Intelligent Tutoring System for MASCARET.

Cédric Buche, Ronan Querrec
Brest Centre Européen de Réalité Virtuelle,
BP 38 - 29280 Plouzané
FRANCE
[buche, querrec]@enib.fr

Abstract: This article takes place in the formalization of processes for the development of differentiated virtual environments for training. It defines the implementation, by the way of agents, of basic models of Intelligent Tutoring Systems, considering the training of decisional behaviour within collaborative procedures. It specifies the behavioural and functional features linked to the procedural problematic and to the use of immersive virtual environments. It also details the interactions between the agents, necessary to implement a pedagogical agent able to assist the teacher.

Keywords: Virtual Environment for Training, Multi-Agents Systems, Collaborative Work, Pedagogy.

1 Introduction.

Our works focus around the use of the virtual reality for the implementation of a differentiated pedagogy focused on the decisional behaviour training for procedural and collaborative work. We build a teaching environment where a teacher has in charge several learners. In this context, every student is immersed in a virtual environment, simulating his physical environment and realizes his social environment [Que04].

We distinguish the role of the teacher, specialist in the domain, of role of the pedagogue, expert in education. In our system, the teacher is in communication with the learner and his charge increases with the number of learners. Therefore, our objective is to simulate a pedagogical reasoning, in order to suggest pedagogical assistances to the teacher, adapted to the simulation context and to the learner. For that, the system analyzes interactions between learner and virtual environment and interacts with the teacher in order to choose the most relevant pedagogical assistance (figure 1).

This approach divides our work in two parts:

1 First, we provide an "informed virtual environment" where the knowledge on the learner, the domain, the interface (system / learner interaction) and the pedagogy are represented, and updated online. For that, we propose an Intelligent Tutoring System (ITS). Our model defines our ITS items in the form of a multi-agents system.
2 Second, we define a pedagogical agent providing an educational reasoning, considering the knowledge of our ITS.

The object of this paper is the first part of our work.

Many applications integrate an ITS in a virtual environment for training. [Lev03] or [Pop04] are two good examples of such an environment. However, these two applications are focused on the simulation of the environment and do not incorporate explicit knowledge for education reasoning. [Ric99] incorporate a knowledge representation but only of the domain and on the learner. The most evolved propose a diagnosis model [Lou01], where the error notion and the proposed pedagogical assistances are specified for every exercise, and therefore are not generic.

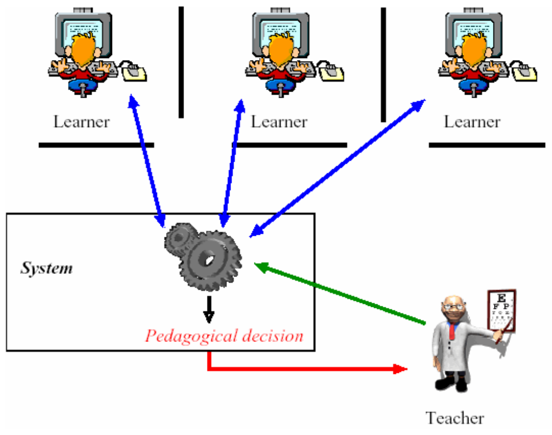

Figure 1: The system interacts with the learners and the teacher.

2 The models.

In this section, we show how we represent the knowledge used by the pedagogical reasoning of our system. These knowledge are contained in models inspired from classical ITS [Buc04]:

1. the *domain model*, representing the expert knowledge on the domain, useful for the pedagogy;
2. the *learner model*, allowing to establish the state of his knowledge, at a given instant, as well as his profile;
3. the *errors model*, proposing to identify the errors and containing a knowledge base on classical errors;
4. the *interface model*, allowing the information exchange between system and users;
5. the *pedagogical model*, taking the teaching decisions.

Models are based on an abstract model (`AgentPackage`) considering an agent (`Agent`), behaviours (`Behavior`) and knowledge (`KB`). Every agent (`InterfaceAgent`, `LearnerAgent`, `ExpertAgent`, `ErrorAgent`, `PedagogicalAgent`, `TeacherAgent`) is in interaction with one or several agents, and consequently able to update its knowledge and to adapt its behaviours.

First, we present the functional links between models, allowing to update knowledge and to define a process bringing to a pedagogical decision. Next, we detail the domain model, the learner model, the errors model and the interface model. They provide the knowledge used by the pedagogical model to make a pedagogical reasoning. The pedagogical model is not the scope of this article.

The principal objective of the system is to propose pedagogical assistances to the teacher according to the learner activities and to the simulation context. For that, each agent maintains its knowledge using information exchange with other agents.

2.1 The interactions between models.

An agent is associated with a model. The teacher intervenes using the `TeacherAgent`. Models interact and exchange data extracted from the simulation or deducted from an internal reasoning. We propose a process in 5 steps, to be performed for each learner (figure 2):

1. *Observe* (`InterfaceAgent`): using the interface model, the system analyzes the learner activities. Relevant elements for the training are provided to the learner model: learner actions (`PCVAction` see section 2.5), elements observable by the learner (`PCVObservation` see section 2.5) and learner motions (`PCVMotion` see section 2.5).
2. *Detect and Identify* (`LearnerAgent`, `ExpertAgent`, `ErrorAgent`): the system analyzes the learner actions (learner model) and compares them to actions to carry out (domain model). This confrontation allows to detect an eventual error. If an error is detected, an error identification mechanism acts (using the errors model). Moreover, the custom rules of the domain model are checked (independent rules of the procedure scheduling).
3. *Propose pedagogical assistances* (`PedagogicalAgent`): using the learner model (features, activities, error) and the domain model (knowledge on organisational structures), a mechanism simulating a pedagogical reasoning recommends the pedagogical assistances adapted to the context.
4. *Pedagogical assistance choice* (`TeacherAgent`): the teacher selects a pedagogical assistance among those recommended.
5. *Represent the pedagogical assistance* (`InterfaceAgent`): the selected pedagogical assistance is presented to the learner in the virtual environment.

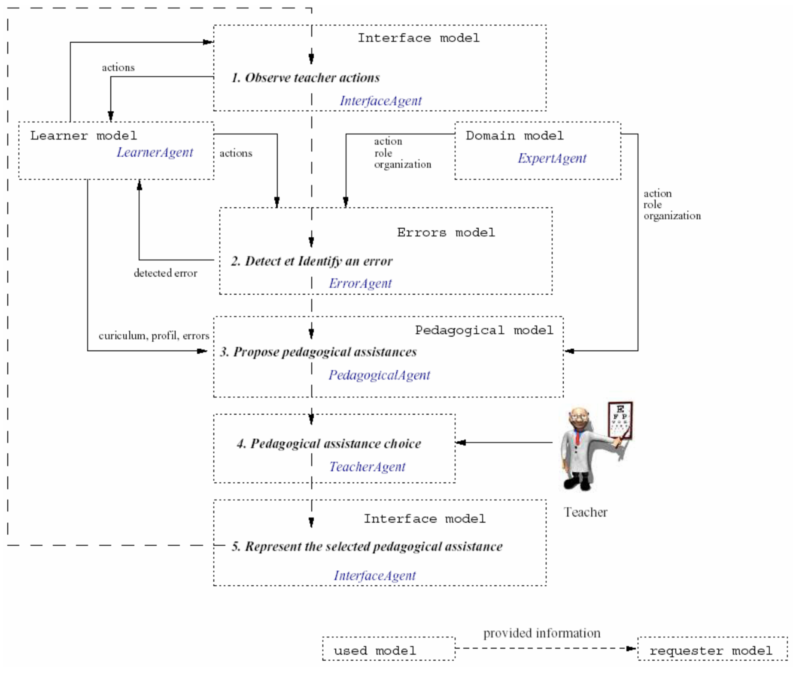

Figure 2: The pedagogical process is a 5 steps cycle.

2.2 The domain model.

The domain model is a knowledge representation the learner has to acquire; it represents the set of the expert knowledge on the learning domain. The system is used by the system to verify if the actions carried out by the learner are correct. It is also used to propose solutions or explanations to learner problems. Finally, it contains information allowing to interpret knowledge, in order to allow the action execution.

Knowledge base

In the case of the collaborative procedures training, these knowledge concern the social environment and the procedures. The domain model we propose is based on the organization model of `MASCARET` [Buc04]. We distinguish knowledge on:

1 *Social environment.*
The knowledge on the social environment is a set of expert knowledge on teams. Every expert knowledge uses the knowledge on the organisational structure of the model (Team in `MASCARET`), allowing to have a representation of roles, hierarchical relations between roles, and responsibilities within the procedures.

2 *Procedures*.
The knowledge on the procedures allows to define the actions scheduling.

3 *Custom rules*.
The domain also contains additional rules, independent from the procedures (`Rule`) (figure 3). Althought not part of the scheduling, these rules must be respected.

Knowledge representation and interpretations

1 *Teams and procedures*
The knowledge representation on teams and procedures uses the organisational structure defined in `MASCARET`. Thus, an agent soliciting the agent maintaining the domain model (`ExpertAgent`), does not ask any specific question (for example: Is it necessary to do the action “to walk towards the object O”?), but asks a generic question (for example “which action have to be done, after the action B, in the mission M1, considering the role R1 in the organization O1?”). The representation of the organization concept in `MASCARET` allows carrying to do such reasoning.

2 *Actions and custom rules*
Actions pre/post conditions and custom rules are represented by logical rules. Since we wish to get explanations on procedures, actions or custom rules in plain English, the represented knowledge must have an interpretation capacity and a description close to natural language. Consequently, the rules representation in the knowledge base uses a symbolic formalism and the interpretation uses a reasoning method based on a forward or backward chaining, depending on the explanations awaited. Rules constitute a knowledge base and are represented in order to be used by an explanations motor. We propose a representation of rules by Prolog terms composites (`LogicalTerm`). A rule can be evaluated, and the terms evaluation allows to generate an explanation.

Behaviors

The `ExpertAgent` proposes reactive behaviours (figure 3) linked to the following elements:

1 Social environment (`OrganisationBehavior`). It has the ability of reasoning on agents and roles. The expert agent is able to inform on the responsibilities of each. Considering the knowledge on the social environment, it replies to the question "is the action A is the responsibility of the role R?".

2 Procedures (`ProcedureBehavior`).

3 The MASCARET model proposes a procedure model (actions scheduling). As `STEVE` [Ric99], the expert agent is able to provide explanations. It can provide the actions scheduling as well as every step goals (procedure / action).

4 Actions (`ActionBehavior`).

5 The `MASCARET` model proposes an actions model subjected to pre/post-conditions. They are represented under the form of rules. As `STEVE`, the `ExpertAgent` is able to provide explanations and to act in place of the learner. It can provide the pre/post-conditions and explain why they are (or are not) satisfied.

6 Domain rules (`RuleBehavior`).

7 The `ExpertAgent` has the possibility to provide explanations on the "custom rules" (independent of the procedure) (`Rule`) using the rule terms representation.

