Intelligent Agents for Distance Learning

Analía Amandi, Marcelo Campo*, Marcelo Armentano, Luis Berdún*,

ISISTAN Research Institute

Facultad de Ciencias Exactas, UNICEN

Campus Universitario, (B7001BBO) Tandil, Buenos Aires, Argentina

{amandi,mcampo,marmenta,lberdun}@exa.unicen.edu.ar

*Also CONICET

Abstract

Distance learning involves a lot of work of human assistants. These assistants need to be connected for answering student doubts and questions. Intelligent agents can do part of this work. However a problem appears with this promised possibility: how to connect educational applications with these agents. This paper presents a solution to this problem, in which both the capture of student's intentions and agent intervention for helping students are specified. These two design points are defined as connection points. Student intentions define situations in which agents might help. Agent interventions define the context in which agent might assist and the type of help that might give.

Keywords: intelligent agents, distance learning.

1. Introduction

Web-based distance learning technologies are producing a new revolution in the way knowledge can be spread out among very different and distant communities. It brings new teaching-learning odds and knowledge diffusion into the open in a wide range of topics, becoming strategic for promoting a meaningful development, particularly, in underdeveloped countries. In order to mediate in the distance-teaching process, a variety of software platforms

have been developed providing from courses visualizations to teachers-students communication support.

The facilities given by those distance-teaching platforms provide, however, minimal advantages that we can get from a software system. The not so new intelligent tutorial proposals give the basis to the definition of components that facilitate the coalescence of teaching strategies and students' models to conduct the teaching thematic of the tutorial. Despite this advantages, innovations in the intelligent agents field (Wooldridge, 1999), particularly a subtype of these named interface agents (Maes, 1994) (Lieberman, 1997), can give us an additional advantage that would have a favorable effect on the teaching-system quality giving a personalized assistance to students.

Extending the distance-learning platforms with agent technologies opens new possibilities in the educational prospect, complementing the required processes of training and personalization. Agents transform distance-learning systems from communication and information media to systems with active elements that take part in the learning-teaching process.

We say that these agents are active elements because of their autonomy. Agents can intervene in the learning-teaching process when they detect an opportunity to collaborate. Collaboration is made in a personalized fashion as they become acquainted to the students, building a profile and acting according to it. The student profile holds both personal interests and the way is the student learning. In this way, a student assisted by an agent uses a tutorial course through Web interfaces, performing different learning activities that the agent will evaluate to give advice and help the student. For example, if the agent detects that the student have difficulties in a specific topic, it can suggest her what to read or what exercises or activities she should accomplish before going forward into the course contents.

Researchers in the agents field have experimented with different kind of agents during the last decade, for example (Lieberman, 1995), (Pannu and Sycara, 1996), (Chen and Sycara, 1998) and (Godoy and Amandi, 2000). However, few of these researches were made on the

educational domain (Johnson and Shaw, 1997), (Shaw et. al, 1999), (Johnson et. al., 2000) and always restricted to a specific course. Providing this kind of components in a distance-learning platform should improve its quality because it supplies personalized assistance (Guey-Fa, 1993). Nevertheless, to achieve this goal we have to solve the problem related to the connection of a basic educational system to assistant agents.

Our work tries to formalize the process of connecting students' assistant agents to educational systems. These agents will behave towards the difficulties in the student learning in a customized way. To work on this intelligent platform, our University count with a distance-learning system called SAVER, developed at the ISISTAN Research Institute (Campo et al., 2002a). We enhanced this system with agents that assist students in the learning process.

The paper is organized as follow. Section 2 presents the educational context of the proposal. Section 3 introduces the assistant agents. Section 4 exposes the educational web application used as base application for connecting the proposed agents. Section 5 analyzes the troubles that developers have to face for adding these agents to educational applications. Section 6 presents our proposal about the connection of educational application with assistant agents. Finally, section 7 details our conclusions.

2 Educational agents

Information and communication technologies have supplied us with new didactic alternatives in the learning-teaching process, due to the facilities, advantages and opportunities that they offer for information spreading, access, processing and production, under a communicational paradigm centered in the horizontality of the interlocutors. The educational use of these technologies requires two pedagogical-communicative models: one related to the information spreading and the other one related to the tutorial dialog communication, with an optimum interaction between them (Mayer 2000).

Many courses in the Internet are based mainly on the visual design. Two main *common sense* ideas underlay this designs: a) the mere use of the computer promote learning and b) the showy appearance of the information contribute inevitably to a better learning.

Frequently, courses spread out on the web are created converting the content of existing distance or in attendance courses to electronic format, without any considerations about the context differences regarding to modality and/or the used media. Other courses include student-teacher communicative elements. In general, these developments are short or even lack of a didactic web-based design that considerate the specific problematic of distance-learning using these media. Moreover, very few of them take into account any kind of virtual tutor materialized as a software agent.

The organization of the course, learning activities and materials, communication, evaluation and the teacher role are the main factors that intervene in the distance-learning process, particularly through the web. The acquisition of relevant data related to these factors during on line classes will facilitate the obtaining of objective information that would allow us to trace strategies for the web courses design. From this information we would be able to measure the efficiency of a virtual teacher, personalized by a software agent.

In this context, it is important to know the properties that should have the learning activities (Jonassen, 2002) and communication strategies (Peal and Wilson, 2001) to favor the meaning learning (Ausubel et al., 1983) of the knowledge in the Internet distance-learning way, provided with a virtual teacher materialized in a software agent.

3 Interface Agents

Interface agents (Maes, 1994) (Lieberman, 1997) are computational components that act like human assistants. Thus, a user working on a given application (that we named base application) is helped by an agent. For achieving this goal, the agent needs to know the preferences and habits of its user. Therefore, the agent observes its user acting on the base application and then generates a user profile. Form this profile, the agent personalizes her assistance.

A classical example of this type of agents is the secretary agent. In this context, if a user is using a system for organizing her calendar, an agent can observe, like a human secretary,

when the user cancels a meeting because she has something else more important, when she doesn't accept to participate in a meeting, when she arranges short meetings, and thus, with those and other data the agent knows the human that assists, and collaborates with her like a human secretary. Here the base system is the calendar system and the secretary agent has the functionality of, for example, to suggest a change of appointments when an invitation arrives to a meeting the user considers important.

The secretary agent, for example, should learn about what sort of meeting is more important for its user and how this relevance will change in different contexts. Furthermore, the agent has to consider when she has an intervention chance to serve the human she is assisting to. For example, she can present a suggestion autonomously.

In the educational context of a tutorial system, we have detected a kind of agent that will help the students. The base application that we use is called SAVER, and was developed in our institute. The next section gives architectural details of this system and one of the problems that will be tackled in this work to connect the agent to the base application.

The agents proposed in this work for student assistance would have to learn how to contribute to the learning process. Thus, each instance of these agents will be in charge of one student, and will learn from her observation. Each agent will record when the student she is assisting reads a subject, do exercises, do evaluations, read a subject again. Additionally, she will record sequences of these tasks, dates and time spent in each one. Using this information, she will search for patterns that will allow she to assist the student in the learning process. The agent for building the student profile, which will continue consolidating, or altering with time, will use the mentioned patterns. In the profile, the agent also record a level of confidence of each detected pattern, which strongly depends on the number of items that support that information, and on the number of against evidences that may occur. From the previous explanation we can deduce that profiles will evolve with time, as occur with a human assistant that is becoming familiar with her boss. In the same way that a human assistant is more

efficient when she knows more about her boss, the more a computational agent knows her user, the more efficient it will be.

We have presented a general overview of our proposal of adding agents to make distance learning as similar as possible to attendance education. In the following section, we will expose the base application we will use, and then we will detail the proposed kind of agent.

4 Teaching platform used as base application

The distance learning with virtual access system, SAVER (for the Spanish words Sistema de Acceso Virtual a la Educación Remota) is a platform developed at the ISISTAN Research Institute to support common distance learning activities. It is implemented in JAVA, and provides traditional web-based services to remote access to courses, automatic evaluations, chat, forums, collaborative working support, Microsoft Word® interface for courses edition, different kind of user management, among others. The distinctive feature of SAVER is that it is built up using an event driven architecture that enables behavior using composite autonomous units, supported by the so-called *Bubble* framework[1] (Campo et. al., 2002b)(Diaz Pace and Campo, 2001a)(Diaz Pace et. al. 2001b), also developed at the ISISTAN. This software architecture provides SAVER with flexibility and adaptability facilities hard to find in other existing platforms.

The main goal of *Bubble* is allow us to define and organize a set of computational units (UC for short) modeling processes in which many individuals intervene and simulating its global behavior from their interaction. These units are basically active objects implemented as lightweight processes (threads).

The central goal of *Bubble* is to provide a reusable support that intends to:

- Provide extensibility and flexibility to easily add new components and substitute or improve existent ones.

- Have reusable components that can be used en other similar models

---

[1] A *framework* is made of a set of abstract classes and other components that encapsulates behaviour patterns that combine objects in collaboration groups.

- Model a specific problem in different levels of abstraction.

- Integrate to the model visualization tools, data gathering inspection, etc.

The UCs are the basic units of the application and are characterized by an internal state and a set of tasks to perform. There are also composed UCs, that are UCs that contains within them groups of units. These composed units lead us to different levels of abstraction modeling using hierarchical structures. The interaction mechanism used is event driven: each UC has a set of associated sensors registered to different events that they are interested in receiving. There is an implicit invocation mechanism that allows the UCs being notified of the events perceived by its sensors.

Behavior is defined using a set of tasks, each organized using a precondition-action scheme. A task defines a set of actions to be performed by an UC when certain preconditions are satisfied. These preconditions are related to the internal state of the UC, and to the perceived events. Furthermore, we can define composed tasks that encapsulate a network of existing tasks. This allows us to compound and relate different behaviors.

Each participant, student, teacher, tutor, etc in SAVER is represented by an UC that defines the functions that she can perform in the system, check up access security aspects and monitors all her activities. Using this architecture, SAVER can be extended with new functionalities and kind of participants.

Specifically, the UC mechanism can be extended to model intelligent behaviors such us those proposed in this project. New kind of artificial participant, we mean educational agents, can be added to provide new kind of interactions. However, adding that kind of participant requires adapting the base application especially in the points related to coordination and communication mechanisms between human and artificial actors. These mechanisms need to be researched.

5 Problems of agents as distance learning assistants

Current distance learning platforms lack personalized assistance. The proposed kind of agent's goal is to face this problem. However, once the goals of these agents have been set up, we have to face new problems that arise from their materialization.

The agent assisting a student has to learn at least what knowledge has her, how she master the subjects of the course is taking (for example, she knows them in theory or she knows their application), whether the student can relate concepts, how she prefers facing a new subject (in theory, in practice, with graphics, top-down in levels of abstraction or a combination of the above).

As we can observe, the learning of the student capabilities is one problem to be solved. Then another problem arises: mapping the student characteristics to a student profile. To solve this problems, we have to face up three specific points: the kind of machine learning techniques we will use to learn those capabilities, how to represent the student profile of each user, and how to map the results of the learning algorithms to the user profile.

The user profiles specifications in our students have to be carefully built up taking into account that the decision algorithms will use them to select the way of action. Hence, we have to ensure that from the profile specification we can deduce a set of necessary goals for the decision of the actions to be carried out by the agent or at least the actions that will suggest performing to the user.

Finally, we have to clearly specify the results' mapping of the learning algorithms to the user profile to check that we obtain the expected results. The solution to this problem strongly depends on the research work about the profiles specification.

6. Connecting the SAVER educational application with assistant agents

Our approach (Amandi and Armentano, 2003) to attack this connection problem follows the ideas on collaborative interface agents (Rich et. al., 2001), as we illustrate in Figure 2. This approach intends to imitate the relationships held between humans that are working together in some specific task involving a shared artifact, as two painters painting a wall together, or

two architects working on a plan together. In our case, a student and an assistant working together in the learning task.

In this approach, an interface agent replaces the human assistant. Both the student and the assistant agent can interact with the application (the shared artifact), communicate each other, and observe each other's actions.

These collaborative interface agents are indeed a direct way for solving the several troubles that students have to face when using web courses. These agents take the role of a personal assistant helping each student.

Therefore, our primary goal is to join existing educative web applications with the benefits of agency without much programming effort, and doing the minimum changes to the application code.

Here we present a proposal to cope with these issues. Our architecture specifies the components for connecting the educational application with an assistant agent. Figure 3 presents an overview of our software architecture, showing the main components of the assistant agent and their relationships. As can be noted there is the clear separation between the assistant agent and the education web application while the relationships shown in Figure 1 still hold.

The main architectural components for the connection are the following: Task Manager Component and Interaction Manager Component. The Task Manager Component specifies the alternative gestures of the student on the user interface of the application. The Interaction Manager Component specifies the triggers of the application for intervention of the assistant agent.

Therefore, two main steps are specified for materializing the connection between the educational web application and the assistant agent. These two steps are the following:

        1:      Detection of student intentions

        2:      Detection of intervention situations for assisting students

Firstly, the agent *observes* the student using the educational application to detect her learning intentions. It makes use of the application's *task model* to detect how each student gesture contributes to the current learning task. Furthermore, the student interaction with the educational application may serve to enrich the student profile, detecting patterns in the tasks performed. The task model is an important source of knowledge of the assistant agent, because it contains information about all possible tasks that the student can perform.

On the other way, communications with the student are given in two ways. Not only the student can request help of its assistant agent, or ask it a question, but also the assistant agent can determine that it can initiate the communication. To do so, it makes use of different kinds of *interaction schemes* representing intervention situations, along with schemes of the messages presented to the student.

Notice that there is no need of modifying the application itself to integrate it to the agent. The "glue" between the application and the agent is given by the task model, whereas interaction schemes and it presentation in a user-agent communication window separate of the application prevent us of modifying it to carry out the interaction.

The following sections detail each one of these steps.

6.1 Detecting student intentions

The ability to reason about user activities is a key point in the developing of any intelligent user interface (Franklin et. al., 2002). If our assistant agent is able to recognize what the student is doing, she can act to cooperate. Looking at the tasks that the student is performing, and reasoning about sequences of actions, the assistant agent can help to the student. In addition, the understanding of student's activities provides a context to make out suitable interventions of the agent and try to be aware of what the student is most likely to do next.

An assistant agent implemented under our architecture is able to recognize the student's intentions (i.e. the subject she is trying to learn), and analyze the situation to collaborate with the student.

The task manager is the component of the architecture whose main function is to recognize the student intention while interacting with the application. To do so, this component keeps information about all the possible application tasks, recording them in the *task model*, and may be able to identify which of these tasks the student is performing in any given moment.

In Figure 4 we extend the part of Figure 3 related to the Task Manager.

Based on the use of the educational application through its user interface, assistant agents must recognize which activity the student wants to perform for finding a way to collaborate. Often, a single action on the user interface is not enough to detect the task the user is carrying out.

For specifying the possible gestures of a student on a user interface of an educational application, we use a task model. Each gesture is defined as a task, for instance, a reading of a given course page. The next section details this design tool.

Then, we define an *active task* as a task from the task model that is consistent with student actions, in the sense that students follow a way in which their intention can be discovered.

At first, all the tasks are active, because any given event was triggered (since the student did not execute any action). The task model may consider the way that a student action can be seen as a contribution to one candidate user activity that represents a student intention.

We can interpret any student action on the application by five different ways: (1) it is a final action of an active task: current goal is achieved; (2) it is the transition to a following task of an active task: the active task goes a step forward. If there is any active task that is not waiting for the last student action, it *deactivates*; (3) identifies the task the student is carrying out: there is only one active task remaining, therefore the assistant agent is certain of the student's goal; (4) identifies who may perform the current goal or a step in the current task; (5) identifies a parameter of current task.

If no one of last situations is given, the assistant agent concludes that the current action initiates an interruption (Rich and Sidner, 1998), that is, a non-expected action. The occurrence of interruptions may be due to actual changes on the task the student were carrying out or due to an incomplete recipe that does not include the current act even though it ought to. We assume that, in general, the agent's knowledge about the task model will be complete and therefore, the agent should handle a non-expected action as a change in the student goal.

The assistant agent may also consider that the interruption is *definitive* or *transitory*. In the first case, the agent will discard all current active tasks (since the current goal will not be resumed) and will use the task model to get a new set of active tasks based on the last student action. On the other hand, if the agent believes that the interruption is transitory, all active tasks will be kept active with a flag indicating that it could be resumed or reminded by the agent later on (when the new task is completed, or when the agent considers appropriate).

6.1.1 Specifying each gesture of the student

A task defines how the student can reach a goal in a specific user interface of an application. The goal is either a desired modification of the state of a system or a query to it.

Given that Graphical User Interfaces (GUI) typically have the goal of supporting a particular set of human tasks, the first job of a designer that use them is to formalize this intentions in an explicit task model. An explicit task model can be used to control the behavior of our assistant agent that helps a student perform typical educational tasks using a GUI (Eisenstein and Rich, 2002).

There are many different task model representations. In this work, we use the following variation of *ConcurTaskTrees* notation (Paterno et al., 1997).

A task is defined by the following attributes:

-Name: used to represent the task.

-Type: defines if the task is concrete or abstract.

-Subtask of: name of the subtask that contains this one.

-Steps: a vector of tasks that performed together, under given restrictions, achieve the task.

-Restrictions: a vector of restrictions in the execution order between the subtasks.

-Iterative: a boolean representing the iterative nature of the task.

-First action: a set of possible initial event of the task.

-Last action: a set of possible final event of the task.

To build a task model, we have to follow the next three steps:

1.Make a hierarchical logical decomposition of the tasks represented by a tree-like structure.

2. Identify the temporal relations between tasks at the same level.

3. Recognize the student events that allow advancing from a task to the following.

The operators that we use to describe the temporal relationships are:

T1 ||| T2, interleaving: the actions of the two tasks can be performed in any order;

T1 |[]| T2, synchronization: the two tasks have to synchronize on some actions in order to exchange information;

T1 >> T2, enabling: when the first task is terminated then the second task is activated;

T1 []>> T2, enabling with information passing, in this case we want to highlight that when T1 task terminates it provides some value for task T2 besides activating it;

T1 [> T2, deactivation, when one action from the second task occurs the first task is deactivated;

T1*, iteration, the task is iterative;

T1(n) finite iteration, how many times the task will be performed is specified;

[T1], optional task, its performance is not mandatory.

T recursion, the possibility to include in the task specification the task itself.

The first problem that may arise if we simply build task models using these operators, is the possible ambiguity of some expressions. For example in Figure 5(a), we can interpret the specification in two ways: either (T1 [] T2) ||| T3 or T1 [] (T2 ||| T3).

To solve this ambiguity we can introduce an abstract task, which disambiguates the expression, as shown in Figure 5(b).

For example, in Figure 6, we expose an example using our educational application SAVER, in which professor tasks are specified. Temporal relationship between the tasks is enabling with information passing. Then, the agent can detect the intentions of the professor for teaching students, which enable the future agent intervention.

6.2 Detecting intervention situations for assisting students

The interaction between the student and the personal assistant agent determines a mixed-initiative way of human-computer interaction. In this way, not only the student has the possibility of requesting help to the assistant, but also the assistant can take the initiative starting the interaction with the student.

The agent intervention in the activities that the student is performing on the application should be carefully designed to ensure that the student feels that she has the control over the educational application in every moment. It is also important to students to know when the assistant agent can intervene and how the intervention is materialized.

An important problem that come up at this point is to specify when the assistant agent believes it has the expediency of starting an interaction with the student and, moreover, what kind of interaction take up. If these major aspects about agent interventions are clear, it is more likely that the student feels to have the control of the system every time.

Both specifying the advice the agent can give and the moment it will interrupt the student to show them are two open problems related to the interface with the student. These two problems involve significant difficulties to be solved to go forward in the state of the art of interface agents (Lieberman, 2001). Therefore, the second contact point between our web educational application SABER and the assistant agent is represented by the situations that appear in the application that trigger the agent intervention. With the goal of defining steps to

specify these triggers, we first classify the alternative types of interaction between students and assistant agents and then we define the triggers that materialize this second connection point. Finally we expose these triggers inside the architecture.

6.2.1 Specifying the interaction between a student and an assistant agent

Agents can take the initiative during their interaction with student in several ways. We have detected two types of interactions: *Advice* and *Query*. Each of these interaction types will be presented in three different ways of abstraction:

Scheme: Interaction is defined in an abstract way by a fact using variables to symbolize application elements that come into play. For example, analyze(*Chapter*,*Subject*), where *Chapter* is a part of a course and *Subject* is the topic that this chapter treats.

Instance: is a scheme with all of its variables linked to a concrete value. For example, analyze(3, "*European History*").

Visualization: Is the way the assistant agent shows the interaction to the student. For example, "I suggest you analyzing the chapter *3* on European History".

We define both schemes and visualizations at design time, but instances are created during the agent runtime.

Advice

Advice type refers suggestions of actions and reports of problems or special situations. Therefore we consider three subtypes of advice: *Suggestion*, *Warning*, and *Remainder*.

Suggestion

A *suggestion* is made up of four components:

1. A set of actions expressing that the agent is suggesting.

2. The trigger of the suggestion, that can be a problem, a special situation, or a student request.

3. A set of facts expressing the context in with the suggestion is given.

4. A set of possible side effects of executing the suggested actions.

For example, a suggestion for the problem of lacking of knowledge could be a making of a given exercise, is defined by "make(*Exercises, Section, Chapter, Subject*)". A visualization for this suggestion is "*I suggest you making the exercises E1 E2, in Section S2 at Chapter 2 about Stone Age*". The triggers of this suggestion may be unKnown(*Subject*) or badKnown(*Subject*). A possible side effect is believe(unknown(Subject)).

When the assistant agent detects a problem, it can suggest an action or just report the problem. For instance, the agent may decide to advise the student of the "unknown a subject" problem instead of suggesting anything to do. We can visualize it, for example, by "You don't know about Stone Age".

Finally, sometimes the student may not take any action, facing up a problem, or may request the agent some time to treat. In this case, the agent can made use of *remainders*, i.e. notifications of an already given problem or special situation. A remainder is conformed by three attributes: (i) the source of the remainder, which could be a special situation or a problem; (ii) a set of facts expressing the context in which the remainder has no more sense; (iii) a time interval in which the remainder will be given again.

Queries

The query type refers to interventions in which the assistant agent asks the student about a situation where the agent is not sure of what to do or what to think about.

A query scheme is described by two components:

1. A sequence of actions expressing the agent doubt.

> 2. A set of possible instantiations of each non-instantiated variable inside the doubt definition. If it is the case of not containing any non-instantiated variable, the second component of a query will be one of the following values: *yes, no, sometimes, I don't know*.

For example, know(*Hamlet*,X) expresses the assistant agent don't know whether the student knows about the subject *Hamlet*. In this case, X could be instantiated with a number representing the degree of knowledge that the student believes to have about this subject.

6.2.2 Interaction triggers

Interaction triggers are facts that stimulate the agent to assist the student. There are basically two types of triggers: internal and external.

Internal triggers are facts in face of which the agent needs to intercede, initiating an interaction with the student. This kind of trigger has three subtypes: *problem, special situation,* and *doubt*.

External interaction triggers are requests from the student to the assistant agent.

> *Problems*
>
> We define a problem as a set of facts representing a conflictive context in presence of which the assistant agent will act.

An example of problem is, as we mention, the "lacking of knowledge" fact. Its scheme could be unKnown(*Subject*), visualized, for instance, as "You don't know about *Subject*". Once variables had been instantiated, we will get something like "You don't know about *Stone Age*".

As an example, consider the student answer a question of an exercise in an acceptable way but there is a better option for that answer. The scheme of this special situation will be correct(*Exercise,Item,Answer*) and betterOption(*Excecise,Item,A*) and *A ≠ Answer*, visualized, for example, as "There is a better answer for the Exercise *Excercise* Item *Item*". Once instantiated, we may obtain "There is a better answer for the Exercise *E25* Item *d*".

For example, the student may ask the assistant agent to suggest the best page to learn a specific subject: "*Suggest me a page to learn about…*". The context the agent should take into account is the other element of a *learn* scheme that the user may already have enter in the application, such as the subject, or desired knowledge level. With context information, and the student's profile, the assistant agent will give a more effective suggestion.

A suggestion request is made up of two components:

A set of facts expressing the request, for example bestPageToLearn("Stone Age").

A set of facts expressing the context in which the student made the request.

6.2.3 Interaction Manager Component

Figure 7 shows an expansion of Figure 2 related to the Interaction Manager component.

In this Figure, we can observe the Interaction Manager component of the architecture. In this component, elements that represent triggers for agent intervention are clearly defined, being explicit their relationships with other components.

Following, we list the steps that we consider necessary for designing this connection with the application.

-Specify schemes of problems that represent triggers for agent intervention.

-Specify schemes of special situations that represent triggers for agent intervention.

-Specify schemes of doubts that represent triggers for agent intervention.

-Specify schemes of suggestion requests that represent triggers for agent intervention.

The detection of these triggers produces that a communication from the agent to the student will be processed. From here, the items in the communication student's menu have two additional sources. The first, is the set of active tasks: each active task is associated to a set of expected events to go a step forward in its execution. Each of these expected events will have an associated entry in the student's menu, such as "*Perform X*", indicating that the student would like the agent execute the task named X, which is a subtask (a step) in the current task.

The second source of generated utterances is the interaction library. When the agent decides taking the communication initiative with the student, the *algorithm for selecting interaction type* chooses one from this library. This selected interaction wills has an associated visualization that is the item added to the student's menu. Furthermore, we must take into account that each interaction has associated a set of student possible answers. Therefore, when the student selects an interaction from the menu, all the possible answers associated to it are added to the utterance generator. Notice that now again, if the student selects an answer to the assistant agent dialog, it may have a set of associated answer interventions, and so on.

From the exposed facts, take place the need of having a tool that allow us to design the talks between the student and the assistant agent, i.e., that allows specifying the responses associated to each intervention. Finally, the utterance manager includes in the student's menu a set of special entries related to help the student in the execution of the task she is pursuing, such as "*Where am I?*", "*What should I do next*", etc. ().

Figure 8 shows an example of automatically generated answers from an agent remainder.

7 Conclusions

In this paper, a solution for designing the connection between an educational application for distance learning and an assistant agent was introduced. Two design points were defined in which the capture of student's intentions can be supported by agents and used for specifying their intervention.

The impact of our work can be measured through our instantiation of this proposal, which was made on a SAVER educational platform. The connection has been materialized without making valuable alteration on the source code.

References

Amandi, A., and Armentano, M. (2003) Connecting Web Applications with Interface Agents. Special Issue on Internet Agents for the International Journal of Web Engineering and Technology.

Ausubel, D., Novak, J., Hanesian, H. (1983). Psicología Educativa: un punto de vista cognoscitivo. Ed. Trillas. México.

Campo, M., Berdun, L., Armentano, M., Ruiz, M., Scandroli, A., Moroso, F. (2002a). Pataforma SAVER para educación a distancia. Reporte Técnico TR001-2002, ISISTAN Research Institute, UNICEN.

Campo, M., Diaz Pace, A., and Zito, M. (2002b). Developing Object-Oriented Enterprise Quality Frameworks using Proto-Frameworks, Software Practice and Experience Theme Issue on Enterprise Frameworks, Wiley & Sons, 32, pp. 837--843.

Chen, L., and Sycara, K. (1998). Webmate : A personal agent for browsing and searching. In Proceedings of the Second International Conference on Autonomous Agents, pp. 132--139, Minnepolis, USA, ACM Press.

Diaz Pace, A., and Campo, M. (2001). Analizing the Role of Aspects in Software Design. Communications of the ACM, Special Issue on Aspect-Oriented Programming, 44 (11).

Diaz Pace, A., Trilnik, F., Campo, M. (2001). Applying Proto-Frameworks in the Development of Multi-Agent Systems. Inteligencia Artificial, v. 13.

Eisenstein, J., and Rich, Ch. (2002). Agents and GUIs from Task Models. In Proceedings of the 7th international conference on Intelligent user interfaces.

Franklin, D., Budzik, J., and Hammond, K. (2002). Plan-based Interfaces: Keeping Track of User Tasks and Acting to Cooperate. In Proceedings of the 7th international conference on Intelligent user interfaces.

Godoy, D., and Amandi, A. (2000). PersonalSearcher: An Intelligent Agent for Searching Web Pages. Advances in Artificial Intelligence. Lectures Notes in Artificial Intelligence, v. 1952, Springer, pp. 43--52.

Guey-Fa, Ch. (1993). Some potential areas of research and development in the space of computer-based learning. Educational Technology, 33 (8).

Jonassen, D. (2002). Design of Constructivist Learning Environments (CLEs). Available in http://www.coe.missouri.edu/~jonassen/courses/CLE/index.html., 2002.

Johnson, W.L., Shaw, E. (1997). Using Agents to Overcome Deficiencies in Web-Based Courseware. AI-ED'97 Workshop on Intelligent Educational Systems on the World Wide Web.

Johnson, W.L., Rickel, J.W., Lester, J.C. (2000). Animated Pedagogical Agents: Face-to-Face Interaction in Interactive Learning Environments. International Journal of Artificial Intelligence in Education 11, pp. 47--78.

Lieberman, H. (1995). Letizia: An agent that assists web browsing. Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI'95), Canada. Morgan Kaufmann.

Lieberman, H. (1997). Autonomous Interface Agents. Proceedings of the ACM Conference on Computers and Human Interfaces.

Lieberman, H. (2001). Interfaces that give and take advice. In John Carrolls (Eds.). Human-Computer Interaction for the New Millenium. ACM Press/Addison-Wesley, pp. 475--485.

Maes, P. (1994). Agents that Reduce Work and Information Overload. Communications of the ACM, 37 (7).

Mayes, T. (2000). Pedagogy, Lifelong Learning and ICT. A discussion paper for the IBM Chair presentation, http://www.ipm.ucl.ac.be/ChaireIBM/Mayes.pdf..

Pannu, A., and Sycara, K. (1996). A learning personal agent for text filtering and notification. Proceedings of the International Conference of Knowledge Based Systems.

Paterno, F., Mancini, C., and Menicori S. (1997). ConcurTaskTrees: A diagrammatic notation for specifying task models. In S. Howard, J. Hammond, and G. Lindgaard, (Eds.), Human-Computer Interaction INTERACT, Chapman and Hall, pp. 362--369.

Peal D., and Wilson B. (2001). Activity Theory and Web-based Training. En Web-based training, Khan B. (Eds.) Englewood Cliffs NJ: Educational Technology Publications, pp. 147--153.

Rich, Ch., and Sidner, C. (1998). COLLAGEN: A Collaboration Manager for Software Interface Agents. TR-97-21a.

Rich, Ch., Sidner, C., and Lesh N. (2001). COLLAGEN: Applying Collaborative Discourse Theory to Human-Computer Interaction. AI Magazine, 22 (4).

Shaw E., Johnson, W.L., and Ganeshan, R.. (1999). Pedagogical Agents on the Web. Proceedings of the Third International Conference on Autonomous Agents, pp. 283--290.

Wooldridge, M. (1999). Intelligent Agents. In Weiss G. (Eds). Multiagent Systems, MIT Press.

Biographic information about authors.

Analía Amandi received a PhD. Degree in Computer Science in the Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil in 1997 and the Computer Science degree at the UNLP University, La Plata, Argentina in 1990. Currently she is a Professor at Computer Science Department and head of the agent group of the ISISTAN Research Institute of the UNICEN University at Tandil, Argentina. She has over 30 papers published in main conferences and journals about agents.
Her research interests includes intelligent agents and software architecture.

Marcelo R. Campo received a PhD. Degree in Computer Science in the Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil in 1997 and the Systems Engineer degree at the UNICEN University, Tandil, Argentina in 1988. Currently he is an Associate Professor at Computer Science Department and Head of the ISISTAN Research Institute of the UNICEN University at Tandil, Argentina. He is also a research fellow of the National Council for Scientific and Technical Research of Argentina (CONICET). He has over 50 papers published in main conferences and journals about software engineering topics.
His research interests includes intelligent aided software engineering, software architecture and frameworks, agent technology and software visualization.

Marcelo Armentano He is a PhD student. He obtained the Systems Engineer degree at the UNICEN University, Tandil, Argentina in 2003. Currently he is an Assistant Professor at Computer Science Department of the UNICEN University at Tandil, Argentina.
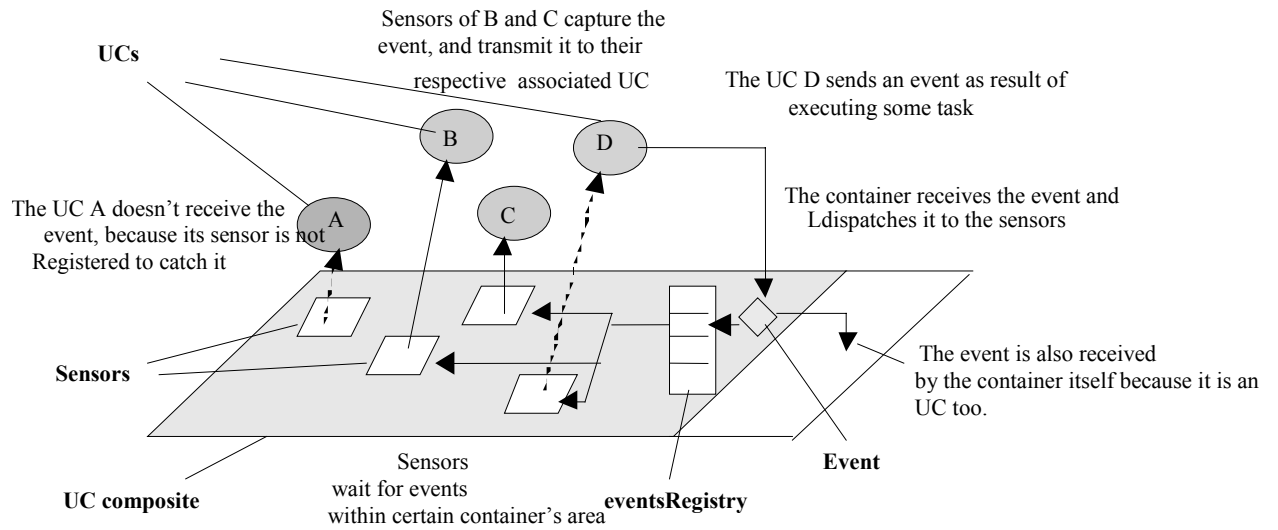His research interests includes intelligent agents and software architecture.

Luis Berdún He is a PhD student. He obtained the Systems Engineer degree at the UNICEN University, Tandil, Argentina in 2002. Currently he is an Assistant Professor at Computer Science Department of the UNICEN University at Tandil, Argentina.
His research interests includes intelligent agents and software architecture.
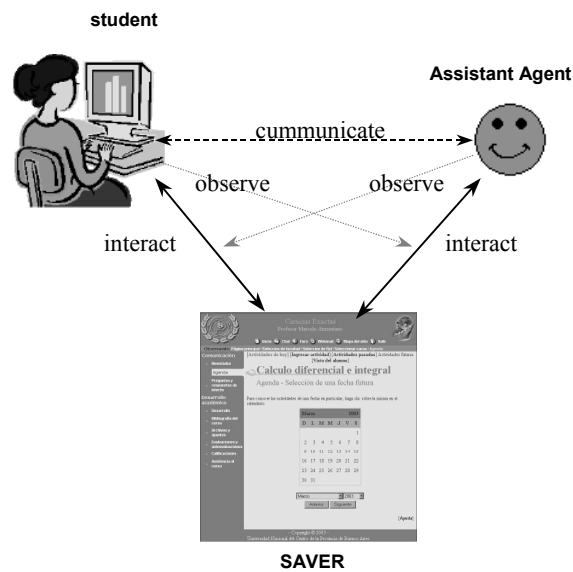
Figures

Figure 1: the SAVER design.

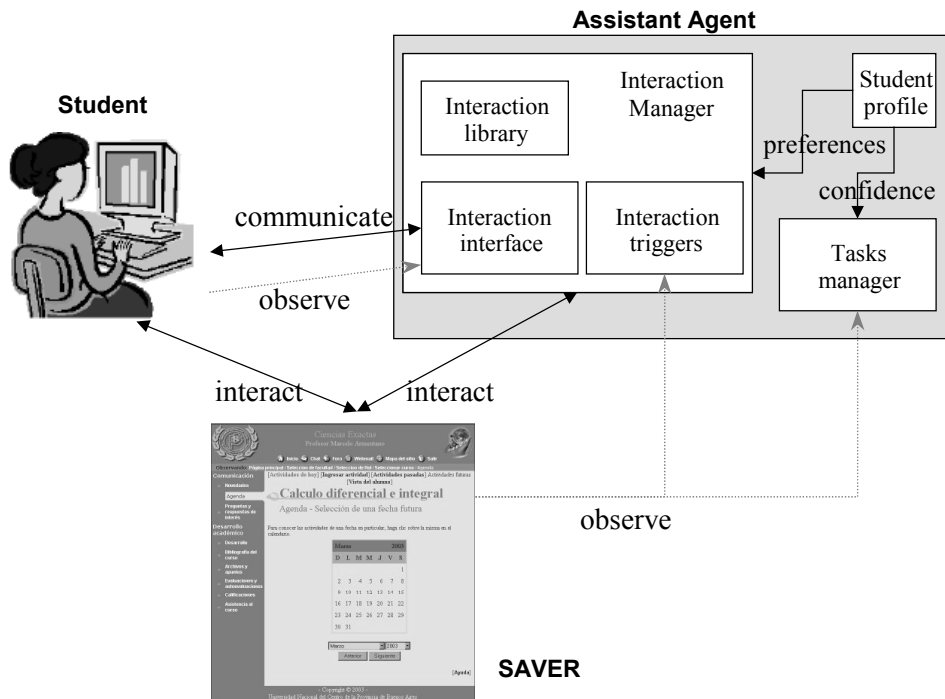**Figure 2:** A student and a computational assistant working on an education application.
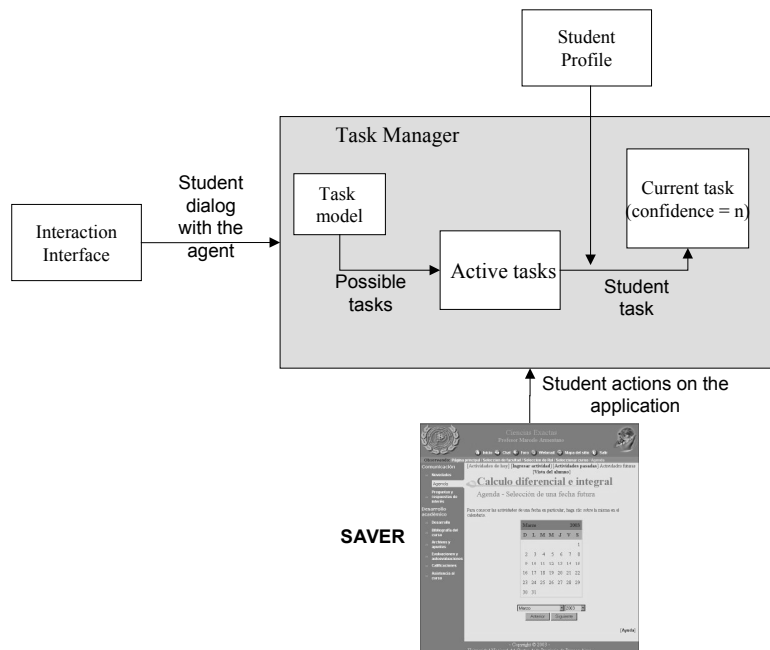
Figure 3: The connection architecture overview.

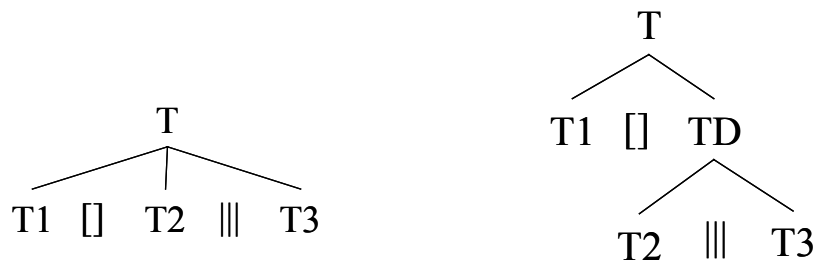
Figure 4: The Task Manager Component

a) An example of ambiguity   **b)** Possible solution to the ambiguity problem
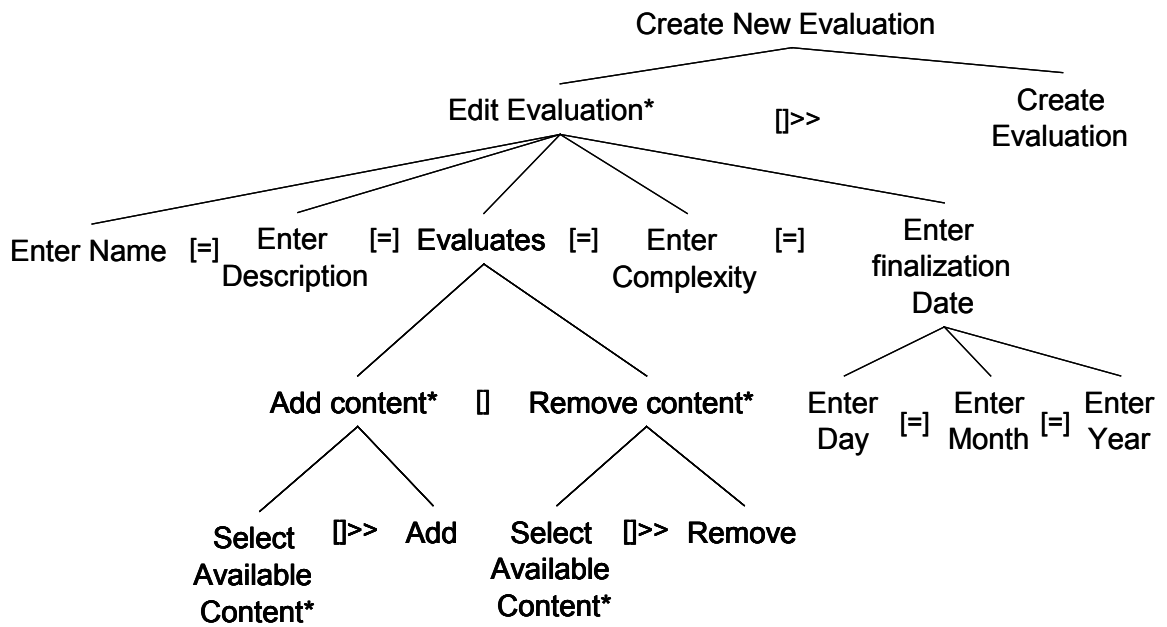
Figure 5: The ambiguity problem

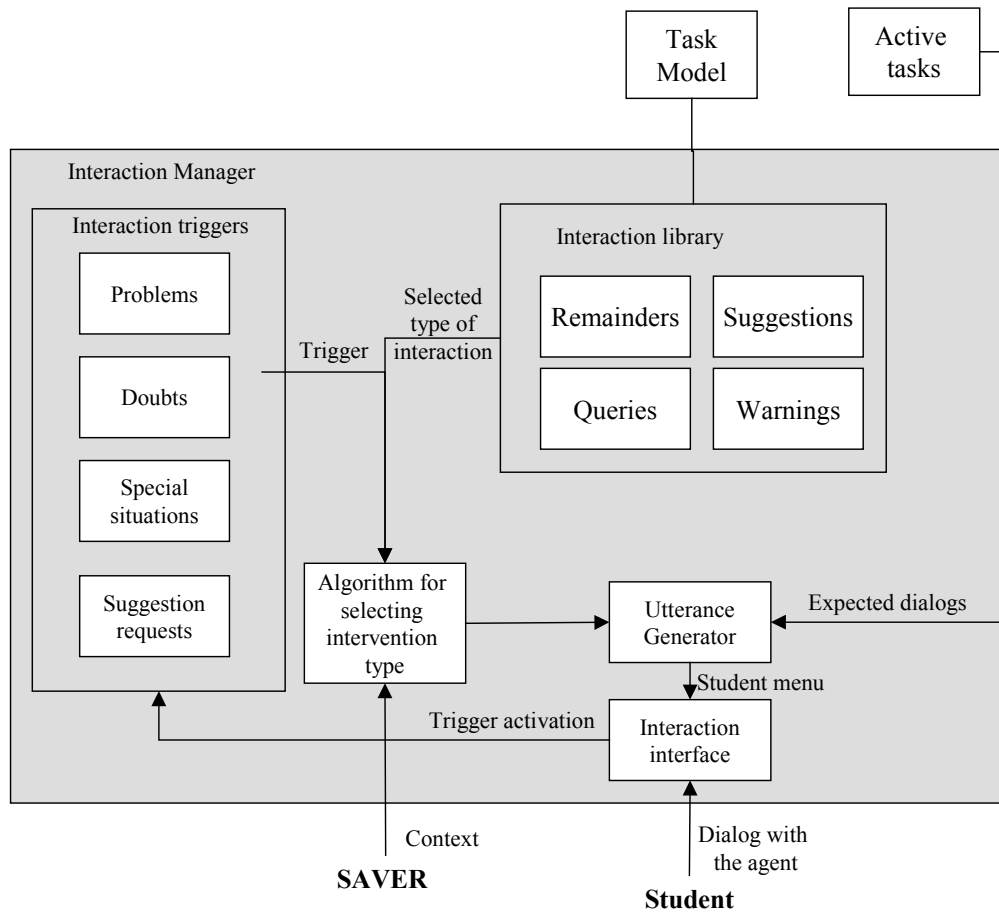**Figure 6:** An example of a task model of our educational application SAVER

Figure 7: Interaction Manager Component

Assistant Agent: *Remember that you must to log in before complete an exam.*

Student: (options)

- *Remember me it later.*

- *¿How do I perform the log in task?*

- *OK, let's go to the login page*

Figure 8: Automatically generated answers example