A tool for Training Students and Engineers in Global Software Development Practices

Miguel J. Monasor[1], Aurora Vizcaíno[2], Mario Piattini[2],

[1] University of Castilla-La Mancha,
Campus Universitario s/n,
0207, Albacete, Spain
MiguelJ.Monasor@gmail.com
[2] 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. Global Software Development (GSD) is an emerging trend in which virtual teams work on the same projects at a distance. Despite the advantages of this shift, the collaboration between distant members becomes more difficult. Team members interact by using collaborative tools, and this collaboration is affected by time, cultural and language differences. These drawbacks lead to the need to train students and software engineers in the new collaborative skills required.
These skills can only be trained by involving learners in practical experiences, but this is not always possible since it necessitates collaboration with distant institutions (universities/firms). We have focused our work on the development of a tool with which to train these skills through the use of a virtual training environment for GSD that avoids this difficulty by placing learners in virtual GSD scenarios in which they will develop the skills needed to work on global software projects.

Keywords: Global Software Development, Engineering Education.

1 Introduction

Collaboration between virtual teams is one of the main challenges of Global Software Development (GSD) [1]. This relatively recent trend allows team members to work on the same projects in different countries by interacting through communication and collaboration tools. Face-to-face contact is not possible in these scenarios, and although GSD provides benefits such as the lower cost, the higher availability of skilled workforce or the broader area of commercialization, it also entails new problems mainly derived from distance. These problems particularly affect team members' communication, which becomes more complex, especially with the appearance of cultural and language differences [2] and time zone differences [3].

The use of a non-native language in communications creates additional drawbacks, as interlocutors are not always able to express their ideas, and the presence of different cultures, terminology and languages often cause misunderstandings and a lack of trust [2], [4]. Cultural differences may also cause problems related to legal issues and knowledge-transfer [3].

These problems are present in the different stages of the software life cycle in different ways, and affect different collaborative tools and processes.

Moreover, software factories managers often complain about the poor skill level of team members in the use of communication tools which leads to delays, a lack of trust and misunderstandings [5]. However, finding solutions to these difficulties is not easy and traditional software engineering education does not deal with these topics.

Software Engineering Education must, therefore, be focused on training students and software engineers in the problems that GSD entails [6]. However, the training of cultural differences and communication and collaboration difficulties requires practice. Since these skills are better learned by doing, the challenge consists of integrating theory into practice rather than simply learning theoretical concepts.

Many current proposals in educational environments confront this issue by coordinating practical experiences with distant learners from different cultures and languages. This entails complex problems for the instructors who must coordinate their efforts with distant institutions. Students also have problems, as they usually encounter scheduling problems when interacting with other distant learners.

The training of GSD activities requires new theoretical contents and training methods in order to avoid the great deal of coordination with distant members and institutions that is implied.

One solution that we propose to this problem is the use of a virtual training environment that can simulate realistic GSD scenarios in which learners are introduced into the context of a problem that they will solve by interacting with Virtual Agents (VAs). This interaction takes into account traditional communication tools (instant messaging and e-mail) and deals with cultural and language differences since VAs of different countries play a specific role in the scenario.

The textual interaction with VAs will allow learners to solve common project issues related to collaboration with multicultural and multidisciplinary members with regard to communication problems, information sharing and documentation.

This paper is organized as follows: Section 2 explains the related work along with the tools proposed and the skills required in GSD according to a systematic review carried out previously. Section 3 describes the virtual learning environment developed. In Section 4 we discuss the results obtained. Finally, Section 5 provides some concluding remarks and outlines our future work.

2 Related Work

As an initial step in our research, we carried out a study in order to discover the main skills needed by software engineers in GSD. We also performed a Systematic Literature Review that allowed us to discover the main strategies and proposals

reported in the field of GSD training and education, along with the main tools applied in this area. The results of our study are summarized below.

2.1 Skills required in GSD

Both students and software engineers must acquire specific skills that will allow them to carry out an effective development in order to confront the problems of globalization which are not part of their conventional education. The skill most commonly reported in literature is that of the use of computer-mediated communications [7], [8] since traditional face-to-face meetings are no longer common in GSD and the appropriate use of communication and collaboration tools is essential in these environments. Furthermore, members of different cultures take part in this interaction, signifying that members must know how to communicate effectively by using a common terminology and language, and by taking into account the different customs of the participants [9], [10].

Members must be familiar with both formal and informal means of communication. On the one hand, they must know how to write formal documents, contracts and emails in a common language [2], and on the other hand they must know how to interact by using the telephone or instant messaging services.

This interaction must be oriented towards gaining the team's trust [9], as this is one of the common problems when interacting with distant members. In order to achieve this, participants must be versed in the concepts of conflict resolution [7] and negotiation [2] which allow them to argue and minimize problems correctly during the interaction. They must also improve their improvisation skills [11], which are essential if fluent communication is to be achieved, and which also helps to improve the team's trust and teamwork skills [12]. It is therefore important for them to have experience in working with a multidisciplinary team [13], in which different degrees of knowledge and abilities are present during communication that may cause problems in reaching an understanding and comprehension difficulties. Learners therefore need to know how to manage the ambiguity and uncertainty that are present in GSD environments [8].

Finally, learners of education programs must know the traditional methods and processes used in distributed projects [5] and acquire realistic experiences in the use of traditional knowledge management tools, document management and version control systems [14], [15]. Only experience in realistic scenarios will allow them to develop leadership skills and learn how to effectively manage their time [16].

2.2 GSD education proposals

Literature deals with GSD education through different kinds of proposals which aim to train the skills required by globalization. We have classified these proposals by considering the following trends:

1. Traditional theoretical classes, adapted as a response to the needs for adjustments in software engineering education [17], [18]. These courses are focused on the collaborative processes, technical issues and cultural

dimensions of GSD in different areas of Software Engineering [19]. The courses are usually organized in collaboration with other distant universities [20]. However, coordination and collaboration difficulties with the different institutions [21] appear. Moreover, the participation of students with different skills and backgrounds must be considered [22].

2. Practical experience training courses, in which students put theory into practice and learn by solving typical problems that can be found in real environments. Learners interact with members of distant institutions by using email, telephone and instant messaging, and learn from their partners' skills and culture [11], [2] and by tackling processes with a close similarity to those applied in industry, including language, time and cultural problems [13] [8], [23], [10].

3. e-Learning approaches, which consist of web-based courses involving discussion boards, mail systems, chat and content management [24], [9]. Some of these use or adap WebCT, FirstClass OLAT or BlackBoard platforms, and are especially focused on improving communicative skills. We also discovered an approach in the context of collaboration that focused on artifact sharing [25].

4. Training courses in companies, which take advantage of their software engineers' real experience in order to apply the concept of a "learning network" [26] [16], in which experts in specific software development activities of the company, combine their training activities with their work as engineers. Learners can take advantage of real work experiences by maintaining contact with specialized professionals.
   Although this approach is less commonly reported in literature, [5] presents a training initiative in which a multinational organization provided a training course related to best practices of communication, trust, cultural differences and coordination.

5. Blended learning environments, applied in companies or universities which use learning platforms designed to support the development of real projects by using of collaborative tools similar to those used in real environments [14], [27], [28].

2.3 Training tools in GSD

One of the results of our systematic literature review was the discovery of a number of tools or environments oriented towards the training of GSD activities.

Genesis [29] is a collaborative environment that can be used in educational environments and which supports formal and informal communications and the definition, enactment and control of software processes using workflow. It also uses an artifact management system, called OSCAR [30] which supports collaborative software development via artifact sharing.

Jazz [28] is another collaborative platform based on Eclipse that supports functionalities such as: source code repository, chat, web interface, reports

generation, and work items. Students can use this platform to generate work items containing the relevant information related to any problem, along with the associated chat conversations.

In the same line, XPairtise [31] supports distributed pair programming practices providing coding and testing functionalities. It also supports communications through chat, shared whiteboards and a graphical Shared Editor in which pairs can cooperate by sharing their ideas. Using this Eclipse plugin, inexperts engineers can invite experts to a pair programming session who will help them to solve certain problems. The role of the "spectator" is also supported, who can watch the interaction among the pair and participate in the session chat, which can be used to teach a group of learners in a specific problem domain.

iBistro [14] is also based on the 'learning by doing' approach and is an environment that can be used to learn project management, software development and social skills. iBistro enables distributed members to collaborate during the software development and addresses miscommunications and information problems in informal meetings. This is achieved by allowing students to capture structures and retrieve knowledge from the meetings by using the audio, video, sketches, notes and the drawings generated. A minute generator tool stores the contextual information and allows the meetings to be represented and analyzed.

iBistro is oriented towards tacking some of the problems in GSD, and also provides intelligent support mechanisms such as that of computer supported group formation, and the ability to find stakeholders and experts in certain areas.

[32] presents a lab course based on the collaborative virtual learning environment CURE [33]. It basically uses virtual places for collaboration. These virtual places may contain pages with different contents, and communication channels such as chat, threaded mailbox, etc., and users, who can interact with other users, stand in the same virtual place.

In [34], the authors propose a framework oriented towards offshoring practices. This framework uses CodeBeamer, which is a collaborative platform that offers integrated support in project management, requirements management and code management and supports asynchronous communication through a wiki system.

In [27], the authors present a platform that integrates CURE and CodeBeamer, which allows students to develop a large software system by collaborating during all the phases of the software life cycle.

A Web-based collaborative platform is presented in [7], in which students can work with their partners in order to achieve the training module's scopes. This is done by using communication and content management tools, including a discussion board, a file sharing repository and a project calendar. Instructors can manage the training modules by defining instructions, milestones, and deliverables.

A further collaborative environment is presented in [35], and provides learners with a set of tools such as a chat, a scribble tool, an application sharing tool, graphics tools for designing UML documents, etc. A Web portal is used to allow students to manage the groups and projects in which they are involved, and to share their personal information. They can also access their partner's schedule and thus agree possible meeting times.

Finally, [12] proposes a framework for training learners in some of the difficulties involved in GSD which consists of tools for project scheduling and tracking, configuration management and for performing technical reviews.

However, these approaches do not completely satisfy the requirements of universities and companies as they create certain problems such as:

- a great deal of coordination with other institutions, which implies a high workload for the instructors
- dependency on other learners' availability
- time limitations
- difficulty in reproducing realistic scenarios
- high economic costs and infrastructure requirements
- high maintainability requirements

It is therefore necessary to offer instructors an appropriate environment and training materials in order to provide learners with realistic experiences which are adjusted to the reality of companies' current requirements [28] by avoiding the aforementioned problems.

3 Our Virtual Training Environment

The Virtual Training Environment presented here places learners in realistic virtual training scenarios in which they must interact with VAs in order to perform certain typical GSD activities. Since the interaction is carried out through VAs, learners can train in cultural and language differences at any time without depending on the availability of real partners.

Our environment helps instructors to manage learners and their activities and to maintain the training scenarios. Learners can access training scenarios, take part in virtual meetings with VAs and access artifacts for that scenario.

The Meetings Simulator allows learners to textually interact with VAs which will answer their questions in relation to a problem. We can thus simulate meetings with any kind of stakeholder involved in the project by defining new VAs, and learners can play any role in the project according to the design of the training scenario.

Our environment basically provides three main components:

- **Learners' interface**: this allows learners to communicate with instructors, manage their assigned tasks and execute simulated meetings through the meetings simulator. They can also submit deliverables, access documents or UML diagrams and answer tests associated with the scenario. A Website is therefore created for each learner in order to allow the deliverables for each scenario to be submitted.
- **Instructors' interface**: this allows instructors to assign tasks and monitor learners' actions. They can also organize teams and send notices and emails to individuals or groups. It also provides an editor which allows new training scenarios to be managed and created or existing ones to be modified. This editor permits instructors to define new VAs with specific cultures and personalities for their scenarios. We thus intend to minimize the instructors'

effort by providing mechanisms that enable easy customization and provide a wide set of training scenarios.

- **Central server**: Both, learners' and instructors' interfaces access the central server, which offers a set of services required by the interfaces and manages the required information stored in its database regarding learners and training GSD scenarios.

3.1 Definition of Training Scenarios

Our virtual environment works with training scenarios consisting of a set of schedules, documents, exams and virtual meetings. One or more VAs can take part in a virtual meeting, and can play a specific role in the GSD project (e.g. customer, requirements analyst, developer, project manager, etc.). A virtual meeting is also guided by a specific VA called a Virtual Colleague (VC) which has been designed to help learners during their training. This VA therefore plays the role of campaigner, which has been successfully used in other learning environments, as is reported in [36] and [37]. The VC will correct the learners' interventions by providing rationale and explaining the consequences of their actions, particularly with regard to cultural and language mistakes, but is also focused on GSD activities. The VC will also guide the learner towards following a logical sequence during the meeting.

3.1.1 Components of a Training Scenario

As is shown in Fig. 1, a training scenario is made up of one or more virtual meetings and a set of artifacts that constitute the "scenario content".

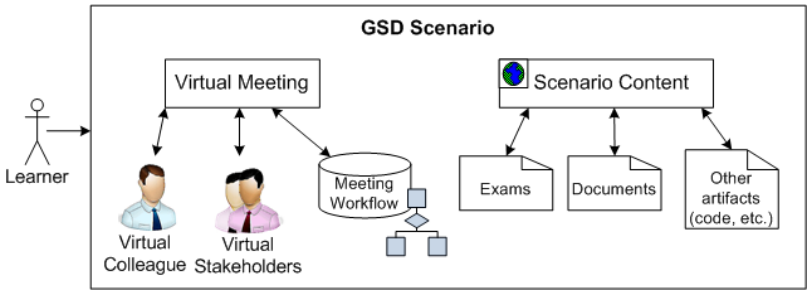

Fig 1. GSD scenario definition

Scenario Content: Each scenario will have specific documents that learners will need or will have to complete, such as a requirements specification report. A scenario also contains exams, tests, schedules, source code, etc.
Virtual meetings: in which VAs and a VC will participate in the conversation guided by the *Meeting Workflow* by following a logical sequence according to the learners' actions.

3.1.2 Meetings Workflow

The Meeting Workflows define the course of the conversation as a set of phases, each of which defines a small part of the conversation. Each phase is defined by a specific piece of conversational knowledge, context specific language and cultural knowledge which are used for that phase in the conversation.

The phases can also store information about their priorities, which can serve to evaluate the learners' actions and the correctness of their decisions. Finally, the phases also define any gesture that VAs could make as regards the context of the conversation. For example, a VA can emulate different emotions, such as anger, anxiety, annoyance, nervousness, distress, excitement, enthusiasm, happiness and disgust.

The phases are arranged by forming a sequential diagram that defines the Meeting Workflow in which the students will influence the execution path of the meeting as a result of their textual responses. Furthermore, these phases can be *simple* or *composed*. Composed phases contain other workflows with the aim of structuring the conversation with a high granularity level, and can contain information and conversational knowledge that is inherited from the phases contained in it.

This design of the virtual meetings avoids speech repetitions and out of context interventions, making it possible to simulate profound and insightful conversations in which the VC can provide immediate feedback depending on the context of the conversation.

The different parts of the conversational knowledge of the meeting are stored in the phases related to the context of the conversation. This knowledge is stored as XML text based on patterns that can be interpreted by a chatbot engine which is used by the VAs.

The instructor´s interface allows *Meeting Workflows* to be created and edited through an editor which permits the phases and conversational knowledge required to be introduced. Apart from the conversational knowledge, the instructor can also associate the cultural and language knowledge with the context that the VC will use to provide feedbacks to the learner.

3.2 Managing Cultural and Language Problems

We have effectively managed cultural problems by designing our virtual meetings on the basis of the existing literature of Hall [38] and Hosfstede [39], and by considering the specific problems for the cultures involved in the meeting. With regard to the language problems it is necessary to study the possible problems that could appear for the languages involved in the meeting for each scenario. For our first scenario we have considered problems related to the use of English as a lingua franca [40], [41], since this is the language usually used in GSD.

The phases of the Meeting Workflow contain information that the VC can use to detect inappropriate interventions by the learners. More specifically, a phase can contain:

- A list of expressions regarding cultural problems, which contains the appropriate and inappropriate use of titles, presentations and greetings, how to start and

finish conversations, requests, means of negotiation, etc. The following example shows how we correct a learner who does not use appropriate titles:

Cultural Problem:

Type: qualification

Pattern: "? Edwards" **Trigger**: "? <> Mr."

Definition: You should refer to Mr. Edwards by using his title (Mr.).

- List of expressions regarding language problems, such as of the overuse of certain verbs of a high semantic generality (do, have, make, put, etc.) or the use of false friends. The following example shows a pattern with which to correct the use of a false friend:

Language Problem:

Type: false friend

Pattern: "politic"

Definition:"Politic" is a false friend in Spanish. Do you mean policy?

- Rules regarding grammatical inaccuracies. Third party dictionaries and grammatical correctors for the target language are used for this purpose. These engines detect any mistake during the conversation and take into account typical mistakes. For example; a common mistake in the Spanish culture consists of changing the termination of a Spanish word in the hope that it will be correct in English. The VC will use these to report errors such as: avoidance of passive forms, incorrect plural formations, the absence of the third person, the use of redundant prepositions, etc.

Each of these entries also has an associated explanation that the VC will use to help the learner. These entries also have a score that will be used to evaluate the learner's interaction. This also serves to show statistics with regard to the number of mistakes made in each category at the end of the meeting.

The composed phases store cultural and language knowledge, signifying that any phase contained in it will use this information. We therefore avoid redundant cultural and language information in different phases, since general knowledge for this context can be stored in the parent phase.

4 Description of a Scenario

The definition of a training scenario requires a great deal of knowledge about the stage of GSD being dealt with, so it is first necessary to study the existing literature on this subject to discover the specific problems that may appear in each stage in order to design a representative training scenario.

The first training scenario we have developed is related to the Requirements Elicitation (RE) stage, since it is a highly communicative process, which is particularly affected by poor communication, and cultural and language differences that frequently causes problems related to ill-defined requirements and misinterpretations that may occur during interactions with customers and team members.

Our RE scenario is focused on the problems that Spanish learners confront when interacting using English. More specifically, the Meeting Workflow designed for this scenario places learners in a virtual meeting in which they will play the role of analysts in order to elicit a set of requirements for a virtual customer from the United States.

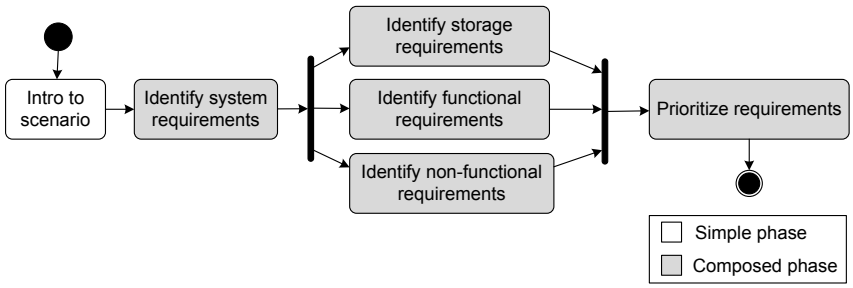

Fig 2. Example of Requirements Elicitation Workflow

Fig 3. Virtual meeting example

One of the Meeting Workflows proposed for this scenario is shown in Fig. 2, and starts with an introduction to the problem to be solved and to the VAs involved. This will be explained by the VC. After the introduction, the learners will actively interact with the VAs through the following phases of the meeting, which are organized by using composed phases that group the workflows and consider the type of requirements elicited (system requirements, functional requirements, non-functional requirements, and storage requirements).

Fig. 3 shows an example of a dialog between a Spanish learner and the virtual customer, in which the VC guides and corrects the learners' mistakes according to the aforementioned Meeting Workflow.

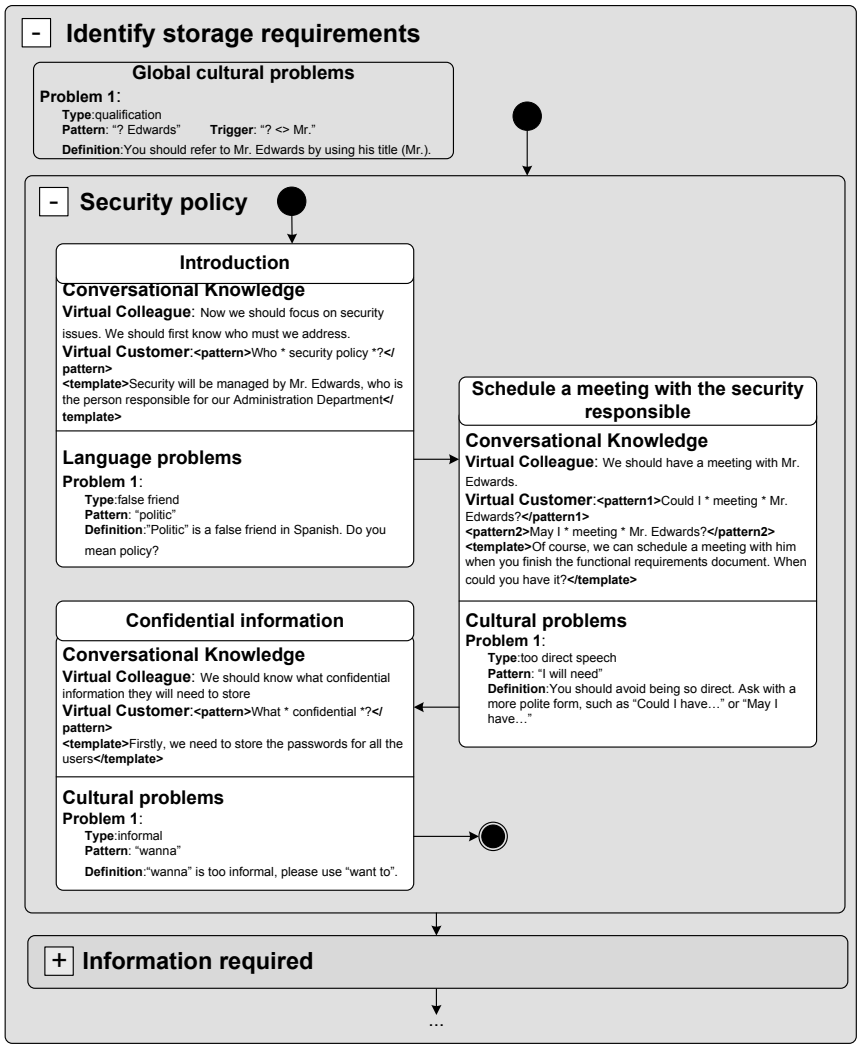

Fig 4. Virtual scenario definition

This is an exaggerated case in which the VC acts many times. However, when the learners' behavior is appropriate the VC's intervention is not necessary. The details of the Meeting Workflow phase with regard to this conversation are shown in Fig. 4, in which we have designed a composed phase (Identity storage requirements) which contains the definition of some of the cultural problems that learners may confront during this phase. For example, in this case it detects that the learner is referring to Mr. Edwards inappropriately. The first sub-phase (Security policy) in turn contains a sub-workflow in which we can see simple phases containing the conversational knowledge along with language and cultural problems specific to these phases.

For each learner intervention, our system will review the text introduced, checking the patterns defined for cultural and language problems. For example, in the "Introduction" phase, if the learner uses the word "politic", the VC will correct him/her by saying: ""Politic" is a false friend in Spanish. Do you mean policy?".

If any conflict were to exist between the cultural or language knowledge of a phase and its parent, the system would give priority to the information corresponding to the child phase, since it contains more specific information for that context than the parent. The scenario concludes when the learner completes all the virtual meetings associated with the scenario, finalizes the requirements elicitation document and fills in a questionnaire.

5 Discussion

Our proposal permits communication skills in GSD problems to be trained through typical communication and coordination channels and avoids the need for coordination with other institutions, thus reducing the instructors' workload and scheduling problems. It also avoids the difficulty of finding team members from different cultures with the appropriate skills and knowledge to carry out the GSD activities.

Learners do not depend on their partners' activities and can work at any moment without depending on another learner's availability. They can also play different roles in the projects, and thus become aware of the different kinds of problems from different perspectives.

VAs will provide learners with opportunities for self-reflection and self-correction by explaining the consequences and rationales of their actions with regard to team ethics and cultural differences. Instructors will also be able to provide the learners with feedback, since they can monitor the learners' activities and communicate with them.

Since it is not possible for instructors to have a profound knowledge of all the stages and problems of GSD [16], one of our future works will aim to provide a wide set of training scenarios oriented towards other stages of GSD such as software design, software construction or software testing in which different types of documentation can be managed. Fig. 5 shows the phases that we plan to implement in a complete training scenario along with the virtual meetings that could take place and the artifacts associated with each phase.

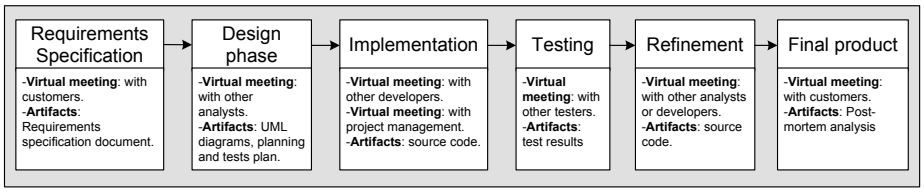

Fig 5. Phases of a complete training scenario

Instructors would thus be provided with realistic training scenarios that reproduce the complexity of GSD environments and, since our simulator permits the customization of the existing training scenarios, the instructors could adapt them to their specific needs.

6 Conclusions and Future Work

Training the collaboration skills required in GSD environments is a difficult task. In this paper we have presented an environment that simulates the complexity of real GSD projects by providing training scenarios that are especially focused on cultural and language differences.

Although our first training scenario is focused on the Requirements Engineering stage, we plan to develop further scenarios once we have completed the evaluation of this one. We also plan to adapt the scenarios to other pairs of languages and cultures apart from those of English (United States) and Spanish (Spain).

In order to facilitate the collaboration artifacts management, we also provide a repository, signifying that the artifacts are under version control and can be made available to the instructor. Our proposal, based on VAs, avoids the problems of other existing approaches related to coordination with other learners or institutions and minimizes the instructors' effort and the costs of infrastructure and maintenance. It is therefore easy to provide learners with a wide set of GSD problems in which they can train by using communication tools (chat, email).

Although we have presented a training scenario oriented towards training learners in collaboration through chat, our environment also permits the definition of scenarios in which the interaction can be made through emails. In addition, although our research is focused on GSD, this proposal is also extensible to outsourcing, offshoring or distributed software development education.

In our future work we intend to define training scenarios in which more than a learner can be involved. Thereby, each learner would play a different role in the scenario and they could interact during the meeting helped by the VC. The idea in this case consists of providing learners with an introduction to the problem and a set of artifacts and tasks that they can discuss during the interaction with their partner in order to collaboratively solve a GSD problem.

Finally, as part of our future work we plan to validate our final model by comparing the performance of the members of a company involved in real GSD projects trained with our model, with other members with similar skills who have not used it.

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. Ågerfalk, P., Fitzgerald, B., Olsson, H.H., Conchúir, E.Ó.: Benefits of Global Software Development: The Known and Unknown. Making Globally Distributed Software Development a Success Story, vol. 5007, pp. 1-9 (2008)
2. Damian, D., Hadwin, A., Al-Ani, B.: Instructional design and assessment strategies for teaching global software development: a framework. Proceedings of the 28th international conference on Software engineering, pp. 685-690. ACM, Shanghai, China (2006)
3. Damian, D., Zowghi, D.: Requirements Engineering challenges in multi-site software development organisations. Requirements Engineering 8, 149-160 (2003)
4. Lutz, B.: Linguistic Challenges in Global Software Development: Lessons Learned in an International SW Development Division. Fourth IEEE International Conference on Global Software Engineering (ICGSE'09), pp. 249-253. IEEE Computer Society, Limerick, Ireland (2009)
5. Prikladnicki, R., Pilatti, L.: Improving Contextual Skills in Global Software Engineering: A Corporate Training Experience. IEEE International Conference on Global Software Engineering (ICGSE'08), pp. 239-243. IEEE Computer Society, Bangalore, India (2008)
6. Herbsleb, J.D., Moitra, D.: Global software development. IEEE Software 18, 16-20 (2001)
7. Ocker, R., Rosson, M.B., Kracaw, D., Hiltz, S.R.: Training Students to Work Effectively in Partially Distributed Teams. Trans. Comput. Educ. 9, 1-24 (2009)
8. Richardson, I., Moore, S., Paulish, D., Casey, V., Zage, D.: Globalizing Software Development in the Local Classroom. Proceedings of the 20th Conference on Software Engineering Education & Training, pp. 64-71. IEEE Computer Society (2007)
9. Toyoda, S., Miura, M., Kunifuji, S.: A Case Study on Project-Management Training-Support Tools for Japanese/Chinese/Indian Offshore Development Engineers. In: Berlin, S. (ed.) Knowledge-Based Intelligent Information and Engineering Systems, vol. 4693, pp. 1222-1229, Heidelberg (2009)
10. Gotel, O., Kulkarni, V., Scharff, C., Neak, L.: Working Across Borders: Overcoming Culturally-Based Technology Challenges in Student Global Software Development. Proceedings of the 2008 21st Conference on Software Engineering Education and Training, pp. 33-40. IEEE Computer Society (2008)
11. Richardson, I., Milewski, A.E., Mullick, N., Keil, P.: Distributed development: an education perspective on the global studio project. Proceedings of the 28th international conference on Software engineering, pp. 679-684. ACM, Shanghai, China (2006)
12. Favela, J., Peña-Mora, F.: An Experience in Collaborative Software Engineering Education. IEEE Softw. 18, 47-53 (2001)
13. Burnell, L.J., Priest, J.W., Durrett, J.R.: Teaching Distributed Multidisciplinary Software Development. IEEE Softw. 19, 86-93 (2002)

14. Braun, A., Dutoit, A.H., Harrer, A.G., Brüge, B.: iBistro: A Learning Environment for Knowledge Construction in Distributed Software Engineering Courses. Proceedings of the Ninth Asia-Pacific Software Engineering Conference, pp. 197-203. IEEE Computer Society (2002)
15. Adya, M., Nath, D., Malik, A., Sridhar, V.: Bringing global sourcing into the classroom: experiential learning via software development project. Proceedings of the 2007 ACM SIGMIS CPR conference on Computer personnel research: The global information technology workforce, pp. 20-27. ACM, St. Louis, Missouri, USA (2007)
16. Lutz, B.: Training for Global Software Development in an International "Learning Network". Proceedings of the International Conference on Global Software Engineering, pp. 140-150. IEEE Computer Society (2007)
17. Lago, P., Muccini, H., Beus-Dukic, L., Crnkovic, I., Punnekkat, S., Vliet, H.V.: Towards a European Master Programme on Global Software Engineering. Proceedings of the 20th Conference on Software Engineering Education & Training, pp. 184-194. IEEE Computer Society (2007)
18. Bmegge, B., Dutoit, A.H., Kobylinski, R., Teubner, G.: Transatlantic project courses in a university environment. Proceedings of the Seventh Asia-Pacific Software Engineering Conference, pp. 30-37. IEEE Computer Society (2000)
19. Liu, X.: Collaborative global software development and education. In: Conference Collaborative global software development and education, pp. 371. (Year)
20. Petkovic, D., Thompson, G.D., Todtenhoefer, R.: Assessment and comparison of local and global SW engineering practices in a classroom setting. Proceedings of the 13th annual conference on Innovation and technology in computer science education, pp. 78-82. ACM, Madrid, Spain (2008)
21. Bellur, U.: An Academic Perspective on Globalization in the Software Industry. Proceedings of the 30th Annual International Computer Software and Applications Conference, vol. 1, pp. 53-54. IEEE Computer Society (2006)
22. Lago, P., Muccini, H., Babar, M.A.: Developing a Course on Designing Software in Globally Distributed Teams. IEEE International Conference on Global Software Engineering (ICGSE'08), pp. 249-253. IEEE Computer Society, Bangalore, India (2008)
23. Petkovic, D., Thompson, G., Todtenhoefer, R.: Teaching practical software engineering and global software engineering: evaluation and comparison. SIGCSE Bull. 38, 294-298 (2006)
24. Swigger, K., Aplaslan, F.N., Lopez, V., Brazile, R., Dafoulas, G., Serce, F.C.: Structural factors that affect global software development learning team performance. Proceedings of the special interest group on management information system's 47th annual conference on Computer personnel research, pp. 187-196. ACM, Limerick, Ireland (2009)
25. Boldyreff, C., Kyaw, P., Lavery, J., Nutter, D., Rank, S.: Towards Collaborative Learning via Shared Artefacts over the Grid. In: Conference Towards Collaborative Learning via Shared Artefacts over the Grid. (Year)
26. Gotel, O., Kulkarni, V., Scharff, C., Neak, L.: Students as Partners and Students as Mentors: An Educational Model for Quality Assurance in Global Software Development. In: Berlin, S. (ed.), vol. 16, pp. 90-106, Heidelberg (2009)
27. Bouillon, P., Krinke, J., Lukosch, S.: Software Engineering Projects in Distant Teaching. Proceedings of the 18th Conference on Software Engineering Education & Training, pp. 147-154. IEEE Computer Society (2005)
28. Meneely, A., Williams, L.: On preparing students for distributed software development with a synchronous, collaborative development platform. Proceedings of the 40th ACM technical symposium on Computer science education, pp. 529-533. ACM, Chattanooga, TN, USA (2009)
29. Gaeta, M., Ritrovato, P.: Generalised Environment for Process Management in Cooperative Software Engineering. Proceedings of the 26th International Computer Software and

Applications Conference on Prolonging Software Life: Development and Redevelopment, pp. 1049-1053. IEEE Computer Society (2002)
30. Boldyreff, C., Nutter, D., Rank, S.: Active Artefact Management for Distributed Software Engineering. Proceedings of the 26th International Computer Software and Applications Conference on Prolonging Software Life: Development and Redevelopment, pp. 1081-1086. IEEE Computer Society (2002)
31. Schümmer, T., Lukosch, S.: Understanding Tools and Practices for Distributed Pair Programming. Journal of Universal Computer Science 15, 3101-3125 (2009)
32. Schümmer, T., Lukosch, S., Haake, J.M.: Teaching distributed software development with the project method. Proceedings of the 2005 conference on computer support for collaborative learning: learning 2005: the next 10 years!, pp. 577-586. International Society of the Learning Sciences, Taipei, Taiwan (2005)
33. Haake, J.M., Schümmer, T., Haake, A., Bourimi, M., Landgraf, B.: Supporting Flexible Collaborative Distance Learning in the CURE Platform. Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 1 - Volume 1, pp. 10. IEEE Computer Society (2004)
34. Berkling, K., Geisser, M., Hildenbrand, T., Rothlauf, F.: Offshore Software Development: Transferring Research Findings into the Classroom. In: Berlin, S. (ed.) Software Engineering Approaches for Offshore and Outsourced Development, vol. 4716, pp. 1-18, Heidelberg (2007)
35. Swigger, K., Brazile, R., Harrington, B., Peng, X., Alpaslan, F.: Teaching Students How to Work in Global Software Development Environments. In: Conference Teaching Students How to Work in Global Software Development Environments, pp. 1-7. (Year)
36. Uresti, J.A.R., Boulay, B.d.: Expertise, Motivation and Teaching in Learning Companion Systems. Int. J. Artif. Intell. Ed. 14, 193-231 (2004)
37. Chou, C.-Y., Chan, T.-W., Lin, C.-J.: Redefining the learning companion: the past, present, and future of educational agents. Comput. Educ. 40, 255-269 (2003)
38. Hall, E.T.: Beyond Culture. Anchor Press (1976)
39. Hofstede, G., Hofstede, G.J.: Cultures and organizations: software of the mind, New York, NY, USA (2005)
40. Grzega, J.: Reflection on Concepts of English for Europe: British English, American English, Euro-English, Global English. Journal for EuroLinguistiX 2 44-64 (2005)
41. Axtell, R.E.: Do's and Taboos of Using English Around the World. Wiley, Canada (1995)