Training Agents: An Architecture for Reusability

Gonzalo Mendez and Angelica de Antonio

Computer Science School,
Technical University of Madrid,
Campus de Montegancedo s/n, 28660 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. 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 that is intended to be easily extended and modified. Also, some experiments to test the suitability of the architecture are shown.

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 it will be described how the traditional architecture of Intelligent Tutoring Systems (ITS) [8,9] has been extended to support Virtual Environments (section 2) and how it has been transformed into an agent-based architecture (section 3). In section 4, an explanation of the functionality of the authoring tool will be given. Section 5 will present a discussion of the results that have been achieved with the MAEVIF project. Then, the basic functioning of the system will be described (section 6), and finally, in section 7, some future work lines will be shown.

2 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].

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.

Since 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 student, and they are frequently used for team training. An ITS is intended to adapt the teaching

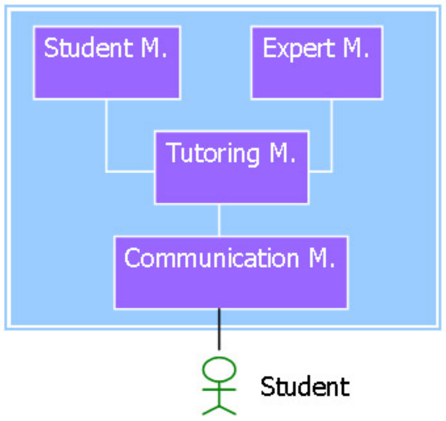

Fig. 1. Architecture of an ITS

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. 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.

3 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 2 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

Analyzing the responsibilities of these agents, some additional roles can be identified that point to the creation of new, subordinate agents that can carry them out, subsequently giving rise to a hierarchical multi-agent architecture.

3.1 Central Communication Agent

The Central Communication Agent is responsible for the communication between the Virtual Environment and the Tutoring System. It 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 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.

3.2 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.

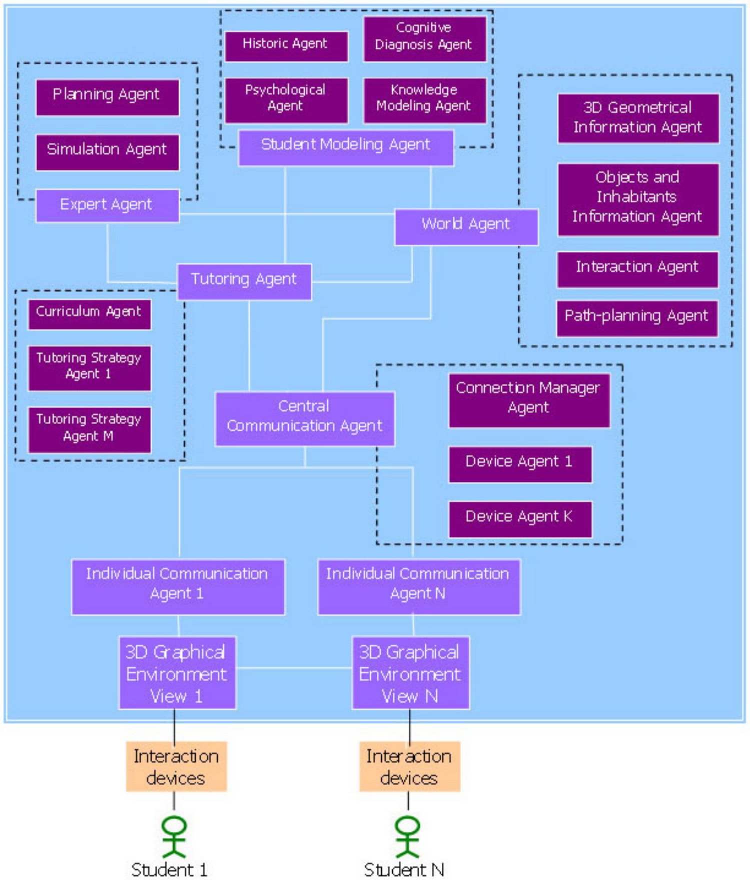

Fig. 2. Agent-based architecture

Figuring out the student's abilities and beliefs/knowledge is usually not a trivial issue. To better individualize training and appropriately understand the student's behavior, a representation of some of its personal features (personality traits, mood, attitudes,...) is defined and maintained. To do this, 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.

3.3 World Agent

The World Agent is in charge of maintaining a coherent model of the VE, so that all the agents and students have the same information about the state of the world.

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.

3.4 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.

3.5 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 adaptation of the tutoring strategy to every particular student may also encompass how the virtual tutor will behave: a student may need a tutor with a particular character (e.g., training children may require a funny, enthusiastic tutor, while for training nuclear power plant operators a more serious one will be more convenient), or with a specific mood (e.g., if a student does not pay much attention to the procedure for long, a disgusted tutor may be effective). Poor or upsetting tutor behaviors will lead to a lack of believability, possibly reducing the student's feeling of presence and therefore the effectiveness of the training.

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.

3.6 Communication with the Virtual Environment

Currently, the proposed architecture has been implemented using JADE (*Java Agent DEvelopment Framework*), while the VE has been built using C++ and OpenGL. The communication between the agents and the VE is made using a CORBA middleware, which has allowed us to distribute the different elements of the training application in different machines (see Fig. 3).

When the application is started, a few general-purpose objects are created that allow the communication of events that affect all users. In addition, when a student connects to a training session, some specifical objects are created, too, so that the communication that only affects that student can be carried out. Every time a message has to be sent from the VE to JADE, the appropriate object receives the information that has to be transmitted.

Some information has to be exchanged between the different VE clients that correspond to each student, such as changes in the positions of the avatars and objects. Microsoft's DirectPlay library has been used with this purpose.

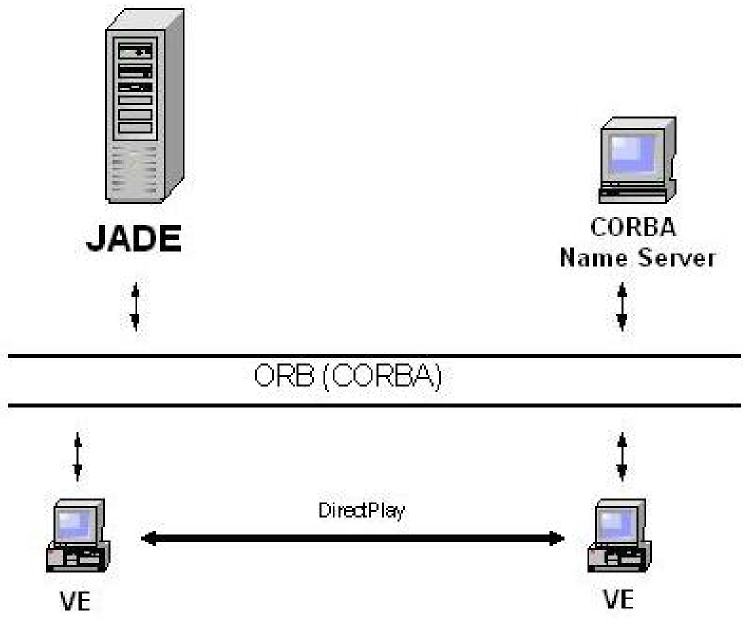

Fig. 3. CORBA communication architecture

4 Authoring Tool

The architecture that has been described in the previous sections has allowed us to build a basic infrastructure of agents that work 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 different kinds of 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. This environment is typically created using 3DStudio Max or a similar application, and is then exported to the format that has been created for the MAEVIF system. A script has been created to be used with 3DStudio with this purpose.

The human tutor 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. pre-conditions, post-conditions, parameters, animations that must be triggered...). These actions are stored in an xml file that is read by the appropriate agents when a training scenario of the MAEVIF system is started.

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. This information is also stored in an xml file that is read by the Tutoring agent when a training scenario is started.

In turn, some variables of the initial state will be generated randomly every time the students have to train the activity, so that they can solve the same problem starting from different situations.

The authoring tool also generates the world map that is used by the A* algorithm for the path-planning task. To do this, all the objects that are present in the VE are projected on the floor of the scenario, which is divided in cells. All the cells are marked as occupied by an object or free, and these free cells are used by the A* algorithm to calculate the best route between two points.

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 in 1999, during a one year period. The re-development using our infrastructure has just taken a few weeks, and the achieved functionality is superior. For instance, the previous implementation was for a single user, the tutor was not embodied, and the communication tutor-student was restricted to correction feedback.

5 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, due to the fact that it is a collaborative process, 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 utilization of a new planner based on SHOP2 (*Simple Hierarchical Ordered Planner 2*) [16] or LPG (*Local search in Planning Graphs*) [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.

However, there are two factors that suggest that collaborative planning is the adequate solution. The first one is the fact that, given a planning algorithm, our solution allows for the real-time inclusion of new agents with different knowledge that can help to solve a problem. In addition, a careful design of the operators and their responsibilities can minimize the impact of a change in the planning algorithm or in the knowledge representation.

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, namely the Virtual Tutor agent, whose responsibility is to control the 3D representation of the tutor (its embodiment), and the Perception agent, who is in charge of monitoring the events of the virtual world. Both of them are under the supervision of the World agent.

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 Tutoring 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.

5.1 Performance Issues

Performance is always an important issue in applications where real-time execution is needed, and agent-based architectures tend to easily raise concerns about this matter. Using an individual agent for each high-level responsibility may seem an unnecessary waste of processing capacity.

Even though our main concern were the software engineering issues, such as extensibility and reusability, we have devoted some effort to identify bottlenecks in terms of performance, given that the architecture will be useless if it can be used due to performance issues.

Three potential sources of problems have been identified: rendering, communications and agent platform, and they have been tested using different configurations (vg. agent platform running in one machine, one VE running in one machine, agent platform and one VE running in one machine, agent platform running in one machine and several VEs running in different machines). The results we have obtained show that the architecture does not influence much the performance of the system.

In contrast, it seems to be the network communication what lowers the execution speed, and the more students there are, the slower the application runs. This effect can be appreciated since the first student connects to the training session, which is leading us to redesign the communication mechanism to improve this aspect.

6 How the System Works

In this section, a sample training session with one student will be presented. The student is a maintenance operator in a Nuclear Power Plant who has to learn how to change a filter that divides two sections of a pipe.

During the training, when an activity is posed to the student, the Planning agent builds the plan that leads to the resolution of the activity, given the initial state at that moment and the desired final state of the world. Moreover, during the planning process, the Path-Planning agent computes the ideal trajectories through the geometric representation of the environment that the students must follow to accomplish the plan.

In order to learn this activity, the student must carry it out in the virtual environment. Not knowing what to do, the student uses a voice recognition system to ask the tutor *"What should I do now?"* (the student can ask some other questions, such as *"What is this for?"*, *"What happens if I. . . ?"*, *"What should I have done?"*). The question is sent to the tutoring system, and the Tutoring agent identifies it is a question he is in charge to answer. He asks the Planning agent for the next action in the plan, builds the answer and sends it to the student who, through a text-to-speech application, hears the tutor saying *"You have to remove the filter"*.

The student tries to carry out the action *remove filter*. To do it, using a data glove, he touches the filter to select it and says *"remove"*. When this happens, the Individual Communication agent associated with the student receives a message and informs about this attempt to the Central Communication agent; eventually, the message is delivered to the Tutoring agent.

Now, the Tutoring agent needs to find out whether the action can be executed under the current conditions in the virtual world, that is, if the preconditions of the action hold. For that, the Tutoring agent resorts to the Interaction agent, since *remove filter* is an action in this level of abstraction. The Interaction Agent determines that he needs to check whether the student's avatar is close enough to the filter and if he is carrying the appropriate tools in order to remove it. To check these preconditions, the Interaction agent writes them in a blackboard that is used as a communication mechanism between agents. The 3D Geometrical Information agent and the Objects and Inhabitants Information agent read the blackboard and see there are preconditions they are able to check. Each precondition corresponds to one and only one of the aforementioned agents.

If all the preconditions of the action hold, the Interaction agent must guarantee the execution of the consequences of the action. For that, it may need to delegate some responsibilities on other agents, such as the World agent and the Simulation agent, using the blackboard again as a communication mechanism. One of the consequences, managed by the Interaction Agent itself, will be launching a 3D animation in the virtual world that represents the student removing the filter. The command is sent to the VE via the Communication agents (in case there are several students, this message is sent to all the students, since all the students should see the animation).

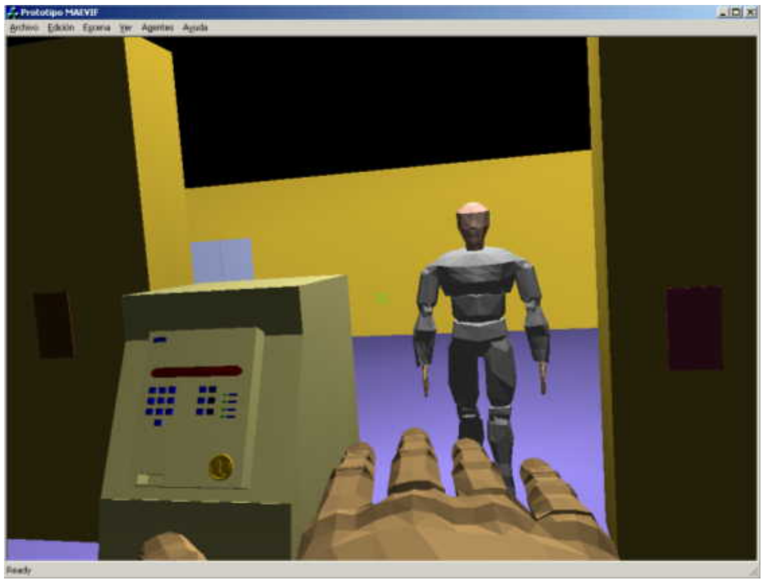

Fig. 4. The MAEVIF system

When the Tutoring agent receives the result of verifying the preconditions of the action from the Interaction agent, it asks the Student Modelling agent to register the action and the result of the verification, and it checks whether the executed action is valid with respect to the plan associated with the activity. If this action is the next correct action according to the plan, the Tutoring Agent asks the Student Modelling agent to register that the student has carried out the correct action. Otherwise, the Tutoring agent allows the student to go on in spite of having executed an incorrect action. This strategy poses a new problem, since the Tutoring Agent needs to know whether the desired final state is reachable from the current state of the world. To find this out, the Planning agent must be endowed with the capacity of re-planning.

The movements of the student in the virtual world are considered a special kind of action that is managed in a different manner to the one explained above. As the student moves through the environment, the Central Communication agent informs the 3D Geometrical Information Agent of the new student's positions. At the same time, the Tutoring agent asks the 3D Geometrical Information agent for these positions, in order to compare them with the trajectory provided by the Path-Planning agent, and to inform the Student Modelling agent so that it can store the trajectory followed by the student during the training session. As a result of the comparison between the ideal trajectory and the student's trajectory, a quality measure of the student's trajectory is calculated by the Path-Planning agent and then stored by the Student Modelling agent.

