Intelligent Virtual Environments for Training: An Agent-based Approach

Angélica de Antonio, Jaime Ramírez, Ricardo Imbert, and Gonzalo Méndez

Technical University of Madrid
Madrid, Spain
{jramirez, angelica, rimbert}@fi.upm.es, gonzalo@gordini.ls.fi.upm.es
http://decoroso.ls.fi.upm.es

Abstract. In this paper we propose an architecture for the development of Intelligent Virtual Environments for Training, which is based on a collection of cooperative software agents. The first level of the architecture is an extension of the classical Intelligent Tutoring System 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. Moreover, the proposed architecture includes agents able to simulate the behavior of human students and tutors, as well as agents able to plan the procedures to be taught (given an initial state and a desired final state) prior to the tutoring process.

1 Introduction

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. An Intelligent Virtual Environment for Training (IVET) results from the combination of a Virtual Environment (VE) and an Intelligent Tutoring System (ITS). IVETs are able to supervise the actions of the students and provide tutoring feedback. Let's 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 VEs for training, 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 refer to the system as 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 University of Southern California (USC) [1]. In the remaining of this paper, we will describe the architecture of the system (sections 2 and 3) and the agents that are endowed with human features (section 4). Then, we will explain how the system works (section 5). Finally, some conclusions and future work are shown (section 6).

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 particular solutions and monolithic architectures, and in many cases forgetting the principles and techniques of the Software Engineering discipline [2]. 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 [3] [4] or the interaction devices and hardware [5]. 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 semiautonomous 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 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, 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 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: Student Model, Expert Model, Tutoring Model and Communication Model.

The ITS architecture, however, does not fit well with the requirements of IVETs in several respects. 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 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 would have 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 very complex virtual reality devices such as HMDs (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 an 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 in 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 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 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 (e.g. 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 described in the previous section as a starting point, 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 cooperate with them in the execution of those tasks. Figure 1 shows how the extended ITS architecture is transformed, from a modular point of view, into an agent-based architecture. Our agent-based architecture has five 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, and 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.

The Communication Agent will delegate 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.

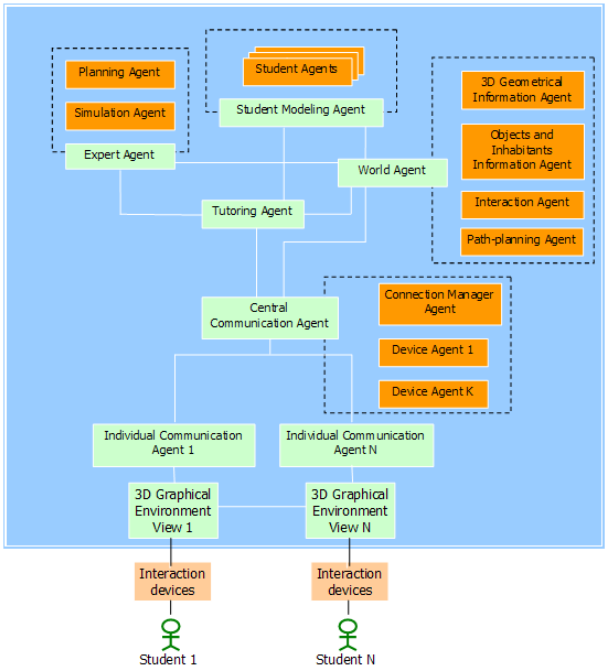

Figure 1: Agent-based architecture for IVETs

The Student Modeling 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 an agent, a Student Agent, which will reflect, as faithfully as possible, all that is known or inferred about the student. A more detailed description of the Student Agents will be presented in section 4.

The World Agent is related to the 3D Geometrical Information Agent; the Objects and Inhabitants Information Agent; the Interaction Agent; and the Path Planning Agent. The 3D Geometrical Information Agent has geometrical information on the objects and the inhabitants of the world. This agent, for instance, will be able to answer questions about the location of the objects. The Objects and Inhabitants Information Agent 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 has knowledge about the possible actions that the students can perform in the environment and the effects of these actions. For instance, it will be able to answer questions like "What will it happen if I push this button?". The Path Planning Agent is capable of finding paths to move along the environment avoiding collisions with other inhabitants and objects. For the purpose of finding these paths, A* algorithm will be applied to a graph model of the environment.

The expert agents contains the expert knowledge about the system 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, in the generic model of an IVET that is being considered, consist of finding an appropriate sequence of actions, or plan, to go from an initial state of the simulated system and the environment to a desired final state. These actions have to be executed by the team of students. The Expert Agent will delegate 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, the Interaction Agent and the Simulation Agent. 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' 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*.

