Model for Verbal Interaction between an Embodied Tutor and a Learner in Virtual Environments

Ronan Querrec, Joanna Taoum, Bilal Nakhal, Elisabetta Bevacqua

Model for Verbal Interaction between an Embodied Tutor and a Learner in Virtual Environments

Ronan Querrec
Lab-STICC ENIB
Brest, France
querrec@enib.fr

Joanna Taoum
Lab-STICC ENIB
Brest, France
taoum@enib.fr

Bilal Nakhal
AUL
Beirut, Lebanon
bilalnakhal@gmail.com

Elisabetta Bevacqua
Lab-STICC ENIB
Brest, France
bevacqua@enib.fr

ABSTRACT

This research work introduce virtual embodied tutors in Virtual Environments for learning devoted to learning of procedures for industrial systems. We present a communicative behavior which, integrated in pedagogical scenario, permits on the one hand to realize the pedagogical communicative actions at a semantic level (e.g., the tutor explains the goal of an action) and on the other hand to realize such actions through human-like communicative channels (i.e., the virtual tutor's voice, facial expressions and gestures). The communicative behavior relies on a taxonomy of questions in order to interpret the learner's communicative actions and to generate the tutor's own questions.

CCS CONCEPTS

• **Computing methodologies** → **Intelligent agents**; **Virtual reality**;

KEYWORDS

Embodied Conversational Agent, Virtual Learning Environment, Verbal Interaction, Intelligent Tutoring System, Interface

ACM Reference Format:
Ronan Querrec, Joanna Taoum, Bilal Nakhal, and Elisabetta Bevacqua. 2018. Model for Verbal Interaction between an Embodied Tutor and a Learner in Virtual Environments. In *IVA '18: International Conference on Intelligent Virtual Agents (IVA '18), November 5–8, 2018, Sydney, NSW, Australia.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3267851.3267895

1 INTRODUCTION

The presence of Embodied Conversational Agents (ECAs) in Virtual Environment for learning has positive effects on the learner engagement and the effectiveness of teaching [7]. Experiments, like that presented in [11], show that using embodied tutors can motivate the user to accomplish the required tasks.

IVA '18, November 5–8, 2018, Sydney, NSW, Australia

ACM ISBN 978-1-4503-6013-5/18/11…$15.00
https://doi.org/10.1145/3267851.3267895

Several virtual tutors have already been developed (AutoTutor [3], TARDIS [6]) , but most of them focuses on social and emotional aspects of the tutor. We consider that a major functionality of a virtual tutor is its ability to reason, in a pedagogical context, about the domain and the learner's actions to guide and help him/her. Such capability to reason for virtual tutors is a fundamental subject that is studied in the domain of Intelligent Tutor Systems (ITSs) [13]. Currently, major works on ITS deals with artificial learning of the tutor behavior by the observation of the interactions between the teacher and the learners. But in order to validate the realization of a specific pedagogical goal, the trainer may want to control the way the ITS interacts with the learner. The classical way to do this is the use of pedagogical scenarios [8]. The major difficulty is to provide a language which is expressive enough to allow the trainer to write its scenarios and formal enough to be automatically interpreted by the ITS.

In this work, we develop a model for tutors in Virtual Environments for learning devoted to domain applications such as learning of procedures for industrial systems. In this paper we improve ITS models by adding a communicative behavior which can be integrated in a pedagogical scenario. The formalization of such a behavior permits on the one hand to realize the pedagogical communicative actions at a semantic level (e.g., the tutor explains the goal of an action) and on the other hand to realize such actions through human-like communicative channels (i.e., the virtual tutor's voice, facial expressions and gestures). In addition, the communicative behavior relies on a taxonomy of questions in order to interpret the learner's communicative actions and to generate the tutor's own questions.

