A SOFTWARE ARCHITECTURE FOR INTELLIGENT VIRTUAL ENVIRONMENTS APPLIED TO EDUCATION

Angélica de Antonio[1] Jaime Ramírez[2] Ricardo Imbert[3] Gonzalo Méndez[4] Raúl Antonio Aguilar[5]

ABSTRACT

This paper describes the software architecture that has been designed as a model for the application of Intelligent Virtual Environments to training activities. CORBA has been used as the middleware to integrate a graphical and interactive environment developed in OpenGL and Visual C++, with a cooperative multi-agent system developed on top of the JADE platform.

Keywords**:** Virtual Environments, Intelligent Tutoring Systems, Intelligent Virtual Training Environments, Agent Based Systems, Intelligent Virtual Agents

INTRODUCTION

The use of Virtual Environments (VEs) for training is a promising option for educational activities, especially in those situations where traditional education can be costly, dangerous or even impossible to realize.

The combination of VEs with Intelligent Systems, and mainly Agent-based Systems, is giving rise to a new category of software usually referred to as Intelligent Virtual Environments (IVEs) [1]. A special case of those are the systems that combine a VE with an Intelligent Tutoring System (ITS) and are applied to training. They will be called Intelligent Virtual Environments for Training (IVETs) [9]. In IVETs, the tutoring process embedded in the system sometimes adopts, just like the learners, a virtual representation within the VE. Virtual characters representing the tutoring component of IVETs are usually referred to as Pedagogic Agents [6], given that they are usually designed and architected as intelligent software agents [7].

From the Software Engineering point of view, there are very few methodological proposals for the development of virtual reality-based environments [5,11]. Moreover, very few tools and platforms are available to support the development of IVETs. These are very complex systems in which it is necessary to combine advanced graphical capabilities, interaction management through complex virtual reality devices, distributed multi-user communication, and intelligent components to assist learners in their learning process. The complexity and heterogeneity inherent in IVETs calls for open and flexible architectures [10] that allow a smooth integration of components and subsystems possibly developed with different technologies, as well as improved extensibility and maintainability.

This paper describes the subsystems and the software architecture underlying a platform for the development of IVETs that has been created in the Decoroso Crespo Laboratory at the Universidad Politécnica de Madrid (UPM). This work has been partially funded by the Spanish Ministry of Science and Technology under grant TIC00-1346.

THE MAEVIF PROJECT

The MAEVIF project (Model for the Application of Intelligent Virtual Environments to Education – in Spanish "Modelo para la Aplicación de Entornos Virtuales a la Formación") started on December 2000. The general goal of the project was the definition of a model for the application of IVEs and IVAs to education. This implied several tasks including:

- The definition of a generic model for IVETs.
- The definition of an open and flexible architecture, based on components, to support the previously defined generic model.

[1] Facultad de Informática, Universidad Politécnica de Madrid, Campus Montegancedo, 28660 Boadilla del Monte, Madrid, España. angelica@fi.upm.es
[2] jramirez@fi.upm.es
[3] rimbert@fi.upm.es
[4] gonzalo@gordini.fi.upm.es
[5] raguilar@bermudas.fi.upm.es

- The development of a prototype platform and an experimental IVET to test the model and architecture.

This paper focuses on the software architecture and its current implementation and the way in which various heterogeneous components and technologies have been integrated to provide a common ground for the development of present and future IVETs.

MAEVIF is basically composed of two subsystems. The first one deals with the graphical visualization of the virtual environments and the interaction with the learners. The second subsystem is a multi-agent system designed to provide "intelligence" to the tutoring system. The following sections describe both subsystems in detail and the way in which these quite different approaches to software have been integrated through a standard for distributed objects.

A GRAPHICS AND INTERACTION SUBSYSTEM FOR IVETS

The generic model for IVETs that has been considered is oriented towards training applications in which one or more learners have to learn to interact with the surrounding environment and between them, in the execution of a cooperative task with a shared goal. The need for the learners to interact with the environment as part of their task leads to and justifies the use of a virtual reality system. Virtual reality technologies range from the most expensive and immersive devices, such as CAVEs, head mounted displays (HMDs), motion tracking systems, or data gloves, to the more economical but least immersive interaction means commonly used in desktop configurations (mouse, monitor, joystick and keyboard). The objective was to select an approach that was valid for any logical combination of interaction devices, being open to the needs and possibilities of many different types of users. The more immersive is the combination of interaction devices available for the learner, the more realistic will be the learning experience.

For the development of the graphics and interaction subsystem (GIS) to be used by the learners, a variety of technologies are required. Graphics technologies are necessary for the creation and rendering of the scene to be perceived by each learner during the training activities. Each learner will be represented in the environment by a virtual character, an avatar, so that, in every moment, the view that the learner has over the environment will correspond to what his/her avatar "sees" through its virtual eyes. The environment, the objects within it, and the avatars, have to be created as three dimensional models, trying to find a proper balance between realism and simplicity for real-time rendering. In the MAEVIF prototype, 3D Studio Max has been used for this purpose, but similar tools could also be used. The resulting models need to be translated into a textual file format that has been defined in MAEVIF. A translator for 3D Studio Max has already been implemented. If other 3D modeling tools were to be used, new specific translators would have to be developed.

The graphics and interaction subsystem is able to load, render and animate any environment and its objects, provided that they are in the defined MAEVIF format, using the OpenGL graphic libraries. This subsystem has been implemented in Microsoft Visual C++, and it also makes use of some of the Microsoft DirectX libraries. In particular, DirectPlay is used for direct communication between the different learner applications that are connected to the same environment in any given moment. In this way, with a peer-to-peer approach, the movements of the avatars that represent the learners are communicated to the other learner views, in order to update their corresponding rendered scenes. Microsoft's DirectInput has also been used to manage conventional interaction devices (keyboard and joystick). For more advanced virtual reality devices, special libraries have been designed to act as intermediaries and avoid a direct coupling of the subsystem with the specific drivers provided by the devices' vendors. Finally, the GIS also makes use of IBM's Via Voice to allow voice synthesis and recognition. Just as it was the case with the creation of 3D models and animations, the combination of languages, libraries and tools that has been selected for the implementation of the GIS could be replaced by other analogous technologies, such as Java (instead of C++), Java3D (instead of OpenGL), Dragon NaturallySpeaking or VoiceXpress (instead of ViaVoice), OpenPlay (instead of DirectPlay), etc.

Figure 1 shows the different packages that compose the GIS and their interdependencies. They will be explained in the following paragraphs.

Scenario Package. It deals with the rendering and continuous update of the virtual world. It also captures all events coming to the GIS from the learner and the operating system (timers, etc.) and retransmits these events to the Manager Package. MFC (*Microsoft Foundation Classes*) classes are used to implement a graphical user interface (GUI) that allows the learner to interact with the system.

Devices Package. It manages the initialization, polling and interfacing with the different interaction devices offered to the learners. It is the package that encapsulates and hides the details of the specific device drivers.

Communications Package. It has the responsibility of coordinating the different views that different learners have over the same virtual environment, depending on the position of their avatars in the world. It is the package that makes use of the DirectPlay libraries. This package is also responsible of the communication with the multi-agent tutoring subsystem, via a CORBA middleware.

World Package. Its classes maintain geometrical information about all the objects in the environment.

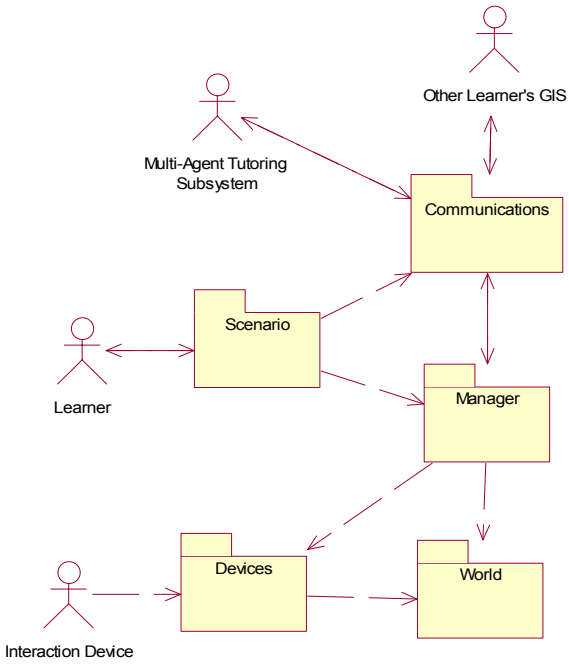

Fig. 1. Packages in the graphics and interaction subsystem of MAEVIF

Manager Package. It is the core of the GIS. Using the classes in the Scenario, Devices and Communications Packages, gathers and interprets all the events that have an effect on the virtual world, which is updated via the classes in the World Package.

The ultimate goal of conventional and/or not conventional interaction devices in an IVET is allowing the learner to *navigate* through the environment and to *interact* with the objects and other entities in the world. These interactions will have effects on the environment and system that is being simulated. For instance, if the virtual environment reproduces the control room of a nuclear power plant, the push of a button by the learner could provoke the opening of a valve and the change in the color of an indicator. Consequently, the IVET should maintain knowledge about the interaction possibilities and their effects on the visual representation of the environment and on the underlying simulated system. This knowledge will also be used by the multi-agent tutoring subsystem to build action plans, provide explanations, answer questions coming from the learner, and other processes.

In order to manage navigation, three generic variables have been defined: position, orientation and viewpoint of the learner's avatar. In order to manage interaction, two additional variables are considered: position and orientation of the avatar's hand. Each of these variables can be controlled by different types of device inputs. The *Manager Package* is in charge of making the translation from the inputs that come from the devices to the pertinent modifications of the variables.

However, even if MAEVIF is open to almost any combination of devices, there are some possibilities that do not make sense and are not allowed. For instance, if a learner wishes to use a HMD to visualize the environment and a motion tracking system to determine his/her head's position and orientation, it would be cumbersome to use a keyboard to generate other events, such as actions on the objects; first, because he/she would not see the keyboard, and second, because he/she would be physically moving around to navigate through the environment.

In the MAEVIF prototype, the selection of a navigation device conditions the possible devices that are allowed for other actions. Table 1 shows the feasible interaction options that are currently supported by MAEVIF.

Table 1. Feasible Combinations of Interaction Devices.

Navigation	Viewpoint			Hand	
	HMD (with position sensor)	Joystick	Keyboard	Data glove	Keyboard
HMD	✓			✓	
Joystick	✓	✓		✓	✓
Keyboard	✓		✓	✓	✓

As it can be seen in figure 2, regardless of the selected devices, the view that a learner gets from the environment in a training session is a first person view. The learner will observe a 3D environment with objects (some of them can be manipulated and others cannot), avatars representing other learners, and a special virtual character that represents his/her virtual tutor. He/she will also see the right hand of his/her own avatar, allowing him/her to point to virtual objects (to ask a question about them to his/her virtual tutor, for instance) and to interact with them (to pick up an object, to push a button, to open a door, etc.). The manipulable objects and the interaction possibilities will be dependent on the particular environment that is being simulated in the IVET. This information can be associated to the different IVETs in the MAEVIF platform by means of an authoring tool. Several XML files are generated by this tool containing this information.

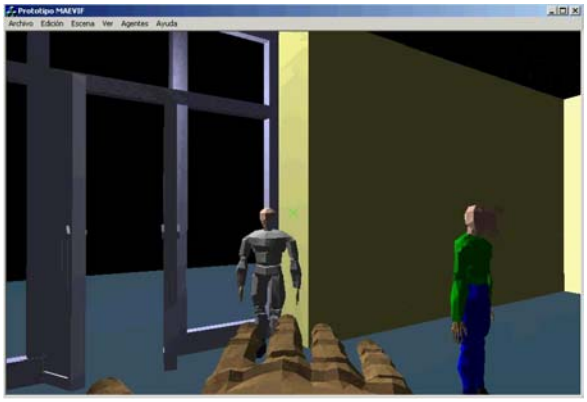

Fig. 2. View that a learner has of the virtual environment in which he/she is immersed

AN AGENT-BASED ARCHITECTURE FOR THE TUTORING SUBSYSTEM

The decision of using multi-agent technology for the design of the tutoring subsystem relies on the belief that that the design of highly interactive IVETs populated by intelligent and autonomous or semi-autonomous entities, in addition to one or more avatars controlled by users, requires highest level software abstractions. Objects and components (CORBA or COM-like components) are passive software entities which are not able to exhibit the kind of pro-activity and reactivity that is required in highly interactive environments. Agents, moreover, are less dependent on other components than objects. An agent that provides a given service can be replaced by any other agent providing the same service, or they can even co-exist, without having to recompile or even to reinitiate the system. New agents can be added dynamically providing new functionalities. Extensibility is one of the most powerful features of agent-based systems. The way in which agents are designed make them also easier to be reused than objects.

The first step in the design of the Multi-Agent Tutoring Subsystem (MATS) was the reinterpretation of the traditional architecture for ITSs (composed of Expert, Student, Tutoring and Communication modules) as an architecture based on cooperative agents. The first problem was that the traditional ITS architecture was meant for a single learner interacting with his/her tutor, while MAEVIF was going to allow distributed multiple learners inhabiting the same environment and interacting with the environment, between them, and with their virtual tutors. On the other hand, the 3D environment and its inhabitants could not be considered any more as an external Communication Module through which the student interfaces with his/her tutor. The students and the tutor are in fact immersed and they are part of the environment. The complex network of possible interactions together with the fact that the students become a part of the interface, lead us to a modification in the original ITS architecture to count for multiple learners and immersive 3D environments. The Student Module was split into several Individual Learner Modules and one Team Module. An additional module was also defined, called World Module that should contain geometric and semantic information about the environment and its objects, its inhabitants, and its interaction possibilities.

Afterwards, a representative agent for each of the five modules in the extended ITS architecture was designed:

- Expert Agent
- Tutoring Agent
- Communication Agent
- Student Modeling Agent
- World Agent

Then, the responsibilities for the representative agents were defined, as well as the interaction and communication scenarios between them. Some subordinate agents were then associated to the representative agents. They can delegate some of their responsibilities and specific tasks on these subordinate agents. The resulting multi-agent architecture is described in [2].

For the implementation of this multi-agent architecture, the JADE platform (*Java Agent Development Framework*) has been used. JADE sticks to the FIPA (*Foundation for Intelligent Physical Agents*) specifications for the development of intelligent software agents. Individual agents are programmed in Java. Here, again, other agent platforms could have been used (FIPA-OS, Zeus, etc.).

An incremental and evolutionary approach to development has been selected for MAEVIF. The current prototype implements a subset of the complete architecture that can be seen in figure 3. The functionality of these agents is briefly described next.

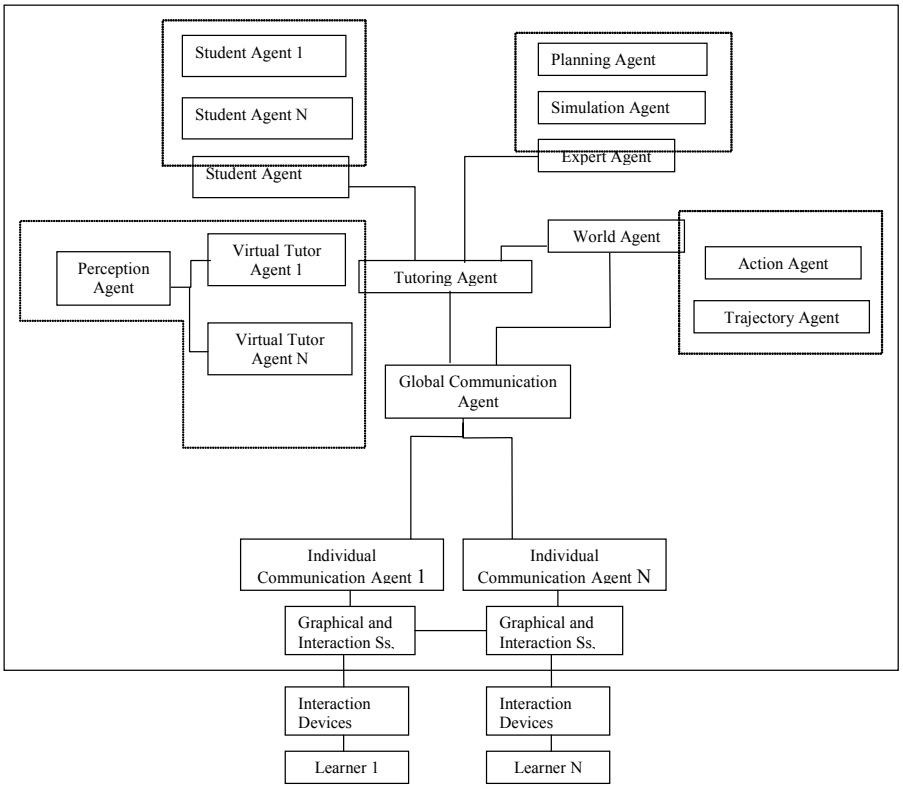

Fig. 3. MAEVIF Multi-agent Architecture

Initiation Agent. It is an auxiliary agent responsible for the creation of the necessary objects and containers upon the initiation of the system, after the agent server has been launched.

Global Communication Agent. It is in charge of the channeling the communication between the MATS and the different GIS running on the learners' machines.

Individual Communication Agents. Each learner, upon connecting to the system, will be assigned an Individual Communication Agent in charge of transmitting messages from the learner's GIS to the MATS and vice versa. For instance, it will inform the MATS of the virtual character that has been selected by a learner in a certain activity and of the actions that the learner is trying to execute, and it will communicate to the GIS the verbal messages to be synthesized coming from his/her virtual tutor (clues, answers to the student's questions, etc.), the animations to be performed by the virtual tutor, etc.

Notification Agent. Given that JADE agents can only send to and receive messages from other agents, the communication between the GIS and the MATS had to be performed using a temporal agent that is created whenever a message has to be sent to the MATS and is destroyed afterwards.

Tutoring Agent. It is responsible for proposing activities to the learners, monitoring their actions in the virtual environment, checking if they are valid or not, and making tutoring decisions. The activities that can be proposed by the Tutoring Agent are dependent on the particular environment that is being simulated in the IVET, and they can also be defined through the authoring tool. Some XML files define the activities in the IVET, the characters that should take part on them, and the role to be performed by each character.

Expert Agent. The expert agents will contain the expert knowledge about the system that is being simulated, as well as the expert knowledge necessary to solve problems and to reach the desired goals. Most of the activities to be posed to the learners, in the generic model of an IVET that is being considered, will consist of finding an appropriate sequence of actions to go from an initial state of the simulated system and the environment to a desired final state. These actions will have to be executed by the team of learners. The expert agent will delegate on a *Simulation Agent*, that contains the knowledge about the simulated system, a *Planning Agent*, that is able to find the best sequence of actions to solve different activities. The Simulation Agent in the prototype IVET was so simple that their functions are currently performed by the Expert Agent. The Trajectory Agent is currently able to compare the paths followed by the learners with predefined trajectories (stored in XML files and defined with the authoring tool) and an algorithm to compute best paths, which is based on the A* algorithm, is being implemented.

World Agent. It maintains information about all the virtual objects, manipulable or not (name, position, size), as well as about all the learners immersed in the environment (name, position, orientation, carried objects in their inventory, virtual character selected by the student for the activity,...). This agent is able to verify conditions on the state of the environment, and update this state according to the consequences of the actions executed by the learners.

Action Agent. It owns the knowledge about the actions that can be performed on the virtual objects in the environment. It cooperates with the Planning Agent (by means of a blackboard structure) to build solution plans. During the simulation of an activity, it is responsible of checking if the preconditions associated to the actions that a learner is trying to execute really hold, and it is also responsible of coordinating the execution of all the consequences associated to those actions (in cooperation with other agents).

Trajectory Agent. It is able to find the best path to go from a certain position in the environment to a selected destination and to compare the best trajectory with the path followed by a learner, communicating the results to the Tutoring Agent.

Perception Agent. A model for visual perception in virtual intelligent agents [4] has been implemented. This model allows the virtual tutors to perceive the surrounding environment in a way inspired by the human perceptual system. The Perception Agent receives from the World Agent information about the position and size of the objects in the environment (maintaining some geometric entities called *nimbus*) and about the position and viewpoint of the virtual tutors (maintaining other geometric entities called *focus*). Using these entities, the Perception Agent is able to calculate what a virtual tutor can see in a given moment and with which clarity of perception [3].

Virtual Tutor Agent. This agent is responsible for controlling the actions of the characters associated to the virtual tutors. Each learner will have his/her own virtual tutor. In the current prototype virtual tutors only have a simple behavior of following the learners in their way through the environment, keeping at a proper distance to observe their actions.

Student Agents. These agents are in charge of maintaining a model of each learner, including personal information, their actions in training sessions, and a model of the learners' knowledge. In the current prototype the student model is maintained in a combination of XML files and a database.

A DISTRIBUTED SOFTWARE ARCHITECTURE

The subsystems integrating MAEVIF software architecture have developed with heterogeneous technologies and languages. The goal was to benefit from the specific features of these technologies according to the required functionalities. On one there is a multi-agent subsystem that benefits from the features offered by the JADE platform. On the other hand there is a graphics and interaction subsystem that benefits from different libraries and drivers available for the management of graphics and interaction devices.

This section details the way in which both subsystems have been integrated in an architectural solution that is open and flexible. CORBA (*Common Object Request Broker Architecture*), one of the most used standards for the development of distributed components, has been used for this purpose.

CORBA provides a specification for the management of distributed objects, by means of a bus (*Object Request Broker: ORB*) through which objects written in different languages and residing on different networks and operating systems, can interoperate [4]. Of the variety of services offered by CORBA, MAEVIF relies mainly on its Name Service, which allows linking an object with a name in order to be able to locate the object by its name afterwards. Figure 4 shows the subsystems in MAEVIF's system architecture.

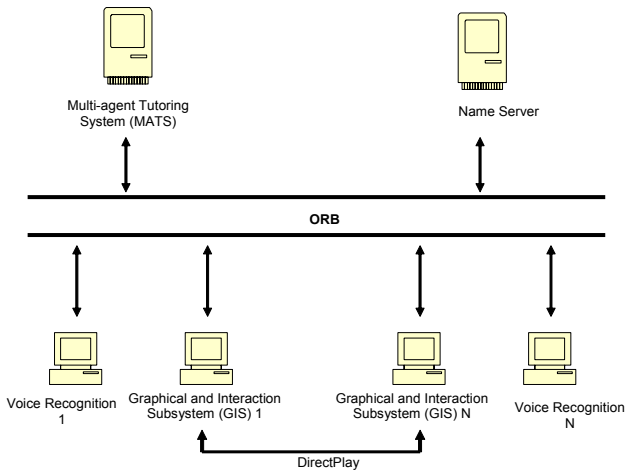

Fig. 4. MAEVIF's System Architecture

MAEVIF is a multi-user system, therefore the CORBA objects cannot be assigned a specific name, because that could lead to name conflicts between the objects created for different learners. To solve this problem, a Name Server developed by Orbacus that allows the definition of hierarchical name contexts has been used. Figure 5 shows the context structure defined for MAEVIF.

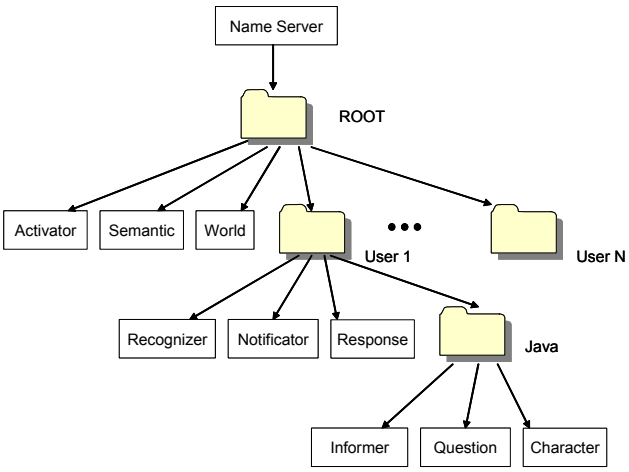

Fig. 5. Name contexts structure in MAEVIF

CORBA objects for MAEVIF fall into two categories:

- *Servers*. These are objects that offer a set of well defined services to other CORBA objects. In MAEVIF there are global servers as well as specific servers for each learner. Table 2 shows the server objects in MAEVIF.
- *Clients*. These are objects that are in charge of maintaining references to server objects registered in the name server, sending requests to them, receiving the responses and providing the answers to the associated subsystem making transparent the fact that server components are distributed. MAEVIF client objects are:
    - Associated to the GIS: Activator, Informer, Character.
    - Associated to the MATS: World, Semantic, Notificator, Response.
    - Associated to the Voice Engine: Recognizer, Semantic, Informer, Question.

Table 2. Server objects registered in the MAEVIF name server

Subsystem	Server	Object	Description
GIS	Global	World	It is an access point to consult information about the objects in the virtual environment.
		Semantic	It manages all the information about manipulable objects in the virtual environment.
	Learner	Response	It informs the virtual environment about the answers to the questions formulated by the learner through the voice recognition system.
		Recognizer	It allows the communication between the GIS and the voice recognition system.
		Notificator	It provides access to the GIS associated to an individual learner for the MATS to communicate modifications on it.
MATS	Global	Activator	It manages the connections of learners and synchronizes the MATS with the first GIS that is activated.
	Learner	Informer	It manages the learner's data and informs the MATS about the learner's actions in the virtual world.
		Question	It sends to the MATS the questions formulated by the learner.
		Character	It informs the MATS about the character selected by the learner among the available characters in one activity..

Generally speaking the procedure to initiate a training session in MAEVIF includes the following steps:

1. Initiate the Orbacus name server in one machine and port.
2. Initiate the JADE platform with the necessary agents for the session, in the same or in other machine.
3. Open an instance of the GIS in each of the machines that will be used by the team of learners.
4. Activate the voice recognition system in the machines of those learners that want to use it as an interaction device.

CONCLUSIONS AND FUTURE WORK

A system and software architecture has been devised to facilitate the development of Intelligent Virtual Training Environments (IVETs). This architecture provides the ground for integrating many heterogeneous technologies and making them interoperate in a distributed setting. The architecture has been devised to be open and flexible, and to facilitate the incorporation of new components and agents, as well as the use of different languages and tools.

New agents and new functionalities for the existing agents are currently being implemented. For the future, it would be desirable to improve the effectiveness of the IVET through enhanced capabilities of the Pedagogic Agents. These agents should be perceived by the learners as believable intelligent entities. An open research line focuses on the study of the learner-IVET interaction and the search for more efficient and

effective metaphors and navigation and interaction mechanisms.

ACKNOWLEDGEMENTS

This work has been partly funded by the Spanish Ministry of Science and Technology through project MAEVIF (TIC00-1346).

REFERENCES

[1] Aylett, R. & Luck, M.: Applying Artificial Intelligence to Virtual Reality: Intelligent Virtual Environments. Applied Artificial Intelligence 14 (1). (2000) pp. 3-32.

[2] de Antonio, A., Ramírez, J., Méndez, G.: An Agent-based architecture for virtual environments for training. In M.I. Sánchez (ed). Developing Future Interactive Systems. IDEA Group. (2004)

[3] Hermida, B.: PEVE: Percepción Visual en Entornos Virtuales para Entrenamiento. Trabajo Fin de Carrera. Facultad de Informática, Universidad Politécnica de Madrid. (2004)

[4] Herrero, P., de Antonio, A.: Diseño de uno Modelo de Percepción para Agentes Virtuales Inteligentes Basados en el Sistemas de Percepción de los Seres Humanos. Revista de la Facultad de Ingeniería, U.T.A. Vol. 11, No. 1., Arica, Chile (2003) pp. 11-24.

[5] Fencott, C.: Towards a Design Methodology for Virtual Environments. Workshop on User Centered Design and Implementation of Virtual Environments. University of York. (1999)

[6] Johnson, W.; Rickel, J. & Lester, J.: Animated Pedagogical Agents: Face-to-Face Interaction in Interactive Learning Environments. International Journal of AIED (2000)

[7] Giraffa, L. & Viccari, R.: Intelligent Tutoring Systems Built Using Agents Techniques. La Salle, Revista de Educación, Ciencia y Cultura. 4 (1), Canoas: Brasil (1999)

[8] López, G., Salas, M., Siles, R., Soriano, F.J.: Arquitectura de Objetos Distribuidos CORBA. Facultad de Informática. Universidad Politécnica de Madrid. (2000)

[9] Méndez, G., de Antonio, A., Herrero, P.: Prvir: An Integration Between an Intelligent Tutoring System and A Virtual Environmente.In SCI 2001.Volume VIII. Orlando, FL, IIIS, IEEE Computer Society (2001) pp. 175-180.

[10] Munro, A., Surmon, D.S., Johnson, M.C., Pizzini, Q.A., & Walker, J.P.: An Open Architecture for Simulation-Centered Tutors. Open Learning Environments: New Computational Technologies to Support Learning, Exploration and Collaboration. 9th Conference on AIDED, Le Mans, France. (1999) pp. 360-367.

[11] Sánchez, M.I.: Una Aproximación Metodológica al Desarrollo de Entornos Virtuales. Tesis Doctoral. Facultad de Informática, Universidad Politécnica de Madrid. (2001)