The Tutoring Agent 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 on them and the role to be performed by each character. In section 4, a more detailed description of the Tutoring Agent will be given.

4 Modeling Human Tutors and Students

One of the key assets of ITSs, against other "non intelligent" computer based instructional approaches, is their suitability to adapt themselves to any student's particular skills, knowledge and personal characteristics. This adaptability can

be only reached through a proper student modeling by every Student Agent. Figuring out the student's abilities and beliefs/knowledge is usually not a trivial issue. To better individualize its training and appropriately understand the student's behavior, a representation of some of its personal features (personality traits, mood, attitudes) should be defined and maintained. With this aim, the Student Agent has been designed following a three-layered agent architecture able to manage emotional-driven behaviors (this architecture is described in detail in [6]). All three layers (viz, a reactive layer, a deliberative layer and a social one, showed interwoven in figure 2), share a common knowledge structure called personal model, which is part of the agent's beliefs.

The personal model manages beliefs about Defining Characteristics (DCs) of the student, all the traits that mark out its general behavior, including its personality traits and physical characteristics. By inferring an approximate value for each student DC, the Student Agent will be able to provide the rest of the IVET agents with better predictions about the student's behavior. That may be crucial to adapt the training to the specific student.

In addition, the personal model maintains beliefs about the student's Transient States (TSs), characteristics whose values represent the current state of the student. The most interesting TSs of the *personal model*, in order to understand the student's behavior, are emotions and physical states. Emotions are of paramount importance for the Student Agent, because students' behaviors are rarely only guided by rational, logical decisions, but also by emotional motivations.

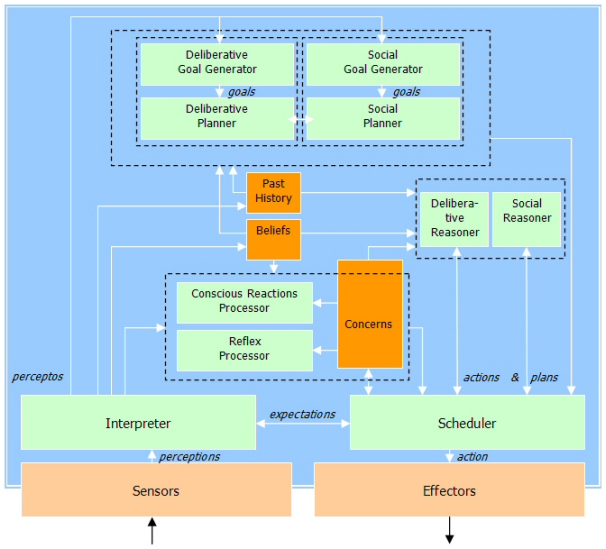

Figure 2: Architecture for agents with behavior influenced by personality and emotions

The designed architecture identifies the appropriate influential relationships among these components of the personal model, along with similar relationships among them and other agent components, such as attitudes and concerns. That

way, the values of the inferred variable elements (namely, TSs, attitudes and concerns) will always be coherent with the student's personality traits and physical characteristics. All these elements have been modeled using fuzzy logic, in order to deal with uncertainty. The relationships identified among them have been represented through special fuzzy rules. The architectural components have been designed so that every one of them is able to deal with those fuzzy values.

Apart from the personal model, the Student Agent manages all the beliefs that the system has -or has inferred- about the student. Those beliefs include the knowledge that the Student Modeling Agent has identified as learnt by the human student. Moreover, the Student Agent copes with information useful to reconstruct and analyze the student's evolution throughout the training process, information stored into the agent's past history structure. So, from the data about the student's knowledge and its personal model, the Student Agent will always be able to infer a plausible student behavior for each situation, based on both logical and emotional reasoning. That is valuable information for some other agents in the IVET, since it makes possible the adaptation of the training strategy considering issues such as having an impulsive student, or a clumsy one handling virtual reality devices, or one in a bad mood. Modeling each specific student by means of the architecture of figure 2 is only one face of the coin. The other one is modeling the trainer, i.e. the Tutoring Agent.

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 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). The reason is that students, despite having a virtual representation in the IVET, are human, and humans expect to identify human-like behaviors in the rest of the IVET characters, including the tutor. 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. As a consequence, the same architecture for agents with behavior influenced by personality and emotions used in the Student Agent has also been used for modeling the Tutoring Agent.

5 Walkthrough of the Agent-Based Architecture

In this section, a walkthrough of the behavior of the IVET during a learning session will be presented. The activities to be taught must be previously specified by using an authoring tool, and the IVET must be correctly configured for the kind of activity at hand. During the training, when an activity is posed to the student, the agents system, based on the planning capacity of the Expert Agent, will compute the plan or sequence of actions associated with the activity, given the initial state and the desired final state of the world. Moreover, during the planning process, the Path Planning Agent will compute the ideal trajectories that the students must follow to accomplish the plan. We situate ourselves in

the domain of the dams and reservoirs, and we suppose that the activity to be taught is related to avoiding the destruction of a dam when the amount of stored water is too high. In order to avoid this catastrophe, an operator must open the dam to allow a certain amount of water to go out the reservoir. For performing that action, the operator will have to use a key that will activate the opening mechanism after introducing a certain code in the control panel of the dam. In order to learn this activity, the student must perform it in the virtual environment. Thus, at a certain moment, the student, using proper interaction devices, tries to carry out the action *use the key*. When this happens, the Individual Communication Agent associated with the student informs about this action to the Central Communication Agent, and eventually the message is delivered to the Tutoring Agent.

5.1 Action Verification and Execution

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 via the World Agent. The Interaction Agent determines that he needs to check whether the student (his virtual representation in the virtual world) is carrying the key, and whether he is close enough to the control panel of the dam to *use the key*. In order to check these preconditions, the Interaction Agent will deposit them in a blackboard, and it will ask the 3D Geometrical Information Agent and the Objects and Inhabitants Information Agent to check the preconditions that are related to each one. A blackboard is used so that the Interaction Agent does not need to know which Agent can check each precondition. In this example, 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, sometimes it needs to delegate on other agents, such as the World Agent (in particular, some of its subordinate agents) and the Simulation Agent, using again a blackboard communication mechanism. One of the consequences, managed by the Interaction Agent itself, will be launching the 3D animation in the virtual world that represents the student using the key. The command is sent via the Communication Agents (for all the students, since all the students should observe the animation). Another consequence of the *use the key* action, this time managed by the Simulation Agent, will be opening the dam, only if the student has introduced the correct code previously; otherwise, using the key will not have any effect on the dam. In order to distinguish between these two situations, the Simulation Agent must know whether the student has introduced the correct code.

5.2 Tutoring Actions

When the Tutoring Agent receives the result of verifying the preconditions of the action from the Interaction Agent, it asks the Student 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 Agent to register that the student has carried out the correct action at this moment. Otherwise, at this point, different tutoring strategies may be applied. One of them may be to allow the student to go on in spite of having just executed an incorrect action, whenever the state resulting from the execution of the action can still be transformed into the desired final state of the activity. 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. For the purpose of finding this out, the Planning Agent must be endowed with the capacity of re-planning. Another more strict tutoring strategy may decide to explain the student the mistake, and to give the student another opportunity to accomplish the activity.

5.3 Dealing with Movements

The movements of the student in the virtual world are considered as 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 ideal trajectory provided by the Path Planning Agent for the current activity, and in order to inform the Student 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, the Tutoring Agent computes a quality measure of the student's trajectory, and this measure is stored by the Student Agent.

6 Conclusions and Future Work

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 (e.g. selecting the Tutoring Agent that implements the desired tutoring strategy); configuring the parameters that govern the behavior of those agents (e.g. 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 (e.g. 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. One of the drawbacks of the proposed architecture in its current state is that it can only be used with collective activities where students do not need to perform actions at the same time, that is, concurrent actions are not permitted. In order to allow this kind of actions, the architecture must be extended so that it can deal with the planning, verification and execution of concurrent actions (the verification and the execution of an action was explained by means of a simple example in section 5). In this sense, cooperative actions, that is, actions that require the intervention of more than one student at the same time, will deserve a special treatment. In a first approach to the problem of planning without concurrent actions, STRIPS and A* algorithms have been employed. However, currently, we are replacing STRIPS with a temporal planner able to deal with concurrent actions. Two temporal planners, SHOP2[1], a hierarchical task network planner, and LPG[2], a domain-independent planner, are being compared. In addition, in order to allow for efficient re-planning, the chosen planner will be modified to take advantage of an already known plan to reach the desired final state in the world.

References

1. Rickel, J., Johnson, W.: Task-Oriented Collaboration with Embodied Agents in Virtual Worlds. In: Embodied Conversational Agents. Eds. Boston: MIT Press (2000)
2. 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
3. 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)
4. 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
5. 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
6. Herrero, P., Imbert, R.: Design of Believable Intelligent Virtual Agents. In: Developing Future Interactive Systems. Idea Group Publishing (2005) 177–211

[1] http://www.cs.umd.edu/projects/shop/
[2] http://zeus.ing.unibs.it/lpg/