To formalize the different components of the ITS models, we use Mascaret [2], a virtual reality meta-model based on the Unified Modeling Language (Uml). In section 2 we show how we use Mascaret to represent the domain model and the pedagogical scenario. Pedagogical scenarios are written by a trainer and they represent a predefined sequence of pedagogical assistances that guides the learner throughout the execution of a procedure. Pedagogical assistances provide information to the learner through verbal signals or through modifications of the virtual environment. To realize these pedagogical assistances in a more human-like way, we propose an interface model based on the concepts of communication behavior and communication actions realized by an ECA (see section 3). In section 4 we show how our tutoring model uses the communication

actions in order to infer the content of the student model to adapt the pedagogical strategy.

2 DOMAIN MODEL AND PEDAGOGICAL SCENARIO

The domain model represents the knowledge that the system aims to transfer to the learner. Ideally, the knowledge has to be set by the domain experts themselves. The difficulty is then to give to these experts a language that is expressive enough to write their expertise and formal to be automatically interpreted by the system. It is classical to use ontology like OWL or DOLCE to propose such languages. If those ontologies are strong enough to represent the static parts of the systems, it is hard to use them to describe the dynamic parts (components' behaviors, procedures). More, if these ontologies are strong enough to permits automatic reasoning on the domain concepts, it is not possible to generate automatically the execution of the dynamic part of the system in a virtual reality environment. We choose then to use Mascaret. Mascaret covers all the aspects of virtual environments semantic representation: domain's ontology, environment's structure, entities' behavior and both user's and agents' interactions and activities. The structure of the system is defined by the concept of `Class`, `Property` and `Association`, the entities' behavior by `StateMachine` and `Event` and the activities and interactions by the concepts of `Activity` and `Action`. A virtual environment using Mascaret is then able to read a Uml model and automatically execute it. The interest of such approach is that, as the language is formal, it is possible for an autonomous agent to make reasoning and, by giving an operational semantic to all the metamodel's concepts, it is possible to generate the execution of the system.

Moreover, in Mascaret, pedagogy is considered as a specific domain model. Pedagogical scenarios are implemented through Uml activity diagrams containing a sequence of actions. These actions can be either pedagogical actions, like explaining a resource, or domain actions, like manipulating an object. For the definition of pedagogical scenarios and actions, we rely on previous works [10].

In Mascaret five types of pedagogical actions are considered:

(1) Pedagogical actions on the virtual environment: highlighting an object, playing an animation.
(2) Pedagogical actions on user's interactions: changing the viewpoint, locking the position, letting the student navigate.
(3) Pedagogical actions on the structure of the system: describing the structure, displaying a documentation about an entity.
(4) Pedagogical actions on the system dynamics: explaining the objectives of a procedure, explaining an action.
(5) Pedagogical actions on the pedagogical scenario: displaying a pedagogical resource, making an evaluation (e.g. a quiz).

3 ECA AS INTERFACE

The global architecture of our system in presented on figure 3. In this section we focuses on the interface model. The main role of the interface model is to recognize the actions made by the learner and to realize the action selected by the tutor behavior or by the pedagogical scenario. It is important to notice that in Mascaret, any entity which acts on the environment is considered as an *agent*. Particularly, the ECA and the human user are *embodied* agents. We formalized, in the interface model, the basic actions that an embodied agent is able to perform:

(1) Communication actions (e.g. giving an information, answering, listening).
(2) Action realization: non-verbal multi-modal communicative signals (e.g. facial expression, gesture, gaze) and actions that can modify the environment (e.g. manipulating or highlighting an object).
(3) Navigation (e.g. observing, moving around).

These primitive actions are used to implement the pedagogical actions, presented in section 2, and the domain actions.

Our system is able to recognize the realization of each one of these actions performed by the user. In order to perceive the user, we connected our system to several devices (e.g. VR peripherals). For example, the VR controllers (e.g. HTC Vive, Oculus Rift) allows the user to act on the virtual environment by selecting and manipulating the virtual objects. The system can also receive information from a microphone and a camera. These data are elaborated by RealSense which can recognize some facial expressions, head orientation and voice intonation. The content of the sentences uttered by the user is parsed using Artificial Intelligence Markup Language (AIML)[1].

