycle. Moreover, this interaction frequently requires carrying out certain activities such as the fulfillment of documents or the source code generation that are not covered by the aforementioned approaches.

B. *Needed Skills*

From another systematic review we also studied what skills a person who work in GSD should have and we found the following skills:

- Knowledge of a common language, communication protocols and interlocutors' customs [9], [19].
- Use of typical GSD tools such as knowledge and document management and control version tools [34], [15]. Knowledge of methods, data, and processes required in a distributed project [28].
- Leadership and knowledge of negotiation skills [6], [35].
- Common teamwork skills [14].
- Conflict resolution when dealing with people with different personalities [27].
- Ability to communicate with a multidisciplinary team using a common terminology and language [3], [7].
- Ability to manage ambiguity and uncertainty [9].
- Experience in computer-mediated communications [27], [9].
- Skills to gain the team's confidence and trust though informal communication [7], [36].

III. SIMULATING THE GSD ENVIRONMENT

With the aim of immersing learners in realistic training scenarios, we designed a framework that simulates GSD environments and trains learners in different scenarios by confronting cultural and language differences. In this way we can simulate a wide variety of problems minimizing the problems of dealing with real projects and partners.

The simulator employs Virtual Agents (VAs), which are useful in helping the learners through the use of textual dialogues and demonstrations of affective reasoning. The displaying of emotions and personality allow VAs to play an important role in the learners' motivation [37] since they appear to care about the students' actions and can express enthusiasm about the tasks to be performed [38]. VAs can emulate different emotions, such as anger, anxiety, annoyance, nervousness, distress, excitement, enthusiasm, happiness and disgust in order to encompass different types of scenarios, and this permits more realistic simulations.

On the one hand, a VA plays the role of colleague or team mate [39] which will help the learner to cope with the scenario and to perform their task. On the other hand, other VAs will be able to play any role in a GSD scenario, such as customer, requirements analyst, developer, project manager, etc., thus allowing the learner to be placed in a wide range of realistic scenarios.

A. *Architecture*

Our simulator provides a set of predefined GSD scenarios oriented towards different problems and stages of the software development, in which learners will chat with VAs in order to solve a problem or perform certain typical GSD tasks. The environment is based on a client-server architecture (shown in Figure 1). Both the instructors and learners have an interface with which they can access the services provided by the central server.

The instructors' interface allows them to plan the simulation by editing existing training scenarios or creating new ones, as well as defining new VAs with specific cultures and personalities. They can also manage learners' information and tasks, correct exercises or validate automatic corrections and examine learners' actions.

The learners' interface allows them to log into the system and execute the scenarios that the instructor has scheduled for them. Firstly, the scenario will be presented to the learners and then they will have a simulated interview or meeting with one or more VAs. Another VA plays the role of colleague and guides the user, correcting some of the learners' interventions or telling them what the following steps are, providing rationales. When the learners have completed the scenario, they may have to complete some documents, pass an exam or fill in a questionnaire that can be automatically evaluated.

The server side provides all the services required by the clients and will manage the interaction with learners and instructors by accessing the information stored in the database server.

The database server manages all the information required by the system and contains the following information the GSD scenarios with all the information required for the execution of any scenario, including the VAs involved, its definition and specific cultural and language rules and best practices for that scenario along with the conversational knowledge needed.

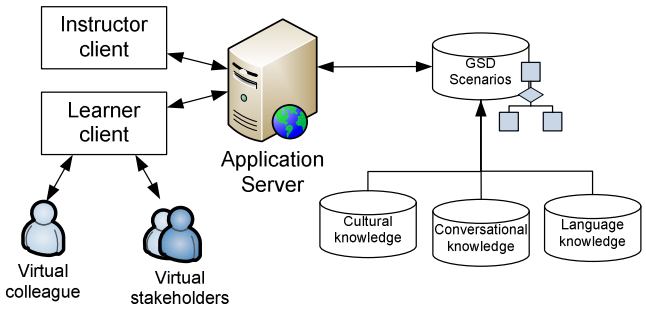

Figure 1. Simulator environment

The system also records and evaluates any questions that the learner has formulated inadequately with regard to cultural differences and communication protocol, and allows document templates (e.g. requirements documents, minutes

from meetings, etc.) to be filled in by following specific templates.

B. Virtual Agents

The VAs (shown in Figure 2) interact with learners by using a chatbot system which will answer the learners' questions in relation to a GSD scenario, using natural language in an attempt to imitate a human being. The VAs simulate stakeholders of different nationalities in order to help the learners to know the problems caused by cultural and language differences, communication difficulties, knowledge management and trust.

Various VAs with different characteristics (appearance, culture, gestures and voice) could be associated with an scenario, providing a rich environment for team training. The implementation of VAs is based on ALICE [40], a popular chatbot system based on the recognition of text patterns that utilizes the Artificial Intelligence Markup Language (AIML), which contains the knowledge related to the conversation in XML format. ALICE architecture has two basic modules to implement a conversational agent; an AIML interpreter (in our case located in the *learner interface* in Figure 1) and an AIML store (specific to each scenario).

The VA thus interacts with learners by using their *Conversational knowledge* of the problem domain which is helped by a text-to-speech engine which also motivates them by employing typical gestures and facial working expressions defined in the *Agents profile* in order to show emotions that are synchronized with the AIML conversational knowledge using customized tags.

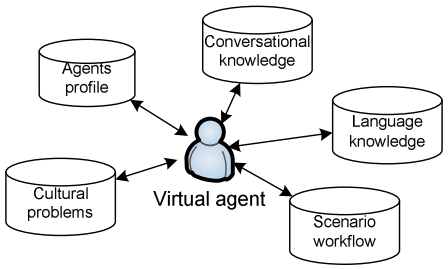

Figure 2. Virtual agents' architecture

The Virtual Colleague (VC) guides the learner by concentrating on the *Scenario workflow* (explained in Section 6) in order to follow a logical sequence during the conversation, and it will correct the learner when s/he does not make an appropriate interaction by concentrating on *Cultural problems* and *Language knowledge* (detailed in Section 5) . The VC detects cultural problems and an inappropriate use of language by using recognition patterns, and proposes a more correct manner in which to construct the sentence for that culture.

IV. Applications of the Simulator

The architecture of our simulator is open to the design of training scenarios oriented towards several types of collaborative activities. However, each stage of GSD must be dealt with in depth, so we shall first focus our research on Requirements Elicitation since this *is one of the most communication-intensive activities in the software lifecycle*, although in the future we also hope to deal with the following stages of GSD (in accordance with ISO/IEC 12207):

- Software requirements analysis
- Software design
- Software construction
- Software testing
- Documentation
- Joint review
- Project management

However, we also hope to apply the simulator to any type of meeting that might occur during the project lifecycle, e.g. tracking meetings, negotiation meetings, project initiation meetings, etc., and encourage the learner to fill in the meeting minutes.

Meeting minutes are vital to GSD teams, as the written language is easier to understand than oral communication, and a document can always be reviewed by both parties in order to discover misunderstandings and to ensure a common comprehension [41]. It is very important to encourage non-native speakers to be aware of misunderstandings and train them in initiating clarification dialogues, and rephrasing statements in a correct manner [41], and this requires a mutual trust between participants.

V. Training in Language and Cultural Differences

The role of cultural differences has been widely studied in both GSD [42], [43] and collaborative online learning [44] contexts. Through our proposal we attempt to create scenarios that will train learners in cultural differences. This requires a tremendous knowledge of the problems in this field which will allow us to develop realistic situations and appropriately characterize the VAs. To achieve this it is necessary to study the differences between the cultures and languages implied in a scenario.

Existing literature based on the Hall [45] and Hosfstede [46] cultural dimensions has been used to create a knowledge database containing a corpora of best practices for the cultural pairs that the VAs in our simulator will take into account. For example, Spanish people are often very direct in the way in which they express themselves, and this may appear rude to people from other countries who are accustomed to a more indirect manner.

We shall also considered **cultural issues** such as:

- The use of titles
- presentations and greetings
- starting and finishing a conversation
- negotiating and resolving disputes
- motivation and reward
- requests
- the resolution of conflicts

In relation to **language issues**, GSD communications frequently use English as a lingua franca [41]. Non-native speakers will confront communicative problems, which are closely connected with intercultural issues. Some typical

written linguistic problems with English in global environments are [47].

- the incorrect use of "false friends"
- incorrect plural formations
- incorrect past tense and past participle formations
- avoidance of passive forms
- the absence of the third person –s
- the if-part of conditional clauses with would
- treating who and which as interchangeable relative pronouns
- the insertion of redundant prepositions
- the overuse of certain verbs of high semantic generality (e.g. do, have, make, put, take)

In order to address language differences, our simulator has a language knowledge base (shown in Figure 1) with the rules for all the language pairs that the VAs will use in order to correct learners' mistakes.

The conversational knowledge along with cultural and language rules can be customized to specific scenarios according to our scenario-based learning model presented in the following section.

VI. SCENARIO-BASED LEARNING

Our proposal is able to simulate complex scenarios, in which a deep and insightful conversation will take place. It must therefore avoid speech repetitions and conversations which do not make sense out of context.

In this respect, literature deals with useful approaches from the perspective of scenario-based learning (SBL) focused on improving synchronous and asynchronous communications [48]. SBL immerses learners in a story in which they have to respond in order to influence the outcome of that story in which the information is placed in a context and revealed in a linear fashion.

SBL systems prompt the learner to choose between several responses that will influence the execution of the scenario. In this way, the learner actively interacts with the scenario and learns cause and effect relationships and interpersonal skills, since s/he is responsible for guiding the story.

In SBL, the scenario is defined by a set of decision points that sends learners in different directions based on their responses or actions. Figure 3 illustrates an example of SBL in which there is both a preferred path, and an alternative one that is also acceptable but not optimal. In this example, the most optimal path is "2, 2.1, 2.1.1". Also are right other paths as "1, 1.2, 1.2.1" or "3, 2.1, 2.1.1".

If the learner chooses a wrong path in the scenario such as "1, 1.1, 1.1.1, 1.1.1.1" or "3, 3.1", the VC can help him/her by providing immediate feedback.

In our simulator, the scenarios start with an introduction to the problem to be solved and to the characters involved. Each phase is focused on a specific part of the conversation. For each phase, VAs only speak about a concrete subject, and the change to the following phase is made through events that are triggered by certain answers, comments or actions carried out by the learner.

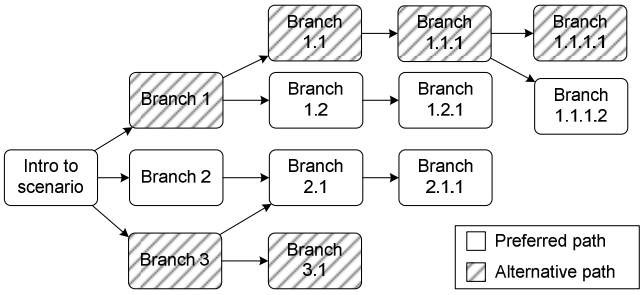

Figure 3. Example of scenario-based learning design

Instructors have a scenario designer (in the "instructor client", Figure 1) that will allow them to define a chart flow in which each phase of the scenario consists of:

- specific conversational knowledge for each VA, with the dialogue needed for that phase.
- events that will force the change to the following phase.
- specific cultural and language knowledge for that phase.
- grading for this phase, in order to evaluate the learner's actions.

Modeling the scenarios in this way serves to ease the design task and helps to structure the cultural and language problems according to the context of the conversation. This will also ease the management of the VAs' gestures and speech, and will increase the realism of the conversation, which will be more coherent and contextual.

Finally, the learners' decisions will be taken into account in the automatic evaluation of the scenario. Each phase of the scenario also has a grade, so the learner is assessed according to the phases chosen and successfully completed.

In our case the model such also employs hierarchy relationships that will allow conversations to be grouped in logical thematic areas. Each phase in the scenario have its own specific cultural and language knowledge and they can be used hierarchically according to the context of the conversation so that if a phase contains a sub-flow chart, this sub-flow chart can use the cultural and language knowledge of its parent.

VII. TOWARDS A REQUIREMENTS ELICITATION SCENARIO

The Requirements Elicitation (RE) stage is particularly affected by poor communication, difficulties in knowledge management and cultural and time differences [49], which result in ill-defined requirements and misinterpretations.

In a typical GSD scenario, analysts have to communicate with customers in order to obtain a detailed requirements specification document of the software. Mistakes during this stage may prove to be very expensive, and the communication of the requirements is a key element in distributed projects' success [41]. It is therefore necessary to apply adapted strategies in order to minimize these effects [50].

In this section we describe a scenario for our simulator designed to allow professionals to acquire the skills needed in Global Requirements Elicitation, in which users play the

role of interviewers who will elicit a set of functional and non-functional requirements by carrying out interviews in natural language using chat. The users will eventually solve the scenarios by filling in a requirements document, helped by a virtual colleague. This document will be automatically evaluated to detect faults such as: ambiguous, incorrect and unspecified requirements.

A. Requirements Elicitation Scenario Example

In this section we present a scenario in which a Spanish learner, who plays the role of analyst, will interview a virtual English customer, by means of a chat in order to elicit a set of functional and non-functional requirements. In this case the scenario must begin by placing the learner in the context. The VC will explain the problem to be solved and the time limit, and will also present the virtual agent, explaining his role in the project and culture.

The learner must greet the VA in an appropriate manner according to its culture. S/he must then conduct the conversation according to the VC's guidelines and the VA will answer in order to obtain the customer's software requirements within the time limit. If necessary, the learner could prepare the interview before the scenario begins.

Real RE interviews usually require more than one meeting, so the system could also have several scenarios dealing with the same subject, but for different proposals.

B. Designing the Scenario

The instructors are able to design adapted scenarios by selecting the VAs available and using a flowchart editor that will allow them to generate SBL models. Based on the goals of the scenario, the instructors will identify phases and decision points to design the SBL model, and will associate the corresponding information (conversational knowledge, cultural and language knowledge, etc.) for each phase.

In Figure 4, we show a flow chart for an RE scenario that starts with an introduction to the context of the problem and continues with the interview through which system requirements, non-functional requirements, functional requirements and storage requirements will be elicited.

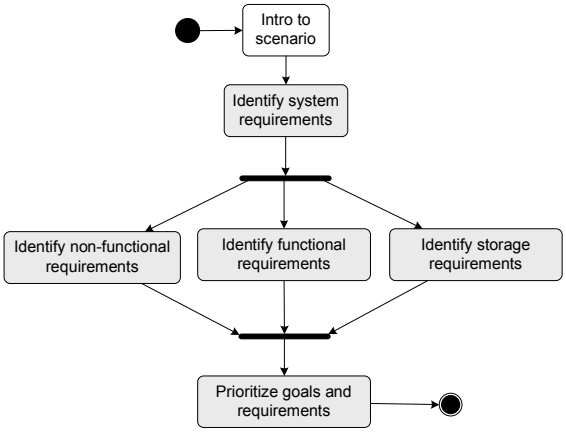

Figure 4. Example of RE flow chart

Each activity contains sub-flow charts that guide the scenario in the context of that phase. As an example, Figure 5 shows details of the "Identify storage requirements" phase. The phases contained in this diagram may also contain sub-flow charts or atomic phases. E.g. the "Determine relevant information" phase could be composed of sub-phases grouping the different types of information required.

The instructor can assign the new scenario to a group of learners, who will log into the system using the "learner client" and will initiate it. During the scenario execution, the learner will unconsciously determine the next phase of the flow chart by chatting to the VAs using natural language.

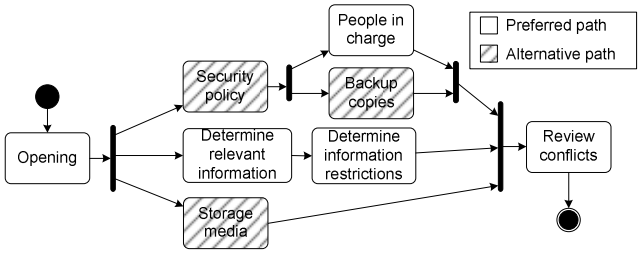

Figure 5. Detail of "Identify storage requirements" phase

Figure 6 shows the chat interface that will be available to the "learner client", with an example of a dialog for the "security policy" phase shown in Figure 5. The SBL model will prevent the learner from returning to a previous conversation.

The characters were provided by Haptek Incorporated, and include gestures, text-to-speech and lip synchronization features.

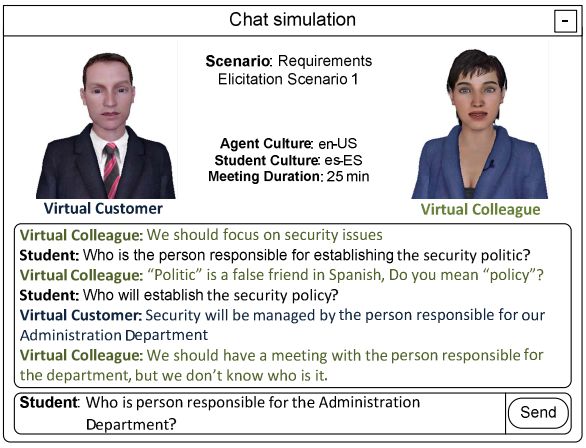

Figure 6. Chat scenario example

After the meeting the learner will fill in a requirements document using predefined templates for functional, non-functional and storage requirements. This information is automatically evaluated by the system according to a requirements document pattern, and could also be reviewed by the instructor.

VIII. Conclusions and Future Work

In this work we have proposed a training model based on the use of a simulator that introduces learners to GSD activities, thus diminishing the risks of involving non-qualified engineers in real projects and minimizing the costs and need for adherence to schedules of distant members. In this phase we have focused our research on RE, concentrating our efforts on requirements meetings, although in the future we hope to deal with other stages of GSD in which language and cultural problems appear during the interaction among virtual teams.

We have presented an RE scenario which we are now improving by studying language and cultural differences, which will provide us with an autonomous and complete training course. Our system will, therefore, be self-sufficient and will allow learners to train in GSD skills at any moment without depending on the availability of other partners or colleagues, and the instructors' interaction with the system will be reduced to assigning scenarios and validating grades.

However, our approach also allows instructors to customize their own scenarios by adapting them to the degree of its learning, and users will be able to simulate meetings without depending on schedules that would require real training environments.

In the future we intend to improve several aspects of the simulator in order to:

- Involve various VAs in the same scenario.
- Involve various learners in the scenario. Previous to the meeting simulation, the learners should prepare a role in the scenario in order to collaboratively solve a problem by using chat or email, paying attention to the VC's guidelines.
- Facilitate documents through sharing between team members.
- Allow the edition of UML diagrams that the VC could understand and correct.

Finally we plan to focus on other phases of GSD, apart from requirements engineering, and study the cultural and language problems that could affect other GSD activities with the aim of developing new training scenarios, also allowing learners to play different roles in the project to make them aware of the different kind of problems.

Acknowledgments

This work has been funded by the PEGASO/MAGO project (Ministerio de Ciencia e Innovación MICINN and Fondos FEDER, TIN2009-13718-C02-01). It is also supported by MEVALHE (HITO-09-126) and ENGLOBAS (PII2I09-0147-8235), funded by Consejería de Educación y Ciencia (Junta de Comunidades de Castilla-La Mancha), and co-funded by Fondos FEDER, as well as MELISA (PAC08-0142-3315), Junta de Comunidades de Castilla-La Mancha, Consejería de Educación y Ciencia, in Spain.

References

[1] J. D. Herbsleb, "Global Software Engineering: The Future of Socio-technical Coordination," presented at the Future of Software Engineering (FOSE 2007), 2007.

[2] M. Jiménez, *et al.*, "Challenges and Improvements in Distributed Software Development: A Systematic Review," *Advances in Software Engineering,* vol. 2009, p. 14, 2009.

[3] L. J. Burnell, *et al.*, "Teaching Distributed Multidisciplinary Software Development," *IEEE Softw.,* vol. 19, pp. 86-93, 2002.

[4] D. Damian and D. Zowghi, "Requirements Engineering challenges in multi-site software development organisations," *Requirements Engineering,* vol. 8, pp. 149-160, 2003.

[5] P. T. Nguyen, *et al.*, "Critical factors in establishing and maintaining trust in software outsourcing relationships," presented at the Proceedings of the 28th international conference on Software engineering, Shanghai, China, 2006.

[6] D. Damian, *et al.*, "Instructional design and assessment strategies for teaching global software development: A framework," presented at the Proceedings of the 28th international conference on Software engineering, Shanghai, China, 2006.

[7] S. Toyoda, *et al.*, "A Case Study on Project-Management Training-Support Tools for Japanese/Chinese/Indian Offshore Development Engineers," in *Knowledge-Based Intelligent Information and Engineering Systems*. vol. 4693, S. Berlin, Ed., ed Heidelberg, 2009, pp. 1222-1229.

[8] O. Gotel, *et al.*, "Introducing Global Supply Chains into Software Engineering Education," in *Software Engineering Approaches for Offshore and Outsourced Development*. vol. 4716, S. Berlin, Ed., ed Heidelberg, 2007, pp. 44-58.

[9] I. Richardson, *et al.*, "Globalizing Software Development in the Local Classroom," presented at the Proceedings of the 20th Conference on Software Engineering Education & Training, 2007.

[10] D. Rosca, "An active/collaborative approach in teaching requirements engineering," presented at the Proceedings of the 30th Annual Frontiers in Education, 2000.

[11] D. Hung and D.-T. Chen, "A Proposed Framework for the Design of a CMC Learning Environment: Facilitating the Emergence of Authenticity," in *Educational Media International*. vol. 40, ed: Routledge, 2003, pp. 7-14.

[12] R. B. J. Vaughn and J. Carver, "Position Paper: The Importance of Experience with Industry in Software Engineering Education," presented at the Proceedings of the 19th Conference on Software Engineering Education and Training Workshops, 2006.

[13] C. Murphy, *et al.*, "A distance learning approach to teaching eXtreme programming," *SIGCSE Bull.,* vol. 40, pp. 199-203, 2008.

[14] J. Favela and F. Peña-Mora, "An Experience in Collaborative Software Engineering Education," *IEEE Softw.,* vol. 18, pp. 47-53, 2001.

[15] M. Adya, *et al.*, "Bringing global sourcing into the classroom: experiential learning via software development project," presented at the Proceedings of the 2007 ACM SIGMIS CPR conference on Computer personnel research: The global information technology workforce, St. Louis, Missouri, USA, 2007.

[16] K. Berkling, *et al.*, "Offshore Software Development: Transferring Research Findings into the Classroom," in *Software Engineering Approaches for Offshore and Outsourced Development*. vol. 4716, S. Berlin, Ed., ed Heidelberg, 2007, pp. 1-18.

[17] D. Petkovic, *et al.*, "Assessment and comparison of local and global SW engineering practices in a classroom setting," presented at the Proceedings of the 13th annual conference on Innovation and technology in computer science education, Madrid, Spain, 2008.

[18] A. Rusu, *et al.*, "Academia-academia-industry collaborations on software engineering projects using local-remote teams," presented at the Proceedings of the 40th ACM technical symposium on Computer science education, Chattanooga, TN, USA, 2009.

[19] O. Gotel, *et al.*, "Working Across Borders: Overcoming Culturally-Based Technology Challenges in Student Global Software Development," presented at the Proceedings of the 2008 21st Conference on Software Engineering Education and Training, 2008.

[20] B. Bmegge, *et al.*, "Transatlantic project courses in a university environment," presented at the Proceedings of the Seventh Asia-Pacific Software Engineering Conference, 2000.

[21] P. Lago, *et al.*, "Towards a European Master Programme on Global Software Engineering," presented at the Proceedings of the 20th Conference on Software Engineering Education & Training, 2007.

[22] U. Bellur, "An Academic Perspective on Globalization in the Software Industry," presented at the Proceedings of the 30th Annual International Computer Software and Applications Conference, 2006.

[23] T. Schümmer, *et al.*, "Teaching distributed software development with the project method," presented at the Proceedings of th 2005 conference on Computer support for collaborative learning: learning 2005: the next 10 years!, Taipei, Taiwan, 2005.

[24] A. Braun, *et al.*, "iBistro: A Learning Environment for Knowledge Construction in Distributed Software Engineering Courses," presented at the Proceedings of the Ninth Asia-Pacific Software Engineering Conference, 2002.

[25] A. Meneely and L. Williams, "On preparing students for distributed software development with a synchronous, collaborative development platform," presented at the Proceedings of the 40th ACM technical symposium on Computer science education, Chattanooga, TN, USA, 2009.

[26] K. Swigger, *et al.*, "Structural factors that affect global software development learning team performance," presented at the Proceedings of the special interest group on management information system's 47th annual conference on Computer personnel research, Limerick, Ireland, 2009.

[27] R. Ocker, *et al.*, "Training Students to Work Effectively in Partially Distributed Teams," *Trans. Comput. Educ.,* vol. 9, pp. 1-24, 2009.

[28] R. Prikladnicki and L. Pilatti, "Improving Contextual Skills in Global Software Engineering: A Corporate Training Experience," presented at the IEEE International Conference on Global Software Engineering (ICGSE'08), Bangalore, India, 2008.

[29] A. B. Bondi and J. P. Ros, "Experience with Training a Remotely Located Performance Test Team in a Quasi-agile Global Environment," presented at the Proceedings of the 2009 Fourth IEEE International Conference on Global Software Engineering, 2009.

[30] E. Deaton, *et al.*, "Virtual environment cultural training for operational readiness (VECTOR)," *Virtual Real.,* vol. 8, pp. 156-167, 2005.

[31] E. M. Sims, "Reusable, lifelike virtual humans for mentoring and role-playing," *Comput. Educ.,* vol. 49, pp. 75-92, 2007.

[32] R. Eliseo, *et al.*, "The role of animated pedagogical agents in scenario-based language e-learning: a case-study," in *Proceedings of the International Conference of "Interactive computer aided learning" ICL2007*, Villach, Austria, 2007, pp. 1-7.

[33] J. N. Anderson, *et al.*, "Language learning with interactive virtual agent scenarios and speech recognition: Lessons learned," *Comput. Animat. Virtual Worlds,* vol. 19, pp. 605-619, 2008.

[34] A. Braun, *et al.*, "iBistro: A Learning Environment for Knowledge Construction in Distributed Software Engineering Courses," presented at the Proceedings of the Ninth Asia-Pacific Software Engineering Conference, 2002.

[35] B. Lutz, "Training for Global Software Development in an International "Learning Network"," presented at the Proceedings of the International Conference on Global Software Engineering, 2007.

[36] I. Richardson, *et al.*, "Distributed development: an education perspective on the global studio project," presented at the Proceedings of the 28th international conference on Software engineering, Shanghai, China, 2006.

[37] C. Elliott, *et al.*, "Lifelike pedagogical agents and a ective computing: An exploratory synthesis," in *Artificial Intelligence Today*. vol. 1600, ed Berlin: Springer Verlag, 1999, pp. 195-212.

[38] S. A. Murphy and M. J. Hine, "The Role of Motivational Systems and Emotions in a Virtual Task," presented at the Proceedings of the 42nd Hawaii International Conference on System Sciences, Big Island, HI, USA, 2009.

[39] C.-Y. Chou, *et al.*, "Redefining the learning companion: the past, present, and future of educational agents," *Comput. Educ.,* vol. 40, pp. 255-269, 2003.

[40] R. S. Wallace, "The Anatomy of A.L.I.C.E," in *Parsing the Turing Test*, S. Netherlands, Ed., ed, 2008, pp. 181-210.

[41] B. Lutz, "Linguistic Challenges in Global Software Development: Lessons Learned in an International SW Development Division," presented at the Fourth IEEE International Conference on Global Software Engineering (ICGSE'09), Limerick, Ireland, 2009.

[42] K. Swigger, *et al.*, "Effects of culture on computer-supported international collaborations," *International Journal of Human-Computer Studies,* vol. 60, pp. 365-380, 2004.

[43] P. Kruchten, "Analyzing intercultural factors affecting global software development - a position paper," in *3rd International Workshop on Global Software Development (GSD2004)*, Edinburgh, Scotland, 2004, pp. 59-62.

[44] R. Vatrapu and D. Suthers, "Culture and Computers: A Review of the Concept of Culture and Implications for Intercultural Collaborative Online Learning," ed, 2007, pp. 260-275.

[45] E. T. Hall, *Beyond Culture*: Anchor Press, 1976.

[46] G. Hofstede and G. J. Hofstede, *Cultures and organizations: software of the mind*, 2nd ed. New York, NY, USA, 2005.

[47] J. Grzega, "Reflection on Concepts of English for Europe: British English, American English, Euro-English, Global English," *Journal for EuroLinguistiX 2,* pp. 44-64, 2005.

[48] M. A. Siegel, *et al.*, "Designing for Deep Conversation in a Scenarios-Based e-Learning Environment," presented at the Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 4 - Volume 4, 2004.

[49] D. E. Damian and D. Zowghi, "The Impact of Stakeholders? Geographical Distribution on Managing Requirements in a Multi-Site Organization," presented at the Proceedings of the 10th Anniversary IEEE Joint International Conference on Requirements Engineering, 2002.

[50] G. Aranda, *et al.*, "A Requirement Elicitation Methodology for Global Software Development Teams," in *Encyclopedia of Information Science and Technology, Second Edition*, M. Khosrow-Pour, Ed., ed: IGI Global, 2008.