AN AGENT-BASED ARCHITECTURE FOR THE DEVEL0PMENT OF INTELLIGENT VIRTUAL TRAINING ENVIRONMENTS

A.DE ANTONIO, R.IMBERT, J.RAMIREZ, G.MENDEZ

Computer Science School, Technical University of Madrid, Campus Montegancedo, 28660 Boadilla del Monte (Madrid), SPAIN

E-mail: angelica@fi.upm.es

Training is a promising application area of three dimensional virtual environments. These environments allow the students to navigate through and interact with a virtual representation of a real environment in which they have to learn to carry out a certain task. They are especially useful in situations where the real environment is not available for training, or it is very costly or risky. A good example is training of nuclear power plant operators. A multi-user virtual environment also allows for the training of teams. An Intelligent Virtual Environment for Training (IVET) results from the combination of a Virtual Environment and an Intelligent Tutoring System (ITS). IVETs are able to supervise the actions of the students and provide tutoring feedback. The Intelligent Tutoring component of an IVET usually adopts a virtual representation (a Pedagogical Virtual Agent) that inhabits the environment together with the virtual representations of the students (avatars). In this paper we propose an architecture for the development of this kind of systems, which is based on a collection of cooperative software agents. The first level of the architecture is an extension of the classical ITS architecture that adds to the expert, student, tutoring and communication modules a new module which is called World Module. Several software agents compose each module. These software agents communicate among them directly via FIPA messages and indirectly via a common data structure which is used for the collaborative development of plans. The students can choose among several interface devices for the interaction with the environment, ranging from the simple monitor + mouse + keyboard combination to the most complex and immersive combination: head mounted display + motion tracking + data glove + voice recognition. Communication agents are responsible for the management of the different interaction devices and combinations. Underlying the virtual environment, some Expert Simulation Agents are in charge of simulating the behaviour of the system that is represented through the VE (for example the behaviour of the nuclear power plant). The effects of the actions in the VE over the simulation are controlled by the Interaction Agent. Student Modelling Agents register the actions performed by the students, and Tutoring Agents compare those actions with the possibly correct procedures for a given task. These are some of the agents that cooperate in the proposed architecture. The development of a new IVTE based on this architecture will consist on the adaptation and particularization of the constituent agents.

1 Introduction

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 particular solutions and monolithic architectures, and in many cases forgetting the principles and techniques of the Software Engineering discipline [6]. 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 [1,4] or the interaction devices and hardware [3].

As a result, current VEs lack many of the desirable quality attributes of any software system, such as flexibility, reusability, maintainability or interoperability.

The size and complexity of VEs will continue to increase in the future, making this situation even worse. Many researchers and developers of VEs are starting to recognize the need of a Software Engineering approach to the development of VEs [2,5].

In particular, there is a need to define “standard” architectures in order to facilitate the development of individual components by different teams or organizations which may have different skills and knowledge. The development of a new VE will then consist of the selection and adaptation of existing components, and their assembly and integration. In this way, components will be reusable, the system will be flexible to be extended with new components, and the interfaces among components will be clearly defined to facilitate interoperability of possible very heterogeneous components.

Unfortunately, we are still very far from this ideal state. Given the broad variety and diversity of VEs and their applications, and taking into account that they may require different architectures, we have

decided to restrict the scope of our research to a certain type of VEs, namely Virtual Environments for Training (VETs).

The goal of this kind of VEs is to train one or more students in the execution of a certain task. They are especially useful in those situations in which training in the real environment is either impossible or undesirable because it is costly or dangerous. Lets consider as an example training the operators of a nuclear power plant in the execution of maintenance interventions. In the real environment the trainees would be subject to radiation, which is of course unacceptable for their health, and additionally it would be impossible to reproduce some maintenance interventions without interfering with the normal operation of the plant.

In VETs, the supervision of the learning process can be performed by human tutors or it can be performed by intelligent software tutors, also known as pedagogical agents (in this case we will call it an IVET). Those pedagogical agents, in turn, can be embodied and inhabit the virtual environment together with the students or they can be just a piece of software that interacts with the student via voice, text or a graphical user interface. Some pedagogical agents have been developed to date, in some cases with quite advanced tutoring capabilities. One of the best known is STEVE, developed in the Center for Advanced Research in Technology for Education (CARTE) of the Intelligent Systems Division of the University of Southern California (USC) [7,8].

2 Extension to the architecture of Intelligent Tutoring Systems

Our approach to the definition of an architecture for VETs is based on the agent paradigm. The rationale behind this choice is our belief that the design of highly interactive VETs 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 (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.

Starting from the idea that a VET can be seen as an ITS (an IVET, Intelligent Virtual Environment for Training), 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 (see figure 1).

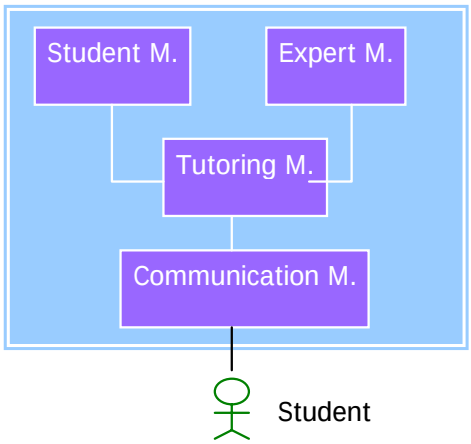

Figure 1. Architecture of an ITS

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 the training of teams. An ITS is intended to adapt the teaching and learning process to the needs of every individual student, but they interact with the system one at a time. However, in a multi-student IVET the systems would have to adapt both to the characteristics of each individual student and to the characteristics of the team.
- 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 very complex virtual reality devices such as HMD (head mounted displays), data gloves or motion tracking systems. 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 a IVET. However there is a fundamental difference among them. In an IVET some of the learning goals may be 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. For instance, a nuclear power plant operator in an IVET may have to learn that in order to open a valve he has to walk to the control panel, which is located on the control room, and press a certain button. Therefore, the ITS needs to have explicit knowledge about the 3D VE, its state, and the possibilities of interaction with 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, plus 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 semantical 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 tutoring decisions specific for a certain student. The expert module will contain all the necessary data and inference rules to maintain a simulation of the behavior of the system that is represented through the VE (for example the behavior of a nuclear power plant). The student module, finally, will contain 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 of the previous section as a reference, the next step was to decide which software agents would be necessary to transform this component-oriented architecture into an agent-oriented architecture. In an agent-oriented architecture, each agent is capable of performing a certain set of tasks, and is capable of communicating with other agents to co-operate with them in the execution of those tasks.

Our agent-based architecture has five principal agents corresponding to the five key modules of the extended ITS architecture:

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

Each of these principal agents may relate to, communicate with and delegate some tasks to other subordinate agents, giving rise to multi-level agent architecture.

In this way, the Communication Agent will delegate on a set of Individual Communication Agents dedicated to each student. There is also a set of Device Agents to manage the different devices that can be used to interact with the environment and make the system independent of any specific combination of

interaction devices. There is also a Connection Manager Agent which is responsible of coordinating the connections of the students.

The Student Modeling Agent is assisted by:

- A Historic Agent, which is responsible of registering the history of interactions among the students and the system.
- A Psychological Agent, which is responsible of 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 Modeling Agent, which is responsible of building a model of the student's current knowledge and its evolution.
- A Cognitive Diagnostic Agent, which is responsible of trying to determine the causes of the student's mistakes.

The World Agent is related to:

- The Objects and Inhabitants Information Agent, which has geometrical and semantical knowledge about the objects and the inhabitants of the world. This agent, for instance, will be able to answer questions about the location of the objects or their utility.
- The Interaction Agent, which has knowledge about the possible interactions with the environment and the effects of those interactions. For instance, 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 move along the environment without colliding with objects or walls.

The Expert Agent, in turn, is related to other agents that are specialists in solving problems related to the subject matter that is being taught to the students. This is one of most variable components in an IVET. Generally, there will be at least one Simulation Agent. In many IVETs the goal of the system is to train students in the execution of procedures. In our prototype for nuclear power plants, for instance, the goal is to teach a team of operators to execute some maintenance procedures. In this case, the Expert Agent should be able to find the best procedure to solve a given malfunctioning situation, and this is achieved by a Planning Agent that is able to apply intelligent planning techniques like STRIPS. If the IVET was to be used for teaching Chemistry, for instance, the Expert Agent should have knowledge about the chemical elements and should be able to plan and simulate reactions.