In this article, we are focusing on the communication actions. The main challenge for the interface model is to be able to automatically interpret and generate natural communication between the learner and the tutor. This can be divided in two main problems: (1) What formal representation for the content of the communication? This content comes not only from the sentence uttered by the learner but also from those uttered by the virtual tutor and (2) How to automatically execute the tutor's communication actions in a natural human-like manner?

We propose a model to formally represent communication actions that can be triggered by the tutor behavior but that can also be scheduled by a pedagogical scenario. To do this we add a specific Uml `Action` call `CommunicationAction` (see section 3.1). In section 3.2 we show how this action can be executed by an embodied agent (ECA).

3.1 Communication Action

The procedure to learn and the pedagogical scenarios are defined in Mascaret through Uml activities. An activity is a set of actions (from the meta class `Action`). The Mascaret meta-model (based on the Uml meta-model) defines several kind of actions (sub class of the meta class `Action`): `CallOperationAction` to realize a domain specific operation, `CallBehaviorAction` to realize a sub activity, etc.. All those actions can be triggered by an agent (a virtual tutor or a learner) while executing the pedagogical scenario, but they can also be triggered by an agent behavior (for example the tutor behavior explained in section 4). We created a `CommunicationAction` to formalize the communication between agents. The `CommunicationAction` class, which inherits from the `Action` class, is instantiated anytime an agent needs to communicate information to other agents. The `CommunicationAction` contains a text in natural language

[1] http://www.alicebot.org/

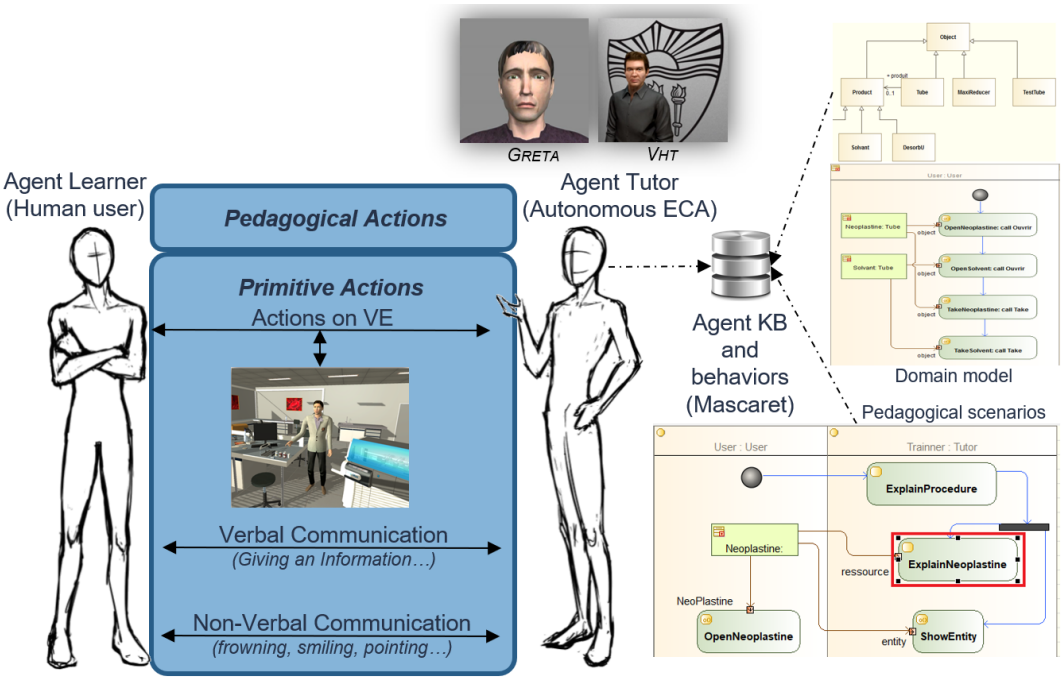

Figure 1: The representation of the interface model.

and/or a formal content in an agent communication language. If the action contains a text in natural language, such a text cannot be treated by the agent. It will be treated only by the SAIBA compliant virtual agent (see section 3.2). Among the existing agent communication languages, we have chosen to implement FIPA-ACL [2] as the communication protocol between autonomous agents, and FIPA-SL (Semantic Language) as the message content language. A FIPA-ACL message is first defined by a performative that specifies the objective of the FIPA-ACL message and then the speech act that an agent wants to deliver to another agent. For example, in our model, when an agent seeks for the value of an attribute of an object in the virtual environment, it must use the "QUERY-REF" performative. In our model, we also manage the sender, the receivers and the message content in FIPA-ACL messages.

