Using Intelligent Agents to Support Collaborative Virtual Environments for Training

GONZALO MENDEZ, ANGELICA DE ANTONIO
Dpto. Lenguajes y Sistemas Informaticos e Ingenieria del Software
Universidad Politecnica de Madrid
Campus de Montegancedo s/n
Boadilla del Monte, Madrid, SPAIN
gonzalo@gordini.ls.fi.upm.es, angelica@fi.upm.es

ABSTRACT
During the last years, Intelligent Virtual Environments for Training have become a quite popular application of computer science to education, and are often used to allow students to experience situations that would be difficult, costly, or impossible in the real world.

These systems involve very different technologies, ranging from computer graphics to artificial intelligence. However, little attention has been paid to software engineering issues, and most of these systems are developed in an ad-hoc way that does not allow the reuse of their components or even an easy modification of the application.

We describe an agent-based software architecture for the tutoring side of the application, which is intended to be easily extended and modified. In addition, we show how this architecture has been integrated with a Virtual Environment to support realistic training. To encourage the use of this infrastructure, an authoring tool has been develop to aid human tutors to create new training courses. Finally, some experiments to test the suitability of the architecture are shown.

KEY WORDS
Intelligent Agents, Software Architecture, Virtual Environment, Training

1 Introduction

Many of the advances in the application of intelligent agents to the field of Intelligent Virtual Environments for Training (IVET) have come from the Artificial Intelligence community, such as Herman the Bug [1], Cosmo [2] or Steve [3, 4].

However, little effort has been devoted to software engineering issues, and in the few cases where some attention has been paid to design methods, such as in Jacob [5], they have focused in object oriented design rather than agent oriented design.

The MAEVIF (*Model for the Application of Intelligent Virtual Environments to Education*) project is the result of several experiences integrating virtual environments and intelligent tutors [6, 7] that served to point out the problems that commonly arise in such integrations. The objective of the MAEVIF project was to define a model for the application of intelligent virtual environments to education and training, which involved:

- The definition of a generic model for intelligent learning environments based on the use of virtual worlds.
- The definition of an open and flexible agent-based software architecture to support the generic model of an IVET.
- The design and implementation of a prototype authoring tool that simplifies the development of IVETs, based on the defined architecture.
- The definition of a set of methodological recommendations for the development of IVETs.

In the remainder of this paper we will be briefly describe the architecture of an Intelligent Tutoring System (ITS) (section 2) and how it has been extended to support Virtual Environments (section 3). Then, we will show how it has been transformed into an agent-based architecture (section 4). In section 5, an explanation of the functionality of the authoring tool will be given. Section 6 will present a discussion of the results that have been achieved with the MAEVIF project, and finally, in section 7, some future work lines will be shown.

2 Structure of an ITS

An ITS is a software application that makes use or Artificial Intelligence to provide support to a student in a learning environment [8]. It consists of four modules, each of which provides a very precise functionality: tutoring module, expert module, student's module and communication module, as originally described in [9] (see Fig. 1).

The *Expert Module* contains the knowledge about the subject to be taught to the student, and it is the base for the analysis of the answers provided by the student to the tutor's questions. This knowledge is divided into informative concepts, which are small pieces of information, and the knowledge necessary to solve the exercises. The expert

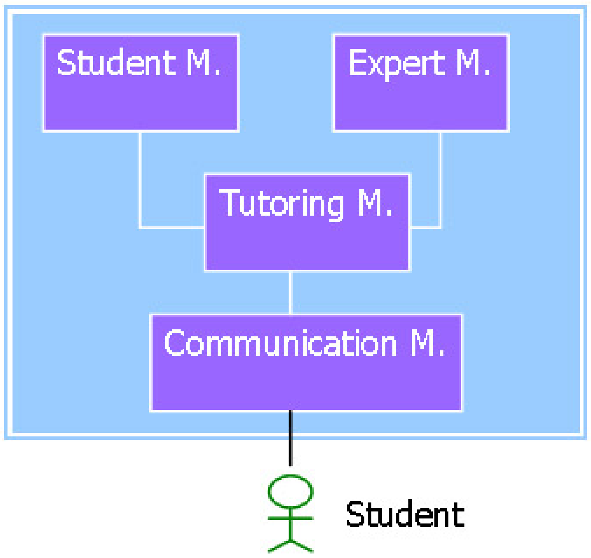

Figure 1. Architecture of an ITS

module must be designed in a way that information is easily accessible and modifiable. The way we have overcome this problem is saving all the information in a relational database. Each informative concept points to other concepts that must be shown necessarily before or after it, as they are all part of a particular block of concepts. Each block, in turn, points to other blocks that must be shown necessarily before or after it within a given module.

The *Tutoring Module* contains the pedagogic knowledge, and is in charge of selecting the appropriate concepts to be shown in the course. It also has the strategies, rules and processes needed to drive the interactions between the student and the system, in order to make decisions about the concepts to teach and the exercises to be done by the student, along with the moment when he must be interrupted in order to correct him or make a suggestion. In addition, it decides when it is appropriate to end showing informative concepts and start with an evaluation. At the end of each block of concepts, there is a bunch of exercises related to the concepts explained in that block, so the student can test his recently acquired knowledge. There is also an evaluation at the end of each module, where exercises from all the blocks that form that module will be chosen.

The *Student's Module* keeps individualized information about every student taking the course. It is responsible for tracing what informative concepts have already been taught to the student, how many exercises he has done and the degree of success and time he has used to complete them. To measure the student's progress, we need some metrics with which to compare what are the minimum and average levels for a student to pass to the next level.

The *Communication Module* is in charge of the communication between the student who is taking the course and the ITS. This module must inform the tutoring module about the actions that are performed by the student all along the course. These actions may be the visualization of an informative concept, the answering to an exercise in any of the forms that it may adopt, or any of the actions performed inside the VE. The communication module must make use of all the available multimedia resources in order to make the course as easy-going as possible, but ensuring it does not make the course be too slow, which in the end could bore the student.

Each of the described modules has a very important role to play in the correct operation of the ITS.

3 An Extension to the Architecture of Intelligent Tutoring Systems

The development of three dimensional Virtual Environments (VEs) has a quite short history, dating from the beginning of the 90s. The youth of the field, together with the complexity and variety of the technologies involved, have led to a situation in which neither the architectures nor the development processes have been standardized yet. Therefore, almost every new system is developed from scratch, in an ad-hoc way, with very specific solutions and monolithic architectures, and in many cases forgetting the principles and techniques of the Software Engineering discipline [10]. Some of the proposed architectures deal only partially with the problem, since they are centered on a specific aspect like the visualization of the VE [11, 12] or the interaction devices and hardware [13]. When we get to IVETs, the situation is even worse.

Our approach to the definition of an architecture for IVETs is based on the agent paradigm. The rationale behind this choice is our belief 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 higher level software abstractions. Objects and components are passive software entities which are not able to exhibit the kind of proactivity 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 coexist. 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.

Starting from the idea that an IVET can be seen as a special kind of ITS, and the pedagogical agent in an IVET can be seen as an embodiment of the tutoring module of an ITS, our first approach towards defining an standard architecture for IVETs was to define an agent for each of the four modules of the generic architecture of an ITS [9] (see Fig. 1).

The ITS architecture, however, does not fit well with the requirements of IVETs in several aspects:

- IVETs are usually populated by more than one stu-

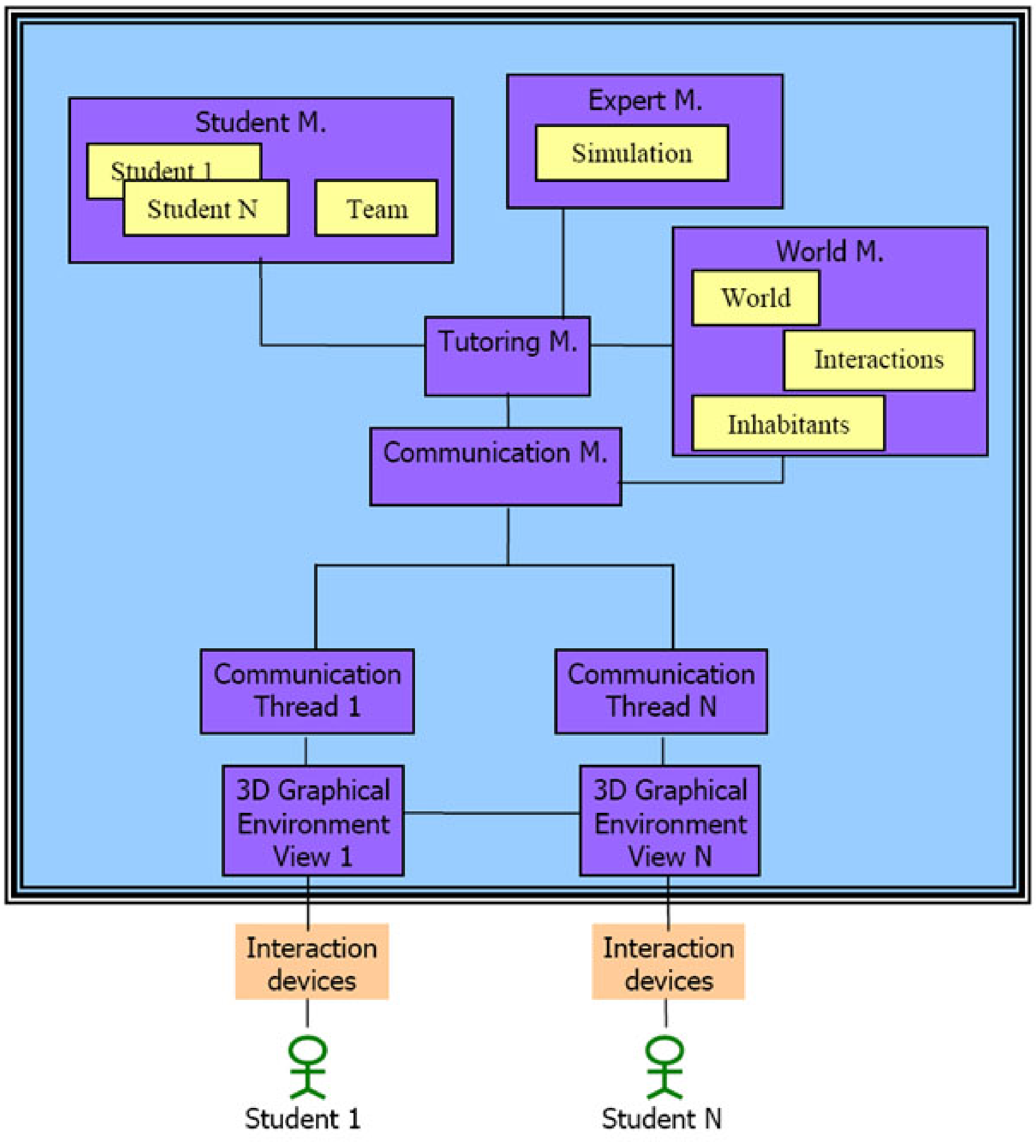

Figure 2. Extended ITS architecture

dent, and they are frequently used for team training. An ITS is intended to adapt the teaching and learning process to the needs of every individual student, but they are supposed to interact with the system one at a time. However, in a multi-student IVET, the system has to adapt both to the characteristics of each individual student and to the characteristics of the team. Consequently, the student module should model the knowledge of each individual student but also the collective knowledge of the team.

- The student is not really out of the limits of the ITS, but immersed in it. The student interacts with the IVET by manipulating an avatar within the IVET, possibly using complex virtual reality devices. Furthermore, each student has a different view of the VE depending on their location within it.

- The communication module in an ITS is usually realized by means of a GUI or a natural language interface that allows the student to communicate with the system. It would be quite intuitive to consider that the 3D graphical model is the communication module of an IVET. However, there is a fundamental difference among them: in an IVET, the learning goals are directly related to the manipulation and interaction with the 3D environment, while the communication module of a classical ITS is just a means, not an end. Therefore, the ITS needs to have explicit knowledge about the 3D VE, its state, and the possibilities of interaction within it.

As a first step we decided to modify and extend the ITS architecture by considering some additional modules (see Fig. 2). First of all, we split the communication module into a set of different views for all the students with a particular communication thread for each student, and a centralized communication module to integrate the different communication threads. Then, we added a world module, which contains geometrical and semantic information about the 3D graphical representation of the VE and its inhabitants, as well as information about the interaction possibilities. The tutoring module is unique to be able to make decisions that affect all the students, as well as specific tutoring decisions for a certain student. The expert module contains all the necessary data and inference rules to maintain a simulation of the behavior of the system that is represented through the VE (e.g. the behavior of a nuclear power plant). The student module, finally, maintains an individual model for each student as well as a model of the team.

4 An Agent-Based Architecture for IVETs

Taking the extended architecture described in the previous section as a starting point, the next step is to decide which software agents are necessary to transform this component-oriented architecture into an agent-oriented architecture, which has been designed using the GAIA methodology [14]. In this methodology, the authors suggest the use of the *organizational metaphor* to design the system architecture, which basically consists of analyzing the real-world organization in order to emulate its structure. It is mentioned that this approach does not always work (depending on particular organization conditions), but in this case, considering the extended architecture of an ITS as the real world, it seems quite appropriate to imitate its structure to develop the system architecture.

Figure 3 shows how the extended ITS architecture is transformed, from a modular point of view, into an agent-based architecture. It has five agents corresponding to the five key modules of the extended ITS architecture:

- A Communication Agent
- A Student Modelling Agent
- A World Agent
- An Expert Agent
- A Tutoring Agent

Each of these agents relate to other subordinate agents, giving rise to a multi-level agent architecture.

4.1 Central Communication Agent

The Communication Agent delegates part of its responsibilities to a set of Individual Communication Agents dedicated to each student. There is also a Connection Manager Agent, which is responsible for coordinating the connections of the students to the distributed system, and a set of Device Agents in charge of managing the data provided by the devices the students use to interact with the Virtual Environment.

4.2 Student Modelling Agent

This agent is in charge of maintaining a model of each student, including personal information, their actions in training sessions, and a model of the students' knowledge. The model of each student will take the form of another agent, a Student Agent, which will reflect, as faithfully as possible, all that is known or inferred about the student.

The Student Modelling Agent is assisted by:

- A Historic Agent, which is responsible for registering the history of interactions among the students and the system.
- A Psychological Agent, which is responsible for building a psychological profile of each student including their learning style, attentiveness, and other personality traits, moods and emotions that may be interesting for adapting the teaching process.
- A Knowledge Modelling Agent, which is responsible for building a model of the student's current knowledge and its evolution.
- A Cognitive Diagnostic Agent, which is responsible for trying to determine the causes of the student's mistakes.

4.3 World Agent

The World Agent is related to:

- The 3D Geometrical Information Agent which has geometrical information on the objects and the inhabitants of the world. Among other responsibilities, this agent will answer questions about the location of the objects.
- The Objects and Inhabitants Information Agent, which has semantic knowledge about the objects and the inhabitants of the world. This agent will be able to answer questions about the utility of the objects or the objects being carried by a student.
- The Interaction Agent, which has knowledge about the possible actions that the students can perform in the environment and the effects of these actions. It will be able to answer questions like "What will it happen if I push this button?"
- The Path-Planning Agent, which is capable of finding paths to reach a destination point in the environment avoiding collisions with other inhabitants and objects. For the purpose of finding these paths, the A* algorithm will be applied to a graph model of the environment.

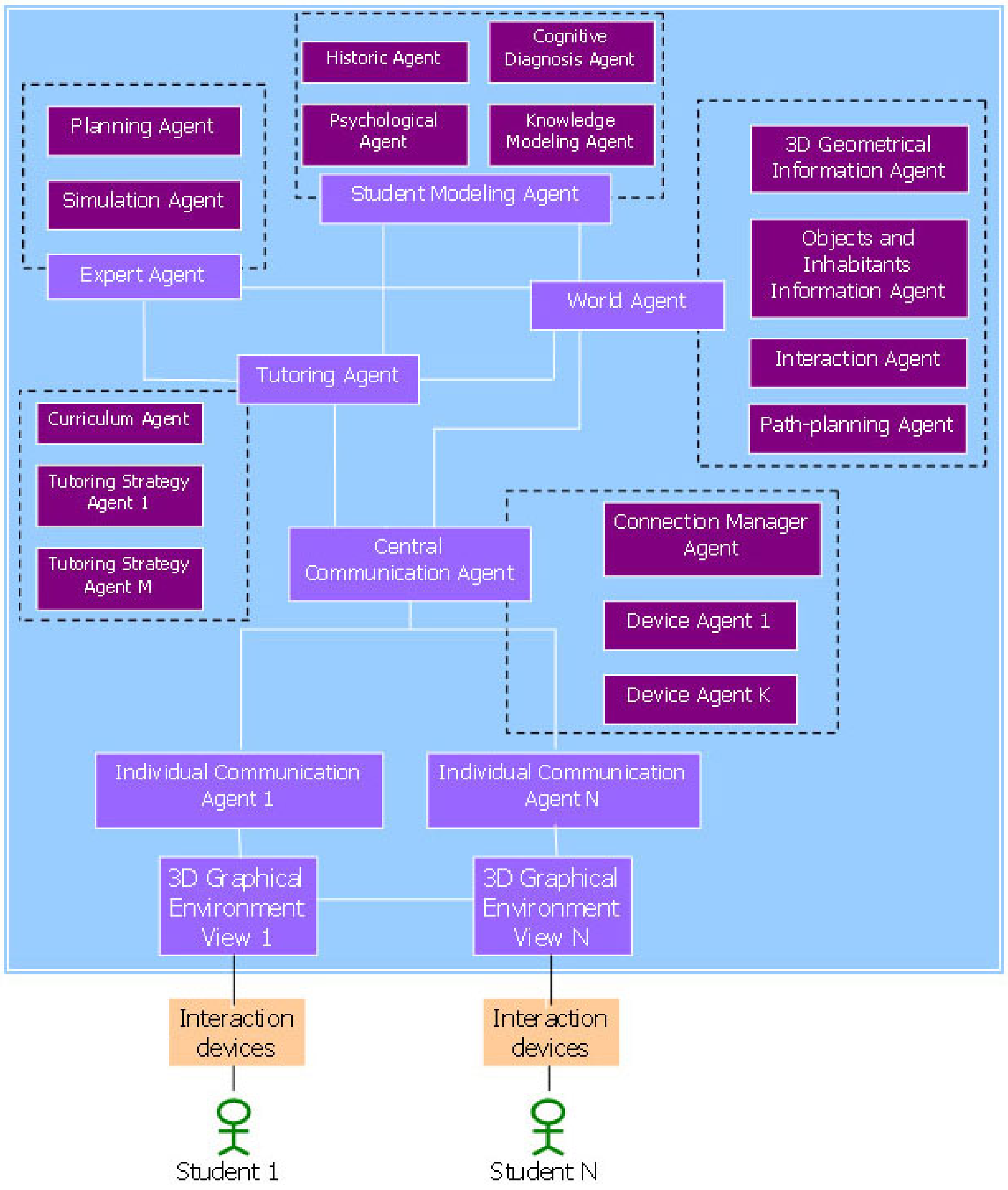

Figure 3. Agent-based architecture

4.4 Expert Agent

The expert agent contains the expert knowledge about the environment that is being simulated, as well as the expert knowledge necessary to solve the problems posed to the student and to reach the desired goals. Most of the activities to be executed by the students consist of finding an appropriate sequence of actions, or plan, to go from an initial state of the environment to a desired final state. These actions have to be executed by the team of students. The Expert Agent delegates some of its responsibilities to a Simulation Agent, that contains the knowledge about the simulated system, and a Planning Agent, that is able to find the best sequence of actions to solve different activities.

The plan for an activity is worked out by the Planning Agent with the collaboration of three other agents:

- The Path-Planning Agent can determine whether there is a trajectory from a certain point of the world to another one.
- The Interaction Agent provides information about the actions that a student can directly execute in the environment.
- The Simulation Agent provides information about some high-level actions that can be executed over the simulated system (e.g., a nuclear power plant). One of these high-level actions will typically require the

execution of one or more student's actions; therefore, a hierarchical planning will be performed. In the nuclear power plant domain, an example of a high-level action may be to raise the reactor's temperature. This high-level action would be decomposed into two student actions, go to the control panel and press the button that closes the input water valve.

4.5 Tutoring Agent

It is responsible for proposing activities to the students, monitoring their actions in the virtual environment, checking if they are valid or not with respect to the plan worked out by the Expert Agent, 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 be defined by means of an authoring tool. Some XML files will define the activities in the IVET, the characters that should take part in them and the role to be performed by each character.

The Tutoring Agent is assisted by a Curriculum Agent, which has knowledge of the curricular structure of the subject matter, and several Tutoring Strategy Agents, which implement different tutoring strategies.

4.6 Integration with the Virtual Environment

The MAEVIF system is a Distributed Virtual Environment that is thought to be used for team training. Thus, the described multiagent system has to support the communication with several students working in different locations.

Each VE client has been developed using C++ and OpenGL, and the multiagent system has been developed using the JADE platform, which is based on the Java programming language and allows for the distribution of the multiagent system in different computers. Among the different options to communicate these two systems, CORBA was chosen as the most promising alternative to achieve this objective.

At the same time, to maintain a coherent representation of the view that each student has of the VE, all the clients have been connected using Microsoft's DirectInput library.

Thus, when a student moves in the VE or interacts with an object or another student, the change in the VE is sent to the rest of the students, whose view is updated to reflect these changes, and it is also sent to the tutoring system, where these changes are processed by the appropriate agents to supervise what the student is doing. The information that is sent to the tutoring system is packed in different CORBA objects, which are read by a communication agent that is in charge of knowing when a change takes place and to communicate it to the appropriate agent (this is a restriction of the JADE platform, which only allows agents to send messages to other agents).

When an action has to be performed by the tutoring system, it is also sent in a CORBA object to the appropriate student. The VE has a well defined interface with which it is possible to interact with it, so the correct method is called from the CORBA object and the action, if necessary, is automatically sent to the rest of the students so that they can see the changes that the tutoring action produces in the VE.

5 Authoring Tool

The described architecture has allowed us to build a basic agent infrastructure that works as a runtime engine. One of the main goals of this architecture is for it to be flexible enough, so it can be used for training in heterogeneous environments without having to extensively modify it.

This can be done by changing the knowledge and goals that the agents have according to the different training needs. To ease this task, an authoring tool has been developed to help human tutors to design new training courses.

The authoring tool allows the human tutor to load an existing 3D environment in which the training process will take place. He can then select the objects with which the students will be able to interact, and he can define the different actions that can be carried out with each object (e.g. take, drop, use, open, put on...) and all the aspects related to those actions (e.g. preconditions, postconditions, parameters, animations that must be shown...).

Subsequently, the author can create new training activities. To do this, he has to decide how many students have to take part in the activity, what their initial positions are in the virtual environment, what goals they must achieve, and the initial state of the world. In turn, some variables of the initial state will be generated randomly every time the students have to train that activity, so that they can solve the same problem starting from different situations.

As a prototype application of our tool we have developed a training system for nuclear power plants operators. We had previously developed this system from scratch, during a one year period. The re-development using our infrastructure has taken a few weeks, and the achieved functionality is superior

6 Discussion

All along the design and development of the described architecture, one of the aspects that has had a bigger impact on it has been the planning process, since a change in the planning method or in the way that knowledge is represented may imply changes in all the agents that take part in it. At the beginning, a simple STRIPS (*STanford Research Institute Problem Solver*) planner [15] was implemented, but we are currently working on the utilisation of a new planner based on Shop2 (*Simple Hierarchical Ordered Planner 2*) [16] or LPG (*Local search in Planning Graphs*)

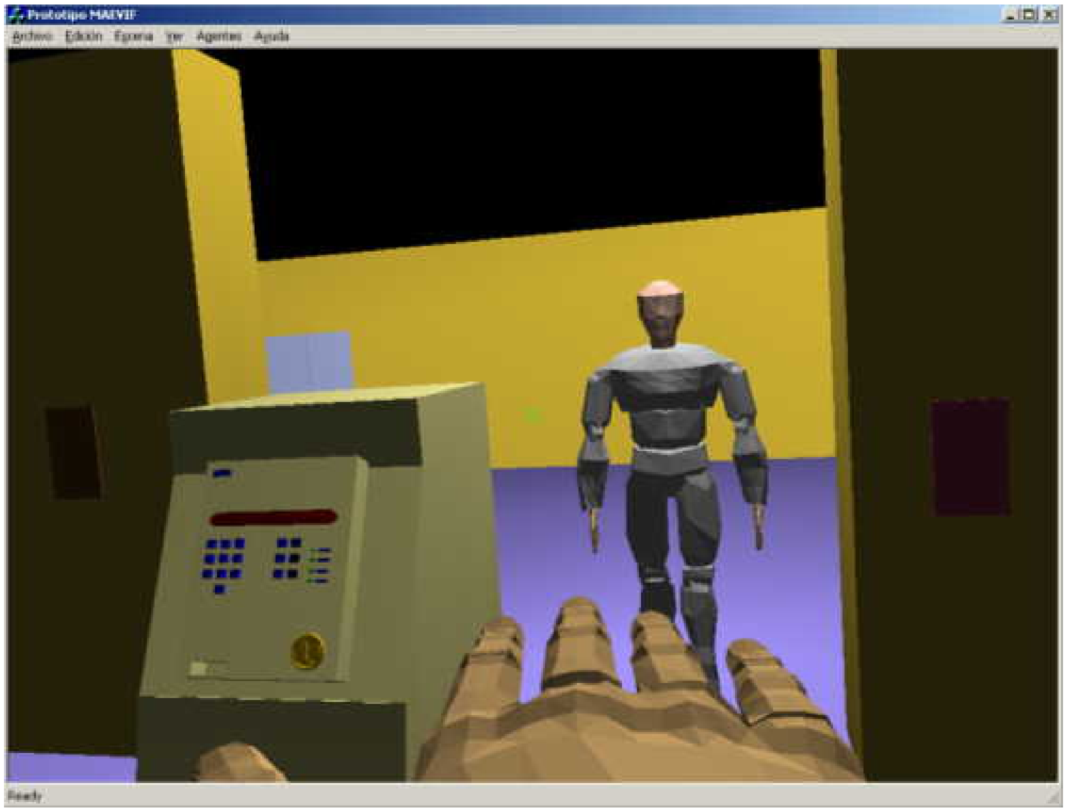

Figure 4. The MAEVIF application

[17]. This change involves the substitution of the planning agent, but it may cause changes in the Interaction, Simulation and Path-Planning agents, which also take part in the planning task.

Another aspect we have tested is how easy it is to add new functionality to the IVET. To do this, we have added an embodied tutor whose goal is to observe what happens in the VE and follow the student to supervise him. It has been necessary to add two new agents: the Virtual Tutor Agent, whose responsibility is to control the tutor's avatar, and the Perception Agent, who is in charge of monitoring the events of the VE.

It has been quite easy to make these changes, since the Perception agent can ask the World Agent for the information it needs and, according to this information, the Virtual Tutor agent can decide how to follow the student and send commands to its 3D representation through the communication agents. Neither the World agent nor the Communication agent have needed further changes.

Finally, we have tested the difficulty of using the described system in a completely different environment, and even with a different purpose. We have designed an experiment where a group of zebras have to drink water in a river, trying to avoid being eaten by a lion, but also trying not to die of thirst.

In this case, the Perception and Virtual Tutor Agents are in charge of controlling the zebras and lions, and the Tutor Agent is responsible for deciding what to do according to their state of thirst and hunger, assisted by the Planning Agent. Some of the existing agents, such as the Simulation Agent, have been removed, since their functionality was not required.

However, some of the agents play a role that is significantly different than the one they were originally thought to play, so if they are to be used in such a way, the architecture will probably have to be modified.

As a result, we can conclude that the architecture has successfully supported the experiments, and has proven to be flexible and extensible enough to allow changes and extensions without having to be redesigned.

7 Future Work

As it has been mentioned in the previous section, one of the elements that can affect more deeply the system architecture is the planning process. In addition, the STRIPS planning algorithm has been used as a test bed for the Planning agent, but it lacks a lot of features that would be desirable in an IVET, such as arithmetic operations or concurrent actions. Therefore, other planning algorithms are being evaluated, because of their improved functionality, but also to test their impact in the system architecture.

It is mainly in the context of nuclear power plants where we have been applying our prototypes. Up to now, the Simulation agent hasn't played a very active role. Therefore, we are in the process of applying the system to other environments where the simulation agent is more complex, so that it can be tested whether its design is adequate or it needs to be modified.

Finally, the Student Modelling group of agents have been subject to less experimentation than the rest, since its behaviour is quite complex from the pedagogical point of view. Therefore, a research line has been established to fully understand its implications and to modify the architecture where needed.

Acknowledgements

This research has been funded by the Spanish Ministry of Science and Technology through the MAEVIF project under contract TIC2000-1346.

References

[1] James C. Lester and Brian A. Stone. Increasing believability in animated pedagogical agents. In *Proceedings of the First International Conference on Autonomous Agents*, pages 16–21. ACM Press, February 1997.

[2] James C. Lester, Jennifer L. Voerman, Stuart G. Towns, and Charles B. Callaway. Deictic believability: Coordinating gesture, locomotion, and speech in lifelike pedagogical agents. *Applied Artificial Intelligence*, 13:383–414, 1999.

[3] Jeff Rickel and W. Lewis Johnson. Animated agents for procedural training in virtual reality: Perception, cognition, and motor control. *Applied Artificial Intelligence*, 13:343–382, 1999.

[4] Jeff Rickel and W. Lewis Johnson. Virtual humans for team training in virtual reality. In *Proceedings of the Ninth International Conference on Artificial Intelligence in Education*, pages 578–585. IOS Press, 1999.

[5] M.J. Evers and A. Nijholt. Jacob - an animated instruction agent for virtual reality. In *Advances in Multimodal Interfaces - ICMI 2000, Third International Conference*, volume 1948 of *LNCS*, pages 526–532, Beijing, China, October 2000. Springer-Verlag.

[6] Gonzalo Mendez, Jeff Rickel, and Angelica de Antonio. Steve meets jack: the integration of an intelligent tutor and a virtual environment with planning capabilities. In *4th International Working Conference on Intelligent Virtual Agents (IVA03)*, volume 2792 of *LNCS-LNAI*, pages 325–332, Kloster Irsee, Germany, September 2003. Springer-Verlag.

[7] Gonzalo Mendez, Pilar Herrero, and Angelica de Antonio. Intelligent virtual environments for training in nuclear power plants. In *Proceedings of the 6th International Conference on Enterprise Information Systems (ICEIS 2004)*, Porto, Portugal, April 2004.

[8] D.H. Sleeman and J.S Brown, editors. *Intelligent Tutoring Systems*. Academic Press, London, 1982.

[9] E. Wenger. *Artificial Intelligence and Tutoring Systems. Computational and Cognitive Approaches to the Communication of Knowledge*. Morgan Kaufmann Publishers, Los Altos, California, 1987.

[10] A. Munro, D.S. Surmon, M.C. Johnson, Q.A. Pizzini, and J.P. Walker. An open architecture for simulation-centered tutors. In *Artificial Intelligence in Education. Open Learning Environments: New Computational Technologies to Support Learning, Exploration and Collaboration. (Proceedings of AIED99: 9th Con-ference on Artificial Intelligence in Education)*, pages 360–67, Le Mans, France, 1999.

[11] M.N. Alpdemir and R.N. Zobel. A component-based animation framework for devs-based simulation environments. In *Simulation: Past, Present and Future. 12th European Simulation Multiconference*, 1998.

[12] K. Demyunck, J. Broeckhove, and F. Arickx. Real-time visualization of complex simulations using veplatform software. In *Simulation in Industry'99. 11th European Simulation Symposium (ESS'99)*, pages 329–33, 1999.

[13] R. Darken, C. Tonessen, and J.K. Passarella. The bridge between developers and virtual environments: a robust virtual environment system architecture. In *Proceedings of the SPIE - The International Society for Optical Engineering*, volume 2409, pages 234–40, 1995.

[14] Franco Zambonelli, Nicholas R. Jennings, and Michael Wooldridge. Developing multiagent systems: The gaia methodology. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 12(3):317–370, July 2003.

[15] R. E. Fikes and N. J. Nilsson. Strips: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4):189–208, 1971.

[16] D. Nau, T.C. Au, O. Ilghami, U. Kuter, W. Murdock, D. Wu, and F.Yaman. Shop2: An htn planning system. *Journal of Artificial Intelligence Research (JAIR)*, 20:379–404, 2003.

[17] Alfonso Gerevini, Alessandro Saetti, and Ivan Serina. Planning through stochastic local search and temporal action graphs. *Journal of Artificial Intelligence Research (JAIR)*, 20:239–290, 2003.