The Tutoring Agent, finally, will be assisted by:

- A Curriculum Agent, which has knowledge of the curricular structure of the subject matter.
- Several Tutoring Strategy Agents, which implement different tutoring strategies.

Figure 2 shows how the extended ITS architecture is transformed from a modular point of view to an agent-based architecture.

4 Implementation

The agent-based architecture for IVETs that was described in the previous sections has been implemented with a combination of quite heterogeneous technologies.

The agents have been implemented in Java, and the direct communication among them has been realized using the Jade platform with FIPA ACL messages. The 3D VE and avatars have been modeled with 3D Studio Max and imported into OpenGL format with a specific tool developed for that purpose. The visualization of the 3D models, animations and interactions are managed by a program in C++, making use of the OpenGL graphical library. Microsoft's DirectPlay library has been used for direct communication among the different 3D graphical environment views in order to take into account the movements and actions of the other students for real-time update of each view. Microsoft's DirectInput has been used to manage some interaction devices, namely the mouse, keyboard and joystick. The head

mounted display and data glove inputs and outputs are managed by specific libraries. Communication among the C++ VE and the Java agents is performed by using a middleware of CORBA objects.

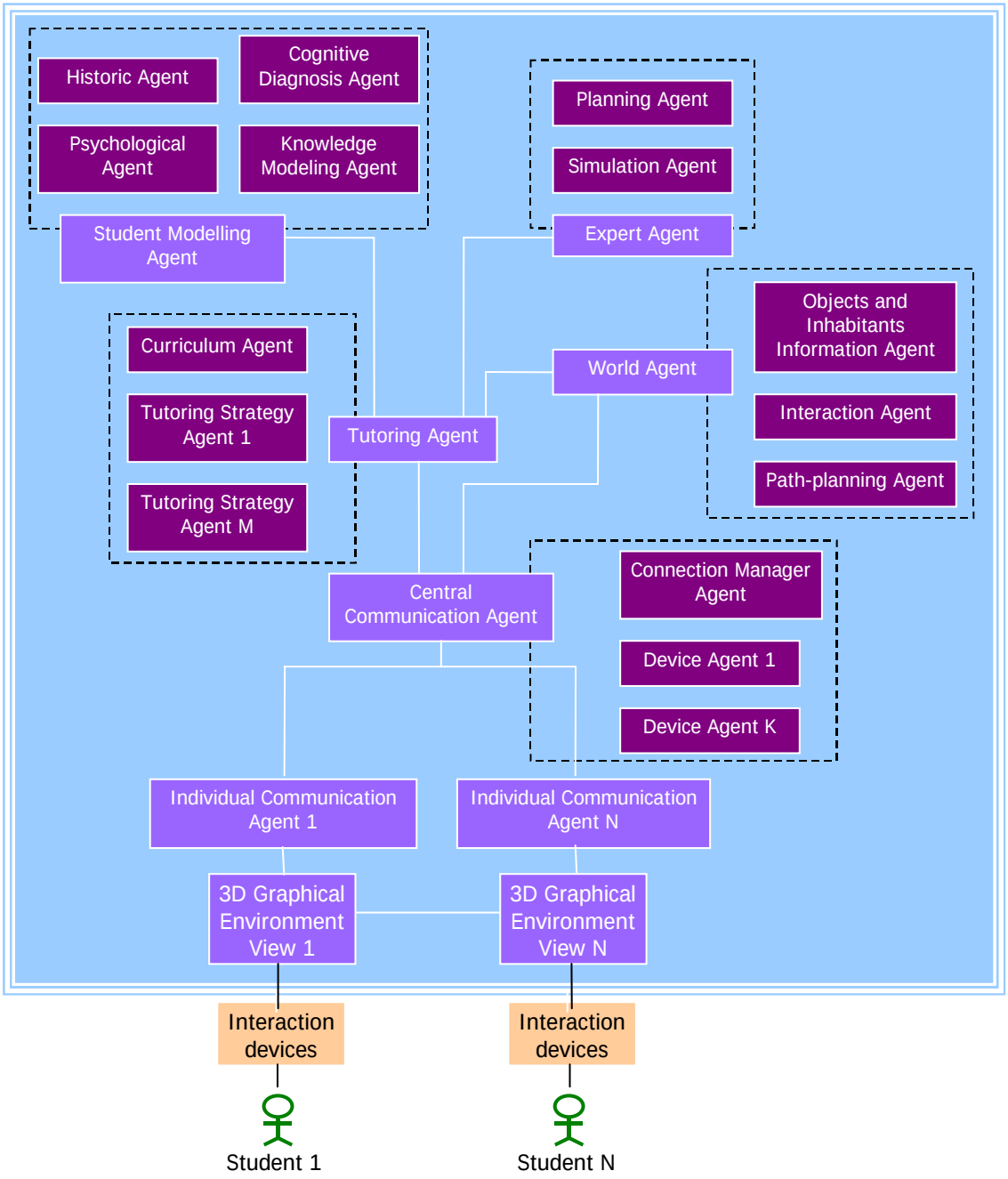

Figure 2. Agent-based architecture for IVETs

5 Conclusions

An agent-based architecture is proposed in this paper for the design of Intelligent Virtual Environments for Training. The roots of this architecture are in the generic architecture of an Intelligent Tutoring System, which has been firstly extended to be applicable to IVETs, and has been then transformed into an agent-

based architecture by the identification of the set of generic agents that would be necessary to accomplish the tasks of each module.

One of the advantages of the proposed architecture is that it is possible to build a basic infrastructure of agents that work as a runtime engine. In order to develop a new IVET, the author's task will consist of: selecting the desired agents among the available ones (for instance selecting the Tutoring Strategy Agent that implements the desired tutoring strategy); configuring the parameters that govern the behaviour of those agents (for instance the duration of the session, the number of mistakes that will be allowed before the Tutoring Agent tells the student the correct answer, etc.); providing the data specific to the new IVET and subject matter (the geometrical model of the VE, the curriculum, the actions that are possible in the new VE and their effects on the simulation, etc.); and in the worst case creating new agents and registering them in the platform.