The effective content of the message is formalized using FIPA-SL language, so we propose a FIPA-SL parser to automatically interpret the contents of the FIPA-ACL messages using the standard parsing rules defined by FIPA. The complete grammar (FIPASL.g4 in ANTLR) can be found in the FIPA normalization. In this article, we just take one typical example of message content to explain how it works in our model. Among the referential operators defined in the FIPA-SL, the "iota" operator allows an agent to ask for a term (property) inside the knowledge base of another agent. The value of a property is stored in MASCARET in a `Slot`. Therefore, when an agent receives a FIPA-ACL message and detects the "iota" operator in its content, it recognizes that it should get the `Slot` value of the specified entity. In the reply message, the "INFORM" performative is used to transmit the `Slot` value.

[2] http://fipa.org

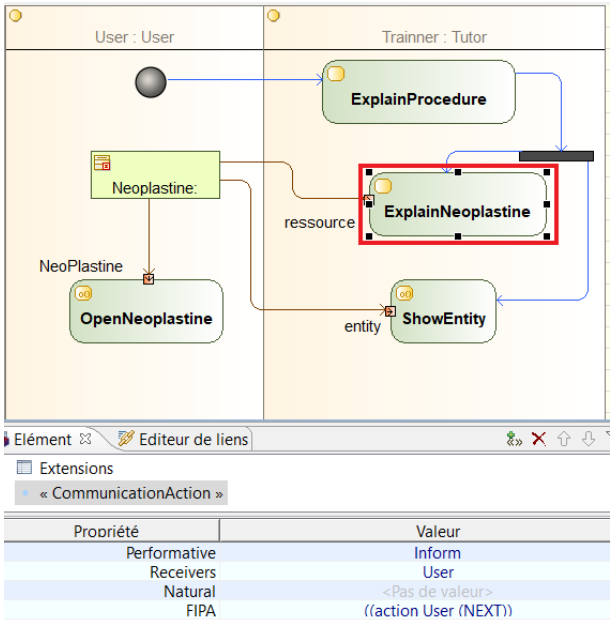

Figure 2: Creation of a communication in an activity in Modelio UML modeler.

As the `CommunicationAction` inherits from `Action`, it can be used in activities. UML activity diagrams are used in MASCARET to define scenarios and procedures through a set of actions. The UML normalization permits to create specific profiles by the

addition of stereotypes. Most UML modelers allow to create those profiles. In this work we use the Modelio modeler. Hence, to create the communication action in an activity, we create an `Action` and add a `CommunicationAction` stereotype to it (Fig. 2).

In MASCARET, agents can realize behaviors that represent the active part of the agent (e.g., to navigate in the environment, to take decisions, etc.). They are represented by functions that are executed every step of the simulation. An agent can execute one or several behaviors only once or cyclically and in parallel. We modified the existing agent behavior which permits to an autonomous agent to execute an activity in order to take into account the execution of `CommunicationAction`. The example in figure 2 represents a communication that is executed by an agent playing the role of *Tutor* and that explains the object that has to be manipulated by an agent (or a user) playing the role *User*. The agent that executes this action parses the FIPA-SL content of this communication action, gets the *Description* of the object, and transmits it to the user. In section 3.2, we presents how this message is automatically generated in natural language. To directly inform the user using a specified natural text, we set the *Natural* property in the `CommunicationAction` in the activity diagram. If an embodied agent is realizing the communication action, it takes the value of the *Natural* property, and vocally tells it to the user through an ECA.