All through the training, the virtual tutor, controlled by the Virtual Tutor agent, follows the student in order to supervise his actions and correct them.

7 Future Work

As it has been mentioned previously, 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 testbed 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.

Another research is being carried out in parallel to design an architecture for the cognition of intelligent agents, with reactive, deliberative and social capabilities, and it is planned to use that architecture for the Virtual Tutor and any other cognitive agents that may be required (such as zebras, lions or simulated students).

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. Lester, J.C., Stone, B.A.: Increasing believability in animated pedagogical agents. In: Proceedings of the First International Conference on Autonomous Agents, ACM Press (1997) 16–21
2. Lester, J.C., Voerman, J.L., Towns, S.G., Callaway, C.B.: Deictic believability: Coordinating gesture, locomotion, and speech in lifelike pedagogical agents. Applied Artificial Intelligence **13** (1999) 383–414
3. Rickel, J., Johnson, W.L.: Animated agents for procedural training in virtual reality: Perception, cognition, and motor control. Applied Artificial Intelligence **13** (1999) 343–382
4. Rickel, J., Johnson, W.L.: Virtual humans for team training in virtual reality. In: Proceedings of the Ninth International Conference on Artificial Intelligence in Education, IOS Press (1999) 578–585
5. Evers, M., Nijholt, A.: Jacob - an animated instruction agent for virtual reality. In: Advances in Multimodal Interfaces - ICMI 2000, Third International Conference. Volume 1948 of LNCS., Beijing, China, Springer-Verlag (2000) 526–532

6. Mendez, G., Rickel, J., de Antonio, A.: 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., Kloster Irsee, Germany, Springer-Verlag (2003) 325–332
7. Mendez, G., Herrero, P., de Antonio, A.: Intelligent virtual environments for training in nuclear power plants. In: Proceedings of the 6th International Conference on Enterprise Information Systems (ICEIS 2004), Porto, Portugal (2004)
8. Sleeman, D., Brown, J., eds.: Intelligent Tutoring Systems. Academic Press, London (1982)
9. Wenger, E.: Artificial Intelligence and Tutoring Systems. Computational and Cognitive Approaches to the Communication of Knowledge. Morgan Kaufmann Publishers, Los Altos, California (1987)
10. Munro, A., Surmon, D., Johnson, M., Pizzini, Q., Walker, J.: An open architecture for simulation-centered tutors. In: Artificial Intelligence in Education. Open Learning Environments: New Compu-tational Technologies to Support Learning, Exploration and Collaboration. (Proceedings of AIED99: 9th Con-ference on Artificial Intelligence in Education), Le Mans, France (1999) 360–67
11. Alpdemir, M., Zobel, R.: A component-based animation framework for devs-based simulation environments. In: Simulation: Past, Present and Future. 12th European Simulation Multiconference. (1998)
12. Demyunck, K., Broeckhove, J., Arickx, F.: Real-time visualization of complex simulations using veplatform software. In: Simulation in Industry'99. 11th European Simulation Symposium (ESS'99). (1999) 329–33
13. Darken, R., Tonessen, C., Passarella, J.: 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. (1995) 234–40
14. Zambonelli, F., Jennings, N.R., Wooldridge, M.: Developing multiagent systems: The gaia methodology. ACM Transactions on Software Engineering and Methodology (TOSEM) **12** (2003) 317–370
15. Fikes, R.E., Nilsson, N.J.: Strips: a new approach to the application of theorem proving to problem solving. Artificial Intelligence **2** (1971) 189–208
16. Nau, D., Au, T., Ilghami, O., Kuter, U., Murdock, W., Wu, D., F.Yaman: Shop2: An htn planning system. Journal of Artificial Intelligence Research (JAIR) **20** (2003) 379–404
17. Gerevini, A., Saetti, A., Serina, I.: Planning through stochastic local search and temporal action graphs. Journal of Artificial Intelligence Research (JAIR) **20** (2003) 239–290