The proposed architecture, and its realization in a platform of generic and configurable agents, will facilitate the design and implementation of new IVETs, maximizing the reuse of existing components and the extensibility of the system to add new functionalities.

6 Acknowledgements

Our work draws from the results obtained in the MAPI project ("Modelo Basado en Agentes Cooperativos para Sistemas Inteligentes de Tutoría con Planificación Instructiva", funded by CICYT from 1996 to 1999) and is currently being funded by the Spanish Ministry of Science and Technology through project MAEVIF (TIC00-1346).

References

1. Alpdemir, M.N., Zobel, R.N. A component-based animation framework for DEVS-based simulation environments. Simulation: Past, Present and Future. 12th European Simulation Multiconference (ESM'98), p.79-83. Manchester, UK. 1998.
2. Brown, J., Encarnaçao, J., Shniderman, B. (1999). Human-Centered Computing, Online communities, and Virtual Environments. IEEE Computer Graphics and Applications. Vol 19, Nº6, p. 70-74.
3. Darken, R., Tonessen, C., Passarella, Jones K. The bridge between developers and virtual environments: a robust virtual environment system architecture. Proceedings of the SPIE – The International Society for Optical Engineering, vol.2409, p.234-40. 1995.
4. Demyunck, K., Broeckhove, J., Arickx, F. Real-time visualization of complex simulations using VEplatform software. Simulation in Industry'99. 11th European Simulation Symposium (ESS'99), p.329-33. 1999
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. Munro, A., Surmon, D.S., Johnson, M.C., Pizzini, Q.A., Walker, J.P. An open architecture for simulation-centered tutors. Artificial Intelligence in Education. Open Learning Environments: New Computational Technologies to Support Learning, Exploration and Collaboration. (Proceedings of AIED99: 9th Conference on Artificial Intelligence in Education), p.360-67. Le Mans, France. 1999.
7. Rickel, J., Johnson, W.L.: Animated agents for procedural training in virtual reality:Perception, cognition and motor control. Applied Artificial Intelligence 13 (1999) p.343–382.
8. Rickel, J., Johnson, W.L.: Task Oriented Collaboration with Embodied Agents inVirtual Worlds. MIT Press (2000).