Moreover, we have implemented a generic Communication Behavior, which is cyclic and executed in parallel with other potential behaviors. The goal of this behavior is to automatically manage the communication between agents. As previously mentioned, users are also considered in our model as agents. Communication behavior is defined according to the implementation of `CommunicationAction` (as described above) based on the communication protocol (FIPA-ACL and performative) and content (FIPA-SL syntax). In a virtual environment implemented using our model, the user can ask different types of questions. The user's questions that we take into account are primarily about the properties of the agents with whom the users interact, the entities they must manipulate, and the actions they must perform. Using the communication behavior, the agents respond to all inquiries of the user even when the received questions are not correctly formulated or are out of scope.

We do not take into account all the twenty two performatives of FIPA-ACL, we consider just two performatives (INFORM and QUERY-REF), since we focus on transfer of knowledge between agents. In order for an autonomous agent to be able to generate an answer to a question asked by a learner (and to ask the learner a question), we defined a taxonomy of questions. This taxonomy permits us to write, in an exhaustive way, the grammar to be interpreted by AIML. This taxonomy is based on the agents' reflection capability, provided by MASCARET, about their knowledge and it consists of the classical type of questions: Why, Where, Who, What. Table 1 shows some examples of the type of questions that a user (or an autonomous agent) can ask about three main concepts of MASCARET: `Entity`, `Agent` and `Activity`.

For each type of questions, we add several ways of uttering the question in the AIML grammar file but all of them refer to the same question's pattern as shown in Table 1. The example on figure 3 shows the transformation of the user's utterance to a FIPA-ACL message and the tutor agent's answer using this behavior.

Agt1 says: "What is *entityName*?"

AIML rule triggered:

```
<pattern>What is *</pattern>
<template>
((iota ?description
(slot ?description ?<star index="1"/>)
))
</template>
```

FIPA-SL received by Agt2:

```
QUERY-REF
Agt2
((iota ?Description (slot ?description
?entityName)))
```

FIPA-SL Reply from Agt2:

```
INFORM      Agt1      ((=(iota ?entity (slot
?description)) Description of entityName))
```

Figure 3: Interpretation of user's utterance.

If the agent is an embodied agent then the FIPA-SL content will be also uttered in natural language. The way this is automatically done is explained in next section.

3.2 SAIBA Integration

To embody the tutor, we integrate different ECA platforms which provide several virtual characters that can be displayed in diverse VR devices (e.g. PC screen, Head Mounted Display and CAVE) and which are able to select and perform multi-modal communicative and expressive behaviors in order to interact naturally with the user. We made it possible to integrate easily all ECA platforms which are compatible with the standard SAIBA framework [9][3]. This framework divides the generation of the virtual agent behavior in three levels of abstraction, the Intent Planner, which determines the agent communicative intentions, the Behavior Planner, which transforms the agent intentions in multi-modal signals and the Behavior Realizer, which realizes the multi-modal signals on the virtual agent representation. Each level communicates with the next one through standard languages, the Function Markup Language (FML) [5] and the Behavioral Markup Language (BML) [14]. The FML is used to encode the agent communicative intentions (such as greet somebody, ask a question, make a reproach, show sadness or happiness, refer to an object, etc.). Messages written following this language are composed in the Intent Planner module and sent to the Behavior Planner where the communicative intentions are translated in multi-modal behavioral signals. For example, the intention to explain the action to perform on an object in the virtual environment is translated in a sentence to utter and a gesture to point the object. These signals are encoded using the BML and sent to the Behavior Realizer. We allow the integration of SAIBA compliant virtual agents by defining two interfaces in MASCARET. The `BehaviorPlannerInterface` provides an abstract method called `parseIntention(CommunicationAction`

[3] http://www.mindmakers.org/projects/saiba

MASCARET Concepts	Question's pattern
Entity or Class	
Description	What is `EntityName`?
Slot description	What is `SlotName` of `EntityName`?
Operation	What can I do with `EntityName`?
Entity	
Slot Value	What is the value of `SlotName`?
Position	Where is `EntityName`?
Agent	
Operation	What can do `AgentName`?
Position	Where is `AgentName`?
Behavior (Action/Activity)	
Behavior/Role	What should I do?
Behavior/Role	What `AgentName` has to do?
Behavior/Role	Who has to do `ActionName`?
Behavior/PostCondition	Why?
Behavior/PostCondition	What to do in order to `PostCondition`?