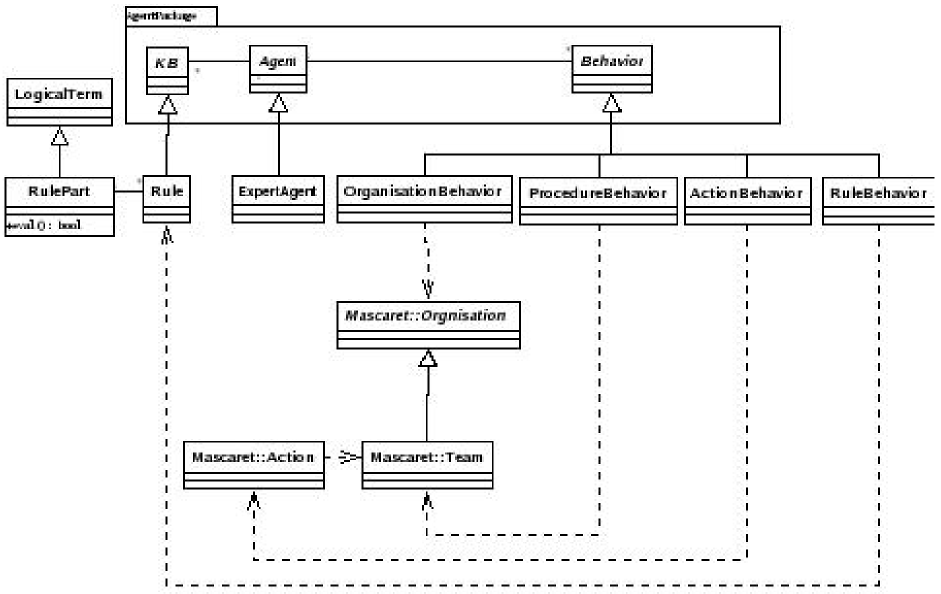

Figure 3: UML diagram of the expert model.

2.3 The errors model.

The objective of the errors model is to represent the knowledge allowing to explain why there is error, and to associate elements being able to be used to facilitate error comprehension.

Knowledge Base

The knowledge concerns a generic characterization of the errors allowing to detect them, and to provide to the pedagogical agent means to react (explanation, warning ...).

Knowledge representation and interpretations

The objective, exposed previously, brought us to tag errors. The different tags, represented in figure 4, distinguish errors on custom rules (`UsageError`), on roles (`TeamError`: team role), on procedure (`ProceduralError`: scheduling) and on actions (`ActionError`: pre-conditions).

Additional information, considering classical domain errors, can be written by the teacher and take the form of a rule. Then, the knowledge is represented specifying features of these classical errors (left part of the rule) and additional associated information (right part of the rule). A rule example is "if (the student do the action C after the action A); (this is a procedure error because Z)". "Z" being a rule constituted of a text and pointing to virtual environment entities (Object3D).

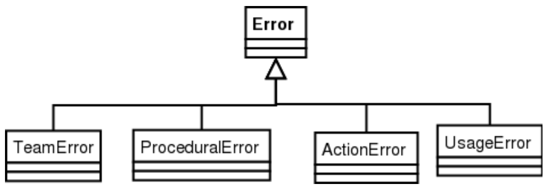

Figure 4: Different errors tags.

Behaviors

The `ErrorAgent` have the following behaviours (figure 5):

1 *Detect and tag errors* (`ErrorDetectionBehavior`) :

It detects a potential error carried out by the student:

- It verifies if the action solicited by the learner is feasible (pre-conditions) (`ActionError`).
- It compares the possible actions allowing to advance of one step in the procedure (provided by the domain model) and the action solicited by the learner (provided by the learner model). Information on these actions are recovered using the `ErrorAgent` and the `LearnerAgent` associated to the learner. It has to verify if the solicited action is in the possible actions (`ProceduralError`). It also assures the agent in charge of the solicited action has the ability (defines by its role) to carry out the solicited action (`TeamError`).
- It verifies the custom rules of the domain model, using a facts base on the virtual environment, provided by the `InterfaceAgent` (`UsageError`). When the `ErrorAgent` detects and tags an error, it transmits it to the pedagogical agent.

2 *Recognize a classical error of the domain* (`ErrorAssociationBehavior`) :

Our errors model contains a knowledge base on classical errors carried out by a student, in the concerned domain. It can be compared to the "Buggy model" present in other ITS [Brow78]. The `ErrorAgent` is able to recognize such an error comparing the error to the left part of the rules (corresponding to classical errors in the knowledge base). Then, it enriches the error (previously taged) with additional elements (`Concept_on_Error`).

3 *Auto enrich*: (`EnrichBehavior`)

The `ErrorAgent` records information on the errors frequency in order to inform the teacher. A mechanism, under the control of the teacher, uses such information to enrich the knowledge on the classical errors of the domain.

Communication

When the `InterfaceAgent` detects a new action from the learner, it informs the `ErrorAgent`. It analyzes the last action of the learner, soliciting the `LearnerAgent` and consulting the `ExpertAgent` to know the action to carry out. The `ErrorAgent` instantiates an Error and communicates it to the `LearnerAgent`

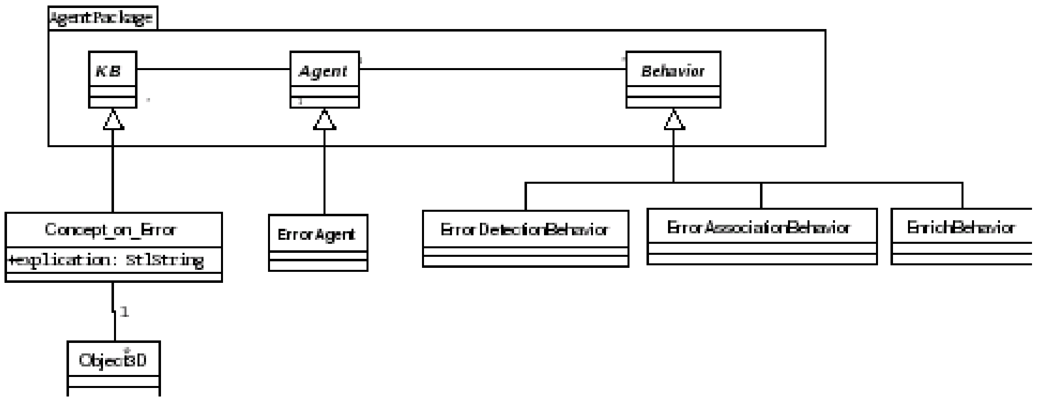

Figure 5: UML diagram of the errors model.

2.4 The learner model.

The learner model has to represent information characterizing the student from the prototypic point of view (student's profile) and from a curriculum point of view (student progression). The objective is not to model a virtual human, but to get information in a training framework.
The objective is to provide a differentiated learning in virtual environments for multi-learners. Consequently for every learner, an `InterfaceAgent` and a `LearnerAgent` are used. The `InterfaceAgent` updates the learner model indicating learner activities.

Knowledge Base
We use the distinction proposed by Delestre [Del00], separating the epistemic part and the non epistemic part:

1 *The epistemic part* (`EpistemicKB`) provides information on the state of learner knowledge for notions present in the domain. It contains a representation of steps and errors done by the learner. These information are dynamically updated and are available using the `LearnerAgent`. As suggested by Lourdeaux [Lou01], our learner model contains a light domain model representing the carried out steps. This is the overlay method [Sta76]. Then we make the simplifying assumption, considering actions reflect the learner knowledge. Moreover, it seems relevant to add information on the carried out errors. Learner actions and errors are stored following the sequence of simulation, and then we obtain his curriculum.

2 *The non epistemic part* (`NonEpistemicKB`) proposes a representation on information from cognitive engineering:

- *psychology*, this knowledge is from an analysis of a form filled before the exercise. It defines the learning strategy privileged by the learner;
- *physiology*, it contains the learner auditory and visual features. These information will be used by the `PCVObservation` (to see models interface 2.5);
- *memory* of the virtual elements observed by the learner, in an instantaneous way (`sensorialTimeMemory`), or in a short time (`shortTimeMemory`).

Behaviors

1 The `LearnerAgent` updates dynamic information concerning the learner activities (figure 6):

- `CurriculumBehavior`: new action / error;
- `MemoryBehavior`: objects having physical representation and the learner could observe. These information are updated using the `PCVObservation`.

2 The `LearnerAgent` replies to others agents requests (`InformBehavior`) concerning the learner activities (example: last action achieved) or learner features (example: privileged strategy).

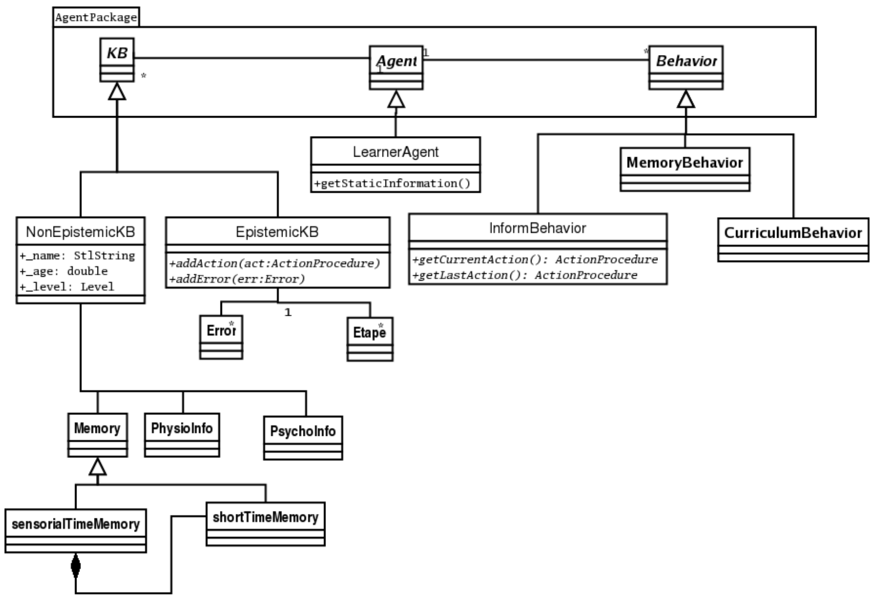

Figure 6: UML diagram of the learner model.

2.5 The interface model.

The interface model is in charge of the bidirectional communication between the learner and the system. In a training framework, the system has to offer to the learner the ability to interact with his environment and to analyze its activities. We consider the virtual behaviour primitives, called PCV, from Fuchs [Fuc03]. Fuchs gathers PCV in four categories:

1 to observe the virtual world;
2 to move in the virtual world;
3 to act in the virtual world;
4 to communicate with others.

Knowledge Base

Our interface model contains a knowledge base on the learner (`LearnerInformationKB`) allowing to take into account its personal features. These information is from the dialog between the `InterfaceAgent` and the `LearnerAgent` (learner model).

Behaviors

The `InterfaceAgent` possesses three behaviours corresponding to every PCV. We did not take care of the communication PCV, because the knowledge at this point seems delicate to exploit pedagogically at the moment. The `PCVAction` lets the learner to interact with objects of the environment and detect the learner actions. The `PCVObservation` and the `PCVMotion` analyze its activities.

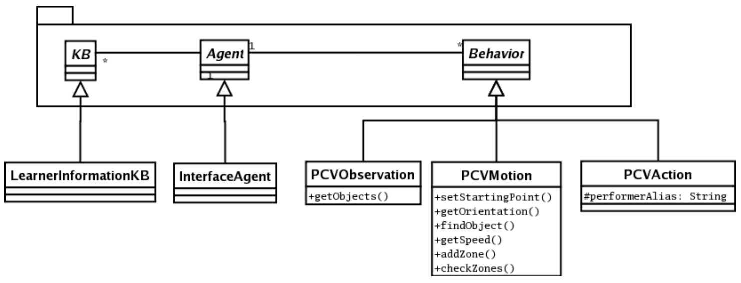

Figure 7: The package PCV is constituted of three parts: observation, movement and action.

1 `PCVAction`

The `PCVAction` offers means of interactions between the learner and the environment, and detects the actions solicited by the learner:

The means of interactions

First, the system has to propose to the learner means of interactions with the objects of the virtual environment corresponding to its decisions. This interaction can be regulated adapting itself to the learner (using the knowledge based on the learner and soliciting the `PedagogicalAgent`). In fact, we can think it is not necessary to overload a novice learner proposing unreachable decisions at a moment, while it can be interesting for a experimented learner. To take a decision, the learner solicits physical objects. They have knowledge on the actions they can carry out. They are controlled by graphs of states. When the learner solicits an action on an object:

- the `PCVAction` recovers the object internal information (possible actions, state running, reachable states),
- next, it asks `PedagogicalAgent` actions to propose, specifying of internal information and the learner features,
- the `PCVAction` posts the proposed actions under the form of a 3D menu.

The solicited actions

Next, the system has to detect the solicited actions and the carried out by the learner, in order to update the learner model. The learner actions on virtual environment objects can be varied (rotation, translation, deformation, etc). Nevertheless, our work framework is not the training to technical gesture, but to decisional behaviour. Therefore the system has to detect and to identify decisions. The learner obtains the possibilities of actions using `PCVAction`. A choice between different solutions is offered to him. Then it takes his decision and solicits an action. The `PCVAction` gets the information and transmits it to system. The `LearnerAgent` gets the information. After an analysis by the `ErrorAgent`, the `PedagogicalAgent` chooses to authorize or not the action. In fact, it can consider interesting to carry out an action, even if there is an error, for some learner. The `PCVAction` indicates if the action is done or not and informs the `LearnerAgent`.

2 `PCVObservation`

It is interesting to know the elements present in the learner vision field to take a pedagogical decision. In fact, consider for example the pre-condition of an

action A needs the presence of the object B. The pedagogical assistance must take into account the learner has the opportunity to observe this object or not.
The observation PCV gets a list of objects appearing in the user vision field, as well as their distances and their orientations to the user. Although simplistic, we made the assumption a visible object is perceived by the learner. In order to adapt the system to all users, the observation PCV takes into account the learner model to obtain physiologic data (user depth and width of vision, contained in `LearnerInformationKB`). The `LearnerAgent` updates the learner model (`Memory`) calling the `PCVObservation` services.

3 `PCVMotion`

It is not necessary to always know the accurate position of the learner in the environment. Yet it is useful knowledge, the learner moves towards the place where the next action takes place, if the learner approaches to a precise object and with which speed. The movement PCV replies to these needs permitting:

- to get the user average orientation from a starting point;
- to know the object towards the user globally is going to;
- to know instantaneous user speed;
- to get knowledge on distances (fuzzy variables: outside, far, near, inside) between zones and the user;

The movement PCV will send information on zones, direction and speed in permanence to the `LearnerAgent` allowing to update the learner model.

3 Conclusion and future works.

We presented the models of our ITS containing the knowledge base used by our system pedagogical reasoning. We detailed the domain model, the learner model, the errors model and the interface model. We presented the dynamics between the models, allowing to update the knowledge of each and we defines a process bringing to a pedagogical decision. Our ITS provides a knowledge base allowing to make a pedagogical reasoning.

The objective is to propose to the teacher pedagogical assistances adapted to the learner. For that, the pedagogical agent (`PedagogicalAgent`) constructs a knowledge base, called the “pedagogical situation”. It corresponds to relevant elements to take into account to choose pedagogical assistances. It is constituted by elements present in models of our ITS previously described (error, domain, learner features and activities), increased by the action context (pre/post condition, resources …). Future works will concern the different parts of our approach (figure 8):

1 the automatic construction of the “pedagogical situation”;
2 the pedagogical model specifying the `PedagogicalAgent` behaviour (pedagogical rules selection) in order to propose to the teacher pedagogical assistances;
3 the artificial learning mechanism allowing the pedagogical model to take into account the teacher choices and to modify its suggestions consequently.

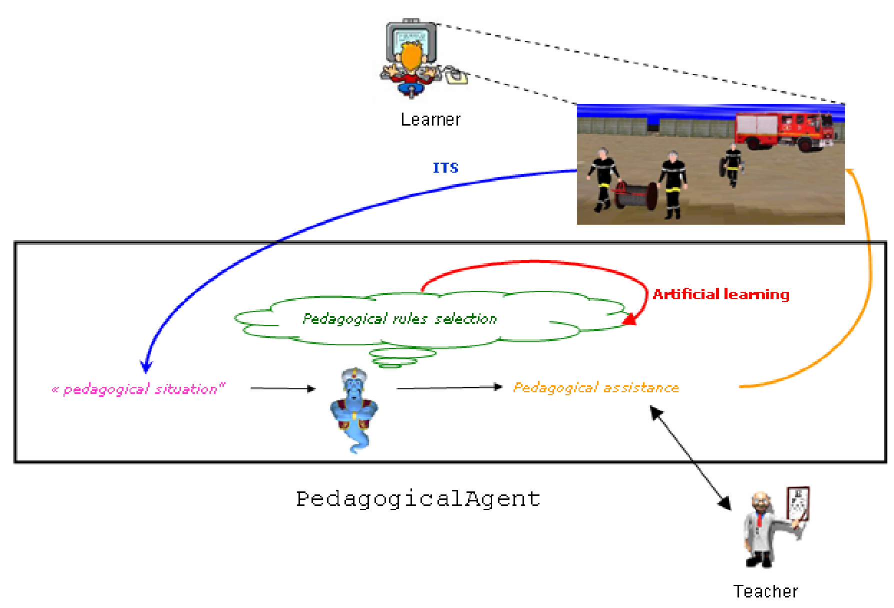

Figure 8: The different parts of the pedagogical agent (`PedagogicalAgent`).

4 Bibliography

[Bro78] Brown, J. S. et Burton, R. R.(1978). A paradigmatic example of an artificially intelligent instructional system. Int. *Journal of ManMachine Studies*, 10 :323339.

[Buc04] Buche, C., Querrec, R., DeLoor, P., et Chevaillier, P. (2004). MASCARET: A pedagogical multiagent system for virtual environment for training. *Special issue "Cyberworlds and education" of the international Journal of Distance Education Technologies (JDET)*, 2(4) :4161.

[Del00] Delestre, N. (2000). *METADYNE, un Hypermédia Adaptatif Dynamique pour l'Enseignement*. PhDthesis, Université de Rouen.

[Fuc03] Fuchs, P. et Moreau, G. (2003). *Le traité de la réalité virtuelle*. Presses de l'école des mines.

[Lev03] Levesque, P. (2003). Creation and use of 3d as built models at EDF. In *FIG Working Week 2003*.

[Lou01] Lourdeaux, D. (2001). *Réalité virtuelle et formation : conception d'environnement virtuels pédagogiques*. PhD thesis, Ecole des mines de Paris.

[Pop04] Popovici, D., Gerval, J.,Chevaillier, P., Tisseau, J., Serbanati, L., et Gueguen, P. (2004). Educative distributed virtual environments for children. *Journal of Distance Education Technologies*, 2(4) :18-40.

[Que04] Querrec, R., Buche, C.,Maffre, E., et Chevaillier, P. (2004). Multi-agents systems for virtual environment for training. Application to firefighting. *Special issue "Advanced Technology for Learner" of International Journal of Computers and Applications (IJCA)*, 1(1):25-34.

[Ric99] Rickel, J. et Johnson, L.(1999). Animated agents for procedural training in virtual reality: Perception, cognition, and motor control. *Applied Artificial Intelligence*, 13.

[Sta76] Stansfield, J., Carr, B., et Goldstein, I. (1976). Wumpus advisor 1. : a first implementation of aprogram that tutors logical and probabilistic reasoning skills. Technical report, Massachusetts Institute of Technology, Artificial Intelligence Laboratory.