VENTURE: Towards a Framework for Simulating GSD in Educational Environments

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 this paper we propose an educational framework for designing training courses which focus specifically on the problems entailed in Global Software Development (GSD). The main issues involved are related to cultural, language and communication problems. The principal element of the framework is VENTURE; a virtual training tool that enables learners to get immersed in realistic GSD scenarios. Using this tool, learners will come face to face with collaborative, organizational and technical problems of GSD by interacting with virtual participants in a simulation of people working together from locations that are distant from each other. By means of this framework we also establish the theoretical contents and procedures which would customize the courses for use in university settings or Master programs.**

We therefore consider that VENTURE is a suitable platform for developing the students' skills in GSD and we believe that it will allow them to deal with real-world problems, thereby minimizing the effort and costs involved in course design.

Keywords: Global Software Development, Engineering Education, Educational Framework, Educational Environment, Virtual Agents.

I. INTRODUCTION

The recognized benefits of Global Software Development (GSD) [1] have been a factor in the increase in its application in recent years. Participants involved in this activity must be trained in dealing with the specific drawbacks that also appear in this type of environments, however. The main problems are usually the result of the distance that separates the virtual teams, as well as the time and cultural and linguistic differences that appear when distant members interact using communication and collaboration tools [2]. Traditional face-to-face meetings are not the rule in distributed projects and interaction usually occurs through the use of communication tools [3]. That being so, communication and coordination become more complex than in traditional development.

In GSD, it is difficult to reach a common understanding, since members from different countries interact using their cultural knowledge and communication styles [4] and interpret the communication from their particular perspective [5]. Participants must know how to adapt their messages so that the interlocutor can understand them, without missing information or creating a misunderstanding [6].

Software engineering students must be trained in the specific problems of GSD. This fact has led to the need for Software Engineering Education to be adapted, making it able to tackle the new challenges that GSD involves [3].

In this paper, we present a proposal which aims to tackle current needs, by carrying out the following steps: (1) designing a framework to support the training of GSD, (2) building a graphical, interactive, educational and customizable simulator for training in GSD (VENTURE), (3) developing a set of courses which include theory and simulation scenarios that also incorporate virtual meetings, (4) designing a proposal for evaluating the use of the environment by both students and instructors not only in actual software engineering courses, but also in formal out-of-class experiments, thus obtaining feedback on the educational aspects.

The framework presented has the goal of giving training in GSD, with two basic objectives: the first is to provide theoretical and practical lessons that allow students to acquire communicative and teamwork abilities through the simulation of multicultural GSD environments. The second goal is to provide instructors with support. That would help them when designing both the theoretical content and the practical simulations that can be produced for training in different stages of GSD. The primary focus of all this is that of specific issues concerning differences in culture.

The main component of this framework is VENTURE (Virtual ENvironment for Training cUlture and language problems in global softwaRe dEvelopment), a virtual training environment that places learners in a simulated GSD scenario in which they are involved in realistic experiences through their interaction with Virtual Agents (VAs) from different cultures. VAs have the advantage of always being available, meaning that users can work with this tool at any moment and at any time.

VENTURE permits us to design virtual meetings and interviews in which the students interact with different avatars. The definition of the meeting is based on a workflow model containing the information required for the interaction. The

aim is to provide training in cultural and communicative problems which occur when using written communication tools, such as e-mail and instant messaging.

In order to achieve our learning scopes, a large part of our work consists in establishing a broad set of cultural problems that may appear in GSD. It also means measuring the degree to which these are affected by the different cultural dimensions set out by House [7], who presents a recent model dealing with cultural differences, defining eight dimensions: uncertainty avoidance, institutional collectivism and in-group collectivism, assertiveness, future orientation, human orientation, performance orientation, power distance and gender egalitarianism.

Therefore, the goal is to establish the relationships between cultural problems and cultural dimensions. In doing so, we will be able to quantify the degree to which each cultural problems may appear during the training process, taking into account the particular cultures of the participants in the virtual meetings.

This paper is organized as follows: Section II outlines the related works in the field of GSD training and education. Section III describes the virtual learning environment developed in the context of this research. Section IV explains the virtual interactions carried out through our environment. Finally, Section V gives some concluding remarks and Section VI gives an outline of our future work.

II. RELATED WORKS

In this section we summarize a rigorous literature review [8] that was performed to answer the following research question: *What initiatives have been carried out in relation to GSD training and education?*

The different contributions towards Distributed Software Development, Open Ended Group Projects and Virtual Teams in the fields of education were also analyzed, since these are also connected with this context.

A. Theoretical and practical approaches

Most studies in the field of teaching and training GSD relate experiences in theoretical and practical classes in university degrees and masters programs which are adapted to the specific characteristics of globalization. These studies commonly report on joint courses with universities from different countries reproducing scenarios like those that can be found in industry, allowing students from different cultures to interact. By way of example, in [9] a European Masters program on Global Software Engineering involving several universities from different countries is presented. Another master course in the field of Computer Networks in an Internationally-Distributed Project-Based Course can be found in [10]. In this, students had to develop a software system involving learning to collaborate and take decisions in a distributed team.

[11] describes a course in collaborative learning which addresses issues of GSD, such as those related to group decision support and the use of collaborative technologies. In this case, teams were designed according to the ability and skills of their members. Interpersonal qualities, as well as technical ones, were taken into account, so that teams had a comparable distribution of skills overall. Team members' cultural and language backgrounds were also borne in mind in the design of this allocation, thereby obtaining multicultural groups and ensuring a realistic simulation of global virtual collaboration.

A similar practice is described in [12], which presents an experience in which distant team members collaborated by documenting each task in a project, sending that electronically to the other university groups in charge of carrying out the tasks.

The field of Open Ended Group Projects (OEGPs) supports a similar idea. OEGPs are make up a structure that is suitable for dealing with real-world problems and for developing the knowledge and problem-solving skills required in GSD [13]. One noteworthy experience is that presented in [14], which provides a flexible course that evolves with the assistance of an action-research program allowing improvements to be made through a combination of learning theories and stakeholder input. The introduction of an external mentor was useful in supporting the project leaders and other participants. It also proved to be helpful for improving the quality of the project results [15].

One of the important factors addressed here is the time that learners need to get a full understanding of the problem that has to be solved. Time for reflection and discourse is also vital in conveying the longer-range core of skills that students will require in their future professional careers. In [13] it is argued that OEGP educational environments are more appropriate towards the end of the education study program.

Finally, we found some educational approaches in enterprises, such as a training course in a multinational organization that applies GSD [16]. In these cases, learners are put in contact with experienced colleagues; the greater availability of an experienced workforce makes the application of the concept of **learning networks** [17] possible. Learning networks consist of a way of teaching that takes advantage of the knowledge of workers in a company who can train learners in specific skills.

B. Learning environments

Experimental and research environments were found. One example of this is the collaborative environment reported in [18], in which distributed teams are trained through the use of a set of tools (including chat, a scribble tool, an application sharing tool, a graphics tools for designing UML documents, etc.). In this category we also found the collaborative platform ClockingIT [19], a customizable web-based platform that provides instant messaging features, task assignment and management, discussion boards, a file area and wiki pages. These wiki pages were essentially used to manage the knowledge and share news and information about the project. The platform is open-source and admits a certain degree of customization to adapt it to certain projects. Similarly, iBistro [20] is an augmented space in which distributed learners collaborate in the development of a software project and so "learn by doing".

[21] explores the interaction with avatar-based humans in virtual collaborative projects. Avatars can access a virtual room and may walk, run and show emotion by waving or smiling. This system allows collaboration skills and intercultural competence to be taught and learnt by performing real-time realistic activities.

Teamlink [22] is a Collaborative Virtual Environment based on configurable avatars in a 3D virtual space. It supports icebreaking activities so that team members will be introduced to each other, establishing trust between virtual team members by using asynchronous communications. Another example is the collaborative virtual learning environment CURE [23], which uses virtual rooms for collaboration. These rooms may contain pages (content), communication channels (such as chat, threaded mailbox, etc.), and users, who will interact with other users located in the same room.

Finally, we should mention the use of Jazz [24] a commercial collaborative development platform that supports certain functionalities of the development life cycle, such as source code repository, chat, web interface, report generation and work items, and which has been applied in GSD educational environments.

C. Conclusions and Implications

One of the problems as regards training in collaborative groups is that participants in educational environments are not always active. Some team members can have *little motivation* or inappropriate knowledge or skills to deal with the problem [25], resulting in inefficient conversations related to topics that are not related to the exercises. Another related problem is ineffective communication through chat or email, which produces missed deadlines. The use of the appropriate tools and environments employed to simulate realistic environments frequently entail *technical issues* [26].

As a consequence of all we have described, the teaching and training of GSD is a complex task that must be supported by practical experience. Reproducing GSD environments in educational contexts is difficult, however. Moreover, existing literature is generally oriented towards a specific stage of the software development, such as the requirements stage or software development [20], or towards familiarizing participants with the use of certain GSD tools [27].

One of the most noteworthy subjects reported consists in improving formal and informal communication skills [28], and also in helping reduce the difficulties created by cultural diversity. It must be said, however, that giving training in dealing with these factors is complex, due to scheduling incompatibilities and the difficulty of working in conjunction with distant institutions [28]. What is more, systematic training in dealing with cultural and linguistic problems by the above means is difficult, ultimately. That is because during the interactions these problems may appear randomly, depending on the specific circumstances of certain settings [26]. On the other hand, these methods do not provide opportunities to make corrections, since teachers cannot attend all of the meetings and provide feedback.

In other disciplines, such as in traditional co-located development, successful proposals exist, focusing on simulation. This allows students to practice realistic software engineering processes in a more rapid and efficient manner. A representative example is SimSE [29], an interactive game with a graphical user interface in which the simulated physical office is displayed, creating a fun atmosphere in which the students are penalized with a bad score when they do not follow proper practices. SimSE is designed to illustrate the particular software engineering process to students. These processes may be designed through a model builder, to simulate realistic conditions, in line with what research literature recommends on the matter.

III. INTRODUCTION TO VENTURE

In this section we present the main component of VENTURE, which consists of a platform that is integrated into an e-learning system providing support to the GSD educational framework presented.

This platform is intended to provide theoretical lessons and simulated practices in GSD, supported by a tool that simulates realistic GSD collaborative environments. The main novelty of this framework is the rigorous support for training in coping with cultural and linguistic differences. It also improves attitudes for collaborative group work in the context of GSD, without requiring real partners.

VENTURE makes it possible to simulate GSD virtual meetings in which students interact with VAs using a common language (usually English). VAs will communicate with students in an autonomous manner, in an attempt to perform some tasks following the lessons and guidelines given in the theoretical lessons. Students thus face cultural and linguistic problems similar to those that appear in real environments.

In class, teachers will provide students with the theoretical lessons on GSD activities for 2 hours per week. In the laboratory, students use VENTURE to get the theoretical material, as well as to execute the training scenarios through the chat and email simulators, interacting with teachers and other students using chat, email, wikis and forums. VENTURE also contains file repositories so that students can access the artifacts required for the training lesson and in addition they can upload their deliverables here.

After studying these theoretical concepts, learners can continue with the practical part. In this regard, the simulation is based on a Problem-Based Learning (PBL) approach [30], in which learners are placed in a virtual training scenario and they work in the resolution of typical GSD activities by interacting with VAs of different cultures who will play a role in the scenario.

The interaction between students and VAs from different cultures is supported by the user chat or email, in order to train both synchronous and asynchronous interaction. The scope for the students is to participate in a realistic software project by playing the different roles of the process, through interacting with the VAs. Each lesson has deadlines for the delivery of the practical material; these are established on a weekly basis. At the end of a course the student may also be asked to answer a questionnaire.

The tool introduces students to the characteristics of the company in which they are working, supposedly, along with the project to be developed and their role. They are also given a software engineering task or a responsibility that they must accomplish, usually by interacting with other participants (actual or virtual). The instructor can model these scenarios so that the student can perform activities in any field of software engineering.

The success of each practical deliverable depends on the quality and the adequacy of the communication with the VA, but it also depends on the technical skills of the student when carrying out the task. The virtual meetings are designed to reflect typical problematic or controversial situations of GSD and take into account common language and cultural mistakes, where students are encouraged to find the solution to the problems by interacting properly.

Trainees have a weekly thirty minute chat scheduled, in which they must get as much information as possible by interacting appropriately in order to complete the exercises. Email can be also used to schedule meetings or obtain more information about specific tasks.

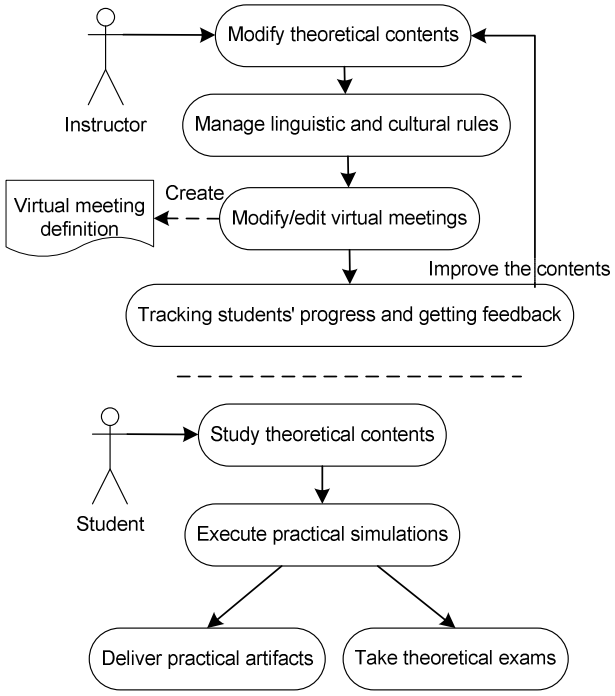


Figure 1. Process followed in our framework

After this interaction students will generally complete some document, such as a requirement elicitation document, or they may develop some software in accordance with purposes of the scenario. The main objective is to minimize misunderstandings and to increase the performance of the students when dealing with cultural problems during the simulated conversations. That being the case, the courses are structured in such a way as to accomplish the following goals:

- To encourage learning through peer-teaching and gain experience in collaboration with a group. Students are able to be in contact with their partners to resolve the problem scenarios.
- To gain experience in the theoretical knowledge learned in class by applying it in practice.
- To provide a well-organized setting with courses and training scenarios, which can be run at lower costs, thus removing the difficulties involved in coordination with distant members and reducing the teacher's workload.
- Incremental improvement of the framework with the incorporation of feedback, new material and training scenarios.
- To provide students with experience in teamwork with people from a foreign culture through VAs.

During the process, the instructors obtain feedback about the use of the tool and these results can be used to iteratively improve or modify the theoretical contents, the linguistic and cultural rules and the virtual meetings. The process described is detailed in Figure 1.

IV. ARCHITECTURE OF VENTURE

In the following sections details of the architecture of VENTURE are explained.

A. Main components of VENTURE

VENTURE presents a modular client-server architecture and is made up of a set of components as shown, which permits the design and development of GSD simulators by instantiating the modules presented in Figure 2.

The server side consists of a web e-learning application, in which the students will access their available courses, made up of the following modules:

Resources repository (1)**:** Students can access the theoretical lessons assigned. A theoretical lesson can contain one or more practical activities, which usually consists of a virtual meeting.

Tasks area (2): Specifying the practical lessons assigned, including the deadlines for the deliveries. Students can execute their activities, upload deliverables and review the evaluation and instructor's comments for these activities.

Forum and wiki module (3, 4): In which the instructors make announcements and students can interact with instructors and partners.

Evaluation area (5): Instructors can design exams or questionnaires, grouped by thematic area, which serves to evaluate the students' learning. These activities can also have a deadline and be limited in time.

By considering this architecture, instructor**s** can manage the training courses and the learners' activities; they can also be aware of the status of the tasks and actions the learners are carrying out and they will be able to communicate with them. On the other hand, students can access the training materials and manage their assigned training scenarios, also accessing their information about their performance, deadlines, qualifications, historical actions. Students can also know which team members are online, receive news, share artifacts or fill out questionnaires and take exams.

The client side is made up of two applications: Workflows Designer, which allows us to modify and design the practical

training simulations (in the case of the instructors and scenario designers), and which lets the GSD Simulator execute the simulations (in the case of the students).

With regard to the training of GSD skills, in the subsequent sections we set out the following components of VENTURE (shown in Figure 2):

C1) Pedagogical module, with the theoretical materials adapted to GSD problems.

C3) Cultural problems module, including the problems of cultural distance.

C2) Linguistic problems module, containing the rules which serve to correct the learners' interaction.

C4) Skills required in GSD repository, which will be used in the training scenarios.

C5)Workflows Engine

C6) VAs profile component, which will participate in the virtual simulations.

C7) Virtual meetings engine, in charge of executing the virtual meetings.

C8) Evaluation module, responsible for the evaluation process.

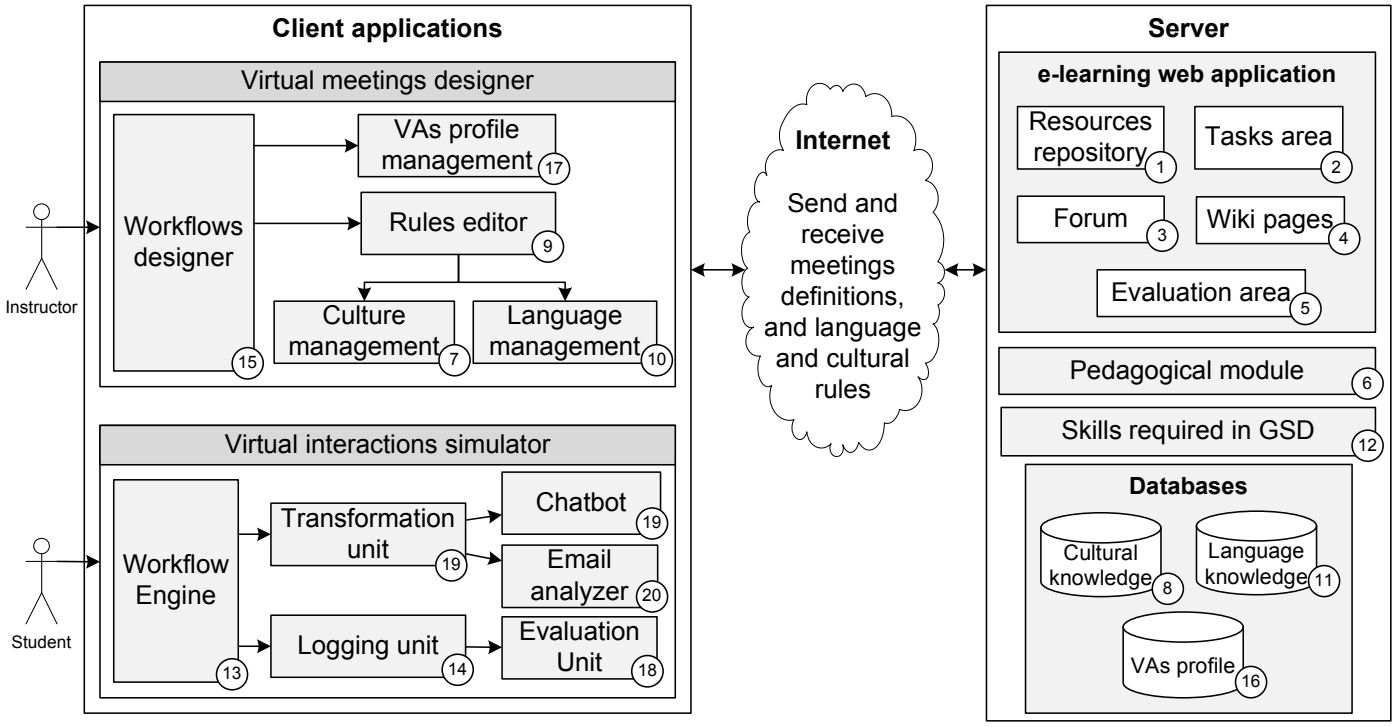


Figure 2. Venture architecture

1) Pedagogical module

The contents of the theoretical classes are intended to improve the students' ability to apply appropriate Software Engineering knowledge to solve problems similar to those that appear in GSD professional domains.

This material is stored in the Pedagogical Module (6) and is structured with reference to the different knowledge areas of Software Engineering, documented in SWEBOK [31]: software requirements, software design, software construction, software testing, software quality, software maintenance, configuration management, software engineering management and software engineering process. In addition, theoretical content in the different skills that are needed in GSD is provided, which will be assigned to each student depending on his/her profile. After executing a virtual training scenario, the evaluation module will indicate the student's profile regarding his/her GSD skills. Taking into account the student's deficiencies in certain areas, the Pedagogical module will assign a strategy to guide the learning process for that student. This strategy consists in assigning the theoretical lessons and practical activities designed to give training in those skills.

The final decision about what lessons are assigned to the student is taken by the instructor, who can be in contact with the trainee and who may examine the logs of the practical sessions, which are useful for analyzing students' behavior.

2) Cultural problems module (7)

A repository containing the description of the classic cultural problems and recognized differences that can affect communication and collaborative work.

This module requires an in-depth study of the state-of-the-art in the problems that derive from each particular culture. In this regard, some typical cultural problems that occur in GSD are:

- Participants do not speak during team discussions until invited to do so, since this is not considered polite in some cultures [32].

- In other cultures, people tend to say that they have understood something when they have not, thus causing problems that may be detected too late [32].
- The 'Mum Effect' [33], refers to the tendency to cover up critical information or to distort negative news by presenting it as more positive information. However, getting timely and accurate negative information may be critical to the project's success.
- The use of direct or indirect style refers to the way in which people reveal their intentions or information. Being too direct may appear very rude for some cultures, while being too indirect implies an excessive deviation from the conversation that can be annoying for other cultures [34].

This knowledge is stored in a *cultural database* (8) which lists any kind of cultural problem regarding the cultures of the members that may be involved during a simulated conversation in VENTURE. The rules included are classified by the kind of problem that they deal with, along with a description of the difficulty, the relative seriousness of making each mistake and the actions that the VC will take to correct the student. The information contained in this database can be managed by the instructors through the *Rules Editor* interface (9).

3) Linguistic problems module (10)

A repository contains the description of the linguistic problems and recognized problems that can affect communication when participants interact with a non-native language.

This content requires an in-depth study of the common problems that can appear and as this is stored in the *language database* (11), it is possible to include any kind of linguistic problem regarding the languages involved, classified by the kind of problem that they deal with and including any relevant information. As in the case of the cultural rules, this information is also managed through the Rules Editor interface.

To be specific, this module is meant to bear in mind that the use of a non-native language brings with it particular problems. Instances of this include the overuse of certain verbs of high semantic generality (do, have, make, put, etc.), along with the use of false friends (where a word looks or sounds similar in two different languages, but differs in meaning, possibly causing misunderstandings). Grammatical inaccuracies may also lead to a breakdown in comprehension [34], [32].

4) Skills required in GSD module

The *skills required in GSD module* (12) is a repository with best practices classified by the following skills, which we obtained from a systematic review on the matter [8]:

- Performance in the use of synchronous and asynchronous means of communication.
- Ability to communicate effectively using a common terminology and language.
- Informal communication and improvisation skills.
- Knowledge of language, cultural and ethical issues.
- Leadership and conflict resolution skills.
- Time management skills.
- Ability to think from the perspective of the other side and understand people from different backgrounds.
- Managing ambiguity and uncertainty. Ability to evaluate information critically.
- Knowledge of negotiation skills and contract writing in a common language.
- Collaborative work skills.
- Skills to gain the interlocutor's confidence and trust.

5) Workflows Engine

Responsible for executing the meeting workflows defined in VTRML format. Its main components are: the workflow orchestrator, which guides the execution of the workflow, and the chatbot system, which processes the natural language by interpreting AIML (Artificial Intelligence Markup Language). This contains the information required to simulate an intelligent conversation with the VAs.

To unpack this idea, we can say that the *Workflows Engine* (13) reads the definition of the meeting and orchestrates the sequential execution of the different phases. It does so by interpreting their content and by extracting the conversational knowledge, together with the linguistic and cultural rules defined to extract the AIML information in the context of a certain phase in the conversation.

Moreover, the engine is responsible for controlling other aspects that are also established in the VTRML file, such as the execution time, the VAs' actions and emotions, and the evaluation of the students' actions when they make mistakes during the conversation. In this respect, the engine also makes it possible to save the log through the *login unit* (14) of the conversation, so that the instructor can review it later.

Finally, as this engine is able to execute several sequential workflows concurrently, simultaneous conversations may be simulated to take place as they do in real environments.

6) Workflows Designer

This is a graphical designer that the course designers use for defining and modifying the virtual meetings. The virtual meetings are structured as sequential workflows made up of a set of phases that are assembled on the *Workflows Designer* (15) canvas by dragging them from a toolbox. Although the aim of the framework presented in this paper is to minimize the role of the instructors, they will also be able to modify and adapt the structure of the predefined meetings provided by using this designer.

All the arguments required by the workflow are assigned within the designer, including the specific properties of the phases, as well as the conversational knowledge and the cultural and linguistic rules.

Based on the graphical workflows, the *Workflows Designer* creates the definition of the workflow on VTRML format automatically.

During its edition, the workflows can be tested by the designer, who can execute a specific phase in order to improve it, if it finds mistakes. After finishing the definition of a

workflow it can be uploaded into the training course so that students may execute it.

7) VAs profile database

This contains the information regarding the virtual characters involved in the training scenarios (VAs and VCs), defining their appearance, emotions and gestures.

Their image may appear in the theoretical material in order to show examples. For instance, the instructor could show examples of typical gestures in certain situations within a specific culture. In the theoretical scenarios these will appear in the conversations. For instance, during a chat they will gesticulate and show emotions according to the context of the conversation.

Instructors can access the management of the *VAs profile (16)* through the *VAs profile management* module (17), to include new characters or modify existing ones, and also to incorporate them into the theoretical materials or practical scenarios.

8) Evaluation unit

The *Evaluation unit* (18) informs both students and instructors about their skills and results from the use of this framework, providing a continuous evaluation of the students' skills.

Instructors can use this information to determine what skills the students must improve, according to what these evaluations show. The goal is to assign an appropriate training module to them, focusing on their particular needs.

To be more specific about all this, we should explain that our framework measures factors like: time consumed in each theoretical module, evaluation in the practical exercises and in the theoretical exams, delay in the upload of the deliverables, etc., and also several factors regarding the activity of the student during the virtual meetings. We could, for example, take into account the percentage of conversational knowledge that was not triggered, average response time, number of corrections made by the VC, including the type of the mistakes corrected and their relative severity in the context of the conversation.

B. Virtual Interactions

The virtual interactions are supported by means of the following plugins:

Chat plugin: which trains in synchronous communication with one or more VAs. In this case, the meeting is guided by a VC. The VC helps the students to cope with the meeting and corrects their actions, providing rationales and explaining their consequences. This plugin is set out in subsection A.

E-mail plugin: for giving instruction in asynchronous communication. In this case, the student will send and receive e-mails from a VA.

The interactions carried out through these plugins are defined in a workflow that determines the flow of the conversation. The definition of these workflows is based on VTRML (VenTuRe Markup Language), which allows the definition of all the elements required. It is an extension to the XAML (Extensible Application Markup Language) declarative language.

The VTRML-based sequential workflows contain the AIML language embedded in their definition. In Figure 3, an example is shown of the definition of a workflow which is based on an XML scheme generated automatically by the *Workflows designer*.

These workflows, executed by the *workflow engine*, are responsible for orchestrating the sequence of actions during the interaction through the different VAs. This engine reads all the information related to the simulation and for each phase it makes the appropriate transformations so as to generate the AIML language. It does this by using the *transformation unit* (19) to obtain information that is understandable by the *chatbot system* (19).

C. Synchronous Interactions Specification

In this section we give a description of how the virtual meetings for the chat plugin are designed, as well as how they are executed form the students' point of view.

In this regard, virtual meetings are defined through the *Workflows designer* as a sequential workflow composed of a set of phases containing the main information for their definition as follows:

- *VAs which are* involved in the scenario and are going to take part in the conversation.
- *conversational knowledge for each VA, containing the patterns and the predefined answers for each template.*
- *cultural and language problems* that can appear in the context of that phase of the interaction.
- *resources* (artifacts) that may be needed during the interaction and which are referenced by the VAs as hyperlinks.
- *emotions and gestures* that the VA's will express in the context of a phase during the interaction.
- *type of cultural or language problem and its severity*, which allows the nature of the problems that the user encounters during the phase to be quantified, depending on the particular cultural and language rules that are triggered.

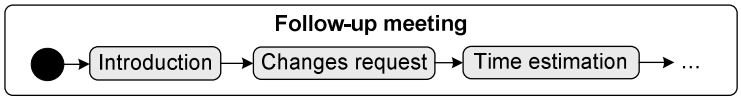
The *Workflows designer* makes it easy to define this information through a wizard, in which the instructor will design the conversation phase by phase. The specific linguistic and cultural rules that can be relevant in the context of each phase are taken into account. These cultural and linguistic rules can be imported from the *linguistic and cultural databases*.

The conversational knowledge, along with the linguistic and cultural rules are codified by using AIML- This is interpreted by the chatbot system used by the chat plugin, in which a *pattern* is a string of characters intended to match one or more user inputs. They may contain wildcards, which match one or more words and can also manage synonyms so that the different ways of saying the same thing are taken into account. A *template* specifies the response to a matched pattern that will be expressed by a VA.

The example shows the definition of a tracking meeting with the client, in which the student plays the role of software analyst through the phases: *introduction*, in which the participants will have the matter presented to them, *changes request*, in which the Virtual Customer makes a request to the

student and *time estimation*, in which they will discuss the time required to perform the change. How the conversational knowledge is represented in the model is set out for each phase, as are the linguistic and cultural rules.

For the definition of this XML-based workflow, we have considered the indications of Clear et al. [35], in which the structure of a meeting in international collaborative settings is described.




```
<Workflow Name="Requirements Elicitation Meeting" duration="25:00">
  <Sequence>
    <Phase Name="Introduction">
        <!--Not detailed"-->
    </Phase>

    <Phase Name="Changes Request">
        <ConvesationalKnowledge>
          <category order="1" gesture="concerned"> <!--VirtualCustomer1 starts the conversation (order=1)-->
              <template VA="VirtualCustomer1">We also need the notifications system for acknowledging
              the reception of new incidents for next week. Will it be ready on time?</template>
          </category>
          <category gesture="thinking">
              <pattern>*information *</pattern> <!--The student requires further information-->
              <template VA="VirtualCustomer1" sendFile="NoficationMessage.pdf">I'm sending you an example of our current
              manual messages here:</template>
          </category>
          <category>
              <pattern>*address the incidents*</pattern> <!--The student asks know how the incidences are addressed-->
              <template VA="VirtualCustomer1">After receiving the notification he will log into the system and indicate
              the time estimated to resolve it.</template>
          </category>
        </ConvesationalKnowledge>
        <CulturalProblems>
          <CulturalProblem type="Say yes without undestanding" severity="5">
            <category gesture="thinking">
              <pattern>* yes *</pattern>  <!--Affirmative response of the student without knowing the consequences-->
              <template VA="VirtualColleague">You should know more the information before responding positively.
                                              Remember that you also have to finish other tasks.</template>
            </category>
            <category>
              <pattern>of course *</pattern>
              <template>
                <srai>* yes *</srai> <!--Using <srai/> we control the multiple possible answers of the student.
                                     In this case, the entry "of course *" will have the same behaviour as "* yes *"-->
              </template>
            </category>
          </CulturalProblem>
        </CulturalProblems>

        <LanguageProblems>
          <LanguageProblem type="false friend" severity="2">
            <pattern>attend</pattern> <!--Incorrect use of the word "attend"-->
            <template>"attend" is a false friend in Spanish. In English it means "to be present at …". Do you mean
            "address"?</template>
            </LanguageProblem>
        </LanguageProblems>
    </Phase>

    <Phase Name="Time Estimation">
      <ConvesationalKnowledge>
        <category order="1" gesture="concerned"><!--VirtualCustomer1 starts the conversation (order=1)-->
          <template VA="VirtualCustomer1">Will it be ready for next week?</template>
        </category>
        <category gesture="angry">
          <pattern>*more time*</pattern> <!--The student considers that s/he need more time"-->
          <template VA="VirtualCustomer1">But we cannot delay this too much</template>
        </category>
      </ConvesationalKnowledge>
    </Phase>
  </Sequence>
</Workflow>
```

Figure 3. Example of an VTRML-based sequential workflow definition.

Figure 4 shows a fragment of a real conversation based on the aforementioned workflow definition. The workflows are time-limited, so that the student can pass through to the next phase of the conversation when s/he considers that they do not

need more information to carry on with another part of the conversation. Each new phase is reached automatically when all the conversational knowledge has been used. The students' scope is therefore to get as much information as possible by triggering as many response templates as possible to achieve their task and by minimizing the cultural and language errors made, taking into account the particular VA culture.

In the example, we show how the Virtual Customer initiates the conversation for each phase. We then see how the student interacts with him, falling into a cultural and language mistake that is corrected by the VC.

Customer: We also need the notifications system for acknowledging the reception of new incidents for the next week. Will it be ready on time?

Student: Yes, it is going to be finished on time.

Virtual Colleague: You should know more the information before responding positively. Remember that you also have to finish other tasks.

Student: What information must the notification provide ?

Customer: I'm sending you an example of our current manual messages here: NoficationMessage.pdf

Student: How is the technician going to attend the problem?

Virtual Colleague: "attend" is a false friend in Spanish. In English it means "to be present at …". Do you mean "address"?

Student: How is the technician going to address the incidents?

Customer: After receiving the notification he will log into the system and indicate the time estimated to resolve it.

Customer: Will it be ready for next week?

Student: But this also requires changes into the tracking system. That needs more time.

Figure 4. Example of a conversation through the chat plugin.

The chat interface also implements a feedback functionality that records student feedback if they believe that they have interacted properly but that the VA did not respond in a logical manner. This feedback is stored and used in the process to refine the scenario in the future.

D. Asynchronous Interactions Specification

In the asynchronous interactions executed by the email plugin, the messages sent to the VA are analyzed and evaluated through the *Email analyzer* (20). It does that by searching for cultural and linguistic mistakes defined for the scenario. In this case the VC will correct the student by sending him/her an email reporting the mistakes made. The student will correct those and send back the email and so the conversation will continue.

The workflows created by the designer define the sequence of emails that will be exchanged. These workflows are different than those defined for the synchronous interactions, since they require different attributes for controlling addressees or time of delays in the answers. The phases in this case are also different, moreover, and they contain the following information, mainly:

- *VAs* involved in the interaction.
- *structure of the email and patterns expected from the student.* The structure of an email is defined as a XML in which the different patterns are embedded using AIML format. *The VC will inform the student if the structure or the content of his/her email is not appropriate.*
- *templates of answer with the responses that the VAs will give to answer a proper interaction of the student.*
- *cultural and language problems* that may appear in the context of that phase of the interaction and which are codified by using AIML.
- *type and severity of the cultural and language problem*, which allows the nature of the problems to be quantified.
- *resources* (artifacts) that can be referenced by the VAs as hyperlinks.

V. CONCLUSIONS

In this paper we have presented a framework that provides support for the training of GSD activities, focusing particularly on cultural and linguistic diversity of these environments.

We argue that the VENTURE tool removes the problems of interacting with distant partners. Moreover, students may have training in GSD at any time, making them more independent. Furthermore, the learning process is more controlled, since the training scenarios are rigorously designed to deal with specific problems systematically. The interaction focuses specifically on improving the student's weaker skills.

Using this framework, the students learn about the different kinds of problems that may occur from different perspectives. This is the reason why they can play different roles in the practical exercises while interacting with VA's. That allows them to train at any time, without depending on the availability of other partners or colleagues. Furthermore, using VAs reduces the effort required to coordinate distant learners or institutions and minimizes the instructors' effort and the costs of infrastructure and maintenance.

Instructors do not need to be experts in all the stages and problems of the GSD, since the lessons and predefined training scenarios are designed as the result of research in the matter, although our framework also permits their customization to be adapted to specific academic needs.

In this sense, we believe that this framework makes a contribution, for both practitioners and researchers. It tackles several points which may be interesting for the following fields of research and practice:

- Global collaboration and coordination of practice and education.
- Management of GSD development activities.
- Global virtual team research.
- Research into collaborative models and technologies.
- Computing Education research, Education Technology, Distance Learning and e-Learning.

ACKNOWLEDGMENT

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 the Consejería de Educación y Ciencia (Junta de Comunidades de Castilla-La Mancha), and co-funded by Fondos FEDER, in addition to MELISA (PAC08-0142-3315), the Junta de Comunidades de Castilla-La Mancha, the Consejería de Educación y Ciencia in Spain, and ORIGIN (IDI-2010043 (1-5)) funded by CDTI and FEDER.

REFERENCES

[1] P. Ågerfalk, *et al.*, "Benefits of Global Software Development: The Known and Unknown," in *Making Globally Distributed Software Development a Success Story*. vol. 5007, ed, 2008, pp. 1-9.

[2] 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.

[3] J. D. Herbsleb and D. Moitra, "Global software development," *IEEE Software,* vol. 18, pp. 16-20, 2001.

[4] K. S. Amant, "Cultures, computers, and communication: evaluating models of international online production," *Professional Communication, IEEE Transactions on,* vol. 44, pp. 291-295, 2001.

[5] L. Bordyuk, "Linguistic and culture-specific factors for professional success," in *CAD Systems in Microelectronics, 2003. CADSM 2003. Proceedings of the 7th International Conference. The Experience of Designing and Application of*, 2003, pp. 530-532.

[6] T. L. Warren, "National cultures in international communication," in *Professional Communication Conference, 1998. IPCC 98. Proceedings. 1998 IEEE International*, 1998, pp. 305-312 vol.2.

[7] R. J. House, *et al.*, *Culture, Leadership, and Organizations: The GLOBE Study of 62 Societies*: Sage Publications, 2004.

[8] M. J. Monasor, *et al.*, "Preparing students and engineers for Global Software Development: A Systematic Review," in *International Conference on Global Software Development (ICGSE 2010)*, Princeton, NJ, USA, 2010, pp. 177-186.

[9] 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.

[10] A. Berglund, "Learning Computer Networks in an Internationally Distributed Project-Based Course," in *33rd ASEE/IEEE Frontiers in Education Conference. [Online]. Available: http://fie.engrng.pitt.edu/fie2003/ [8 November 2003]*. Boulder, Colorado, 2003, p. S1F19.

[11] T. Clear and D. Kassabova, "A Course in Collaborative Computing: Collaborative Learning and Research with a Global Perspective," in *Proceedings of the 39th ACM Technical Symposium on Computer Science Education*, M. Guzdial and S. Fitzgerald, Eds., ed Portland, Oregon: ACM, 2008, pp. 63-67.

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

[13] A. Hauer and M. Daniels, "A learning theory perspective on running open ended group projects (OEGPs)," in *Conferences in Research and Practice in Information Technology*. vol. 78, Simon and M. Hamilton, Eds., ed Wollongong, NSW, Australia: ACS, 2008, pp. 85-92.

[14] M. Daniels, *et al.*, "Engineering Education Research in Practice: Evolving Use of Open Ended Group Projects as a Pedagogical Strategy for Developing Skills in Global Collaboration (Special issue on Applications of Engineering Education Research)," *International Journal of Engineering Education* vol. 26, pp. 795-806, 2010.

[15] Å. Cajander, *et al.*, "Introducing an External Mentor in an International Open Ended Group Project," in *39th ASEE/IEEE Frontiers in Education Conference.*, San Antonio, Texas 2009, pp. T1A1-T1A6.

[16] 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.

[17] O. Gotel, *et al.*, "Students as Partners and Students as Mentors: An Educational Model for Quality Assurance in Global Software Development." vol. 16, S. Berlin, Ed., ed Heidelberg, 2009, pp. 90-106.

[18] K. Swigger, *et al.*, "Teaching Students How to Work in Global Software Development Environments," in *International Conference on Collaborative Computing: Networking, Applications and Worksharing, 2006. CollaborateCom 2006.*, Atlanta, GA, USA, 2006, pp. 1-7.

[19] Å. Cajander, *et al.*, "Students analyzing their collaboration in an International Open Ended Group Project," in *39th ASEE/IEEE Frontiers in Education Conference.*, San Antonio, Texas 2009, pp. T1A1-T1A6.

[20] 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.

[21] D. Corder and A. U, "Integrating Second Life to enhance global intercultural collaboration projects," *ACM Inroads,* vol. 1, pp. 43-50, 2010.

[22] T. Clear and M. Daniels, "2D & 3D Introductory Processes in Virtual Groups," in *33rd ASEE/IEEE Frontiers in Education Conference.*, Boulder, Colorado, 2003, pp. S1F1-SIF6.

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

[24] 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.

[25] A. Soller, "Supporting Social Interaction in an Intelligent Collaborative Learning System," *International Journal of Artificial Intelligence in Education (IJAIED),* vol. 12, pp. 40-62, 2001.

[26] M. Daniels, *et al.*, "RUNESTONE, an International Student Collaboration Project," in *IEEE Frontiers in Education Conference*, Tempe, Arizona, 1998.

[27] 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.

[28] 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.

[29] E. O. Navarro and A. v. d. Hoek, "SimSE: an educational simulation game for teaching the Software engineering process," *SIGCSE Bull.,* vol. 36, pp. 233-233, 2004.

[30] L. Wilkerson and G. Feletti, "Problem-based learning: One approach to increasing student participation," *New Directions for Teaching and Learning,* pp. 51-60, 1989.

[31] "Guide to the Software Engineering Body of Knowledge - SWEBOK," Los Alamitos, California 0-7695-1000-0, 2004.

[32] D. Anawati and A. Craig, "Behavioral Adaptation Within Cross-Cultural Virtual Teams," *IEEE Transactions of proffessional communicacion pc,* vol. 49, pp. 44-56, 2006.

[33] S. Ramingwong and A. S. M. Sajeev, "A Multidimensional Model for Mum Effect in Offshore Outsourcing," presented at the Proceedings of the 2008 2nd IFIP/IEEE International Symposium on Theoretical Aspects of Software Engineering, 2008.

[34] P. Shachaf, "Cultural diversity and information and communication technology impacts on global virtual teams: An exploratory study," *Inf. Manage.,* vol. 45, pp. 131-142, 2008.

[35] T. Clear, "International Collaborative Learning - The Facilitation Process," in *ED-MEDIA '99 - World Conference on Educational Multimedia, Hypermedia and Telecommunications*, Seattle, Washington, 1999, pp. 1759-1764.