Table 1: Extract of the taxonomy of questions.

`intentions)` that must be override by the concrete class which implements such an interface. The overridden method has to encode in FML the agent communicative intentions, received as parameter, and send the resulting message to the Behavior Planner of a specific agent platform. Similarly, the `BehaviorRealizerInterface` has an abstract method called `addBehavior(string signals)` that all classes which implement such an interface have to override. The overridden method has to encode in BML the multi-modal behavioral signals received as parameter and send the resulting message to the Behavior Realizer of a specific agent platform. Even though the FML and the BML aim at being standard languages, at present not all the existing agent platforms accept messages written in the same FML and BML format. For such a reason, we have to write a concrete class implementing the `BehaviorPlannerInterface` for each virtual agent platform that we connect at the Behavior Planner level and a concrete class implementing the `BehaviorRealizerInterface` for each virtual agent platform that we connect at the Behavior Realizer level. We integrated several platforms like(VIRTUAL HUMAN TOOLKIT [4]) which provide only a Behavior Realizer, so we connect them just at this level. In this case, a basic concrete class implementing the `BehaviorPlannerInterface` is provided. It translates the agent communicative intentions in a minimal set of multi-modal signals, such as speech and pointing.

We integrated also the GRETA platform [12]. This platform has both a Behavior Planner and a Behavior Realizer so we can connect MASCARET at both levels. However, we are interested to connect only the Behavior Planner of GRETA. The latter is powerful enough to select automatically a set of multi-modal signals which transmit the communicative intention received from MASCARET. In our model, the `GretaBehaviorPlanner` receives the communication action through the `ParseIntention` method. The goal of this method is to transform the communication action into a FML message and send it to the GRETA platform. Firstly, a correspondence between the communication action's performative and the FML performative is done. Then, according to the performative and the FIPA-SL content in the communication action, a sentence in natural language is generated and set in the `speech` tag of the FML message. If the communication action is related to resources (objects in the environment), a `world` tag is added in the message to refer to those resources. GRETA has its own world representation so some information about the agent and the objects position are also sent to the GRETA platform. This information is sent solely when it is needed, for example, when the agent has to point to an object in the environment.

As explained in the previous section, the agent communicative intentions are generated by agent's behaviors (for example communication behavior and procedural behavior seen in the previous section or tutor behavior explained in the next section). These behaviors can be considered as our implementation of the Intent Planner module of the SAIBA framework.

4 ADAPTIVE TUTOR MODEL

The tutor model uses the knowledge of the domain model and the actions performed by the learner in order to choose the pedagogical actions that will be realized through the interface model. More precisely, the tutor behavior takes into account the actions done by the student (or his/her inaction) by recognizing them through the interface. The goal of our proposed tutor model is to adapt the execution of the pedagogical scenario to the student model represented in our work by the student's memory.

We have implemented the generic framework of memory proposed by Atkinson and Shiffrin [1] in the context of learning procedures (figure 4). In this implementation we created a link with MASCARET to formalize the user's memory content and a transformation flow from the incoming stimuli to the learner's memory. In our work, incoming stimuli from the virtual environment and the virtual tutor are restricted to those related to vision and hearing. Thus, the student can see 3D objects and hear instructions uttered by the tutor about the activities to realize. Therefore, we encode data about objects and activities. To formalize the encoding of information, we rely on MASCARET and the formalization of `CommunicationAction` explained in section 1.

In this work, we distinguish three structural components in human memory in which a sequence of cognitive processes is implemented to process information (encoding, storage, retrieval). The first operation involved in the information processing is the encoding of information. It is the transformation of incoming stimuli from the virtual environment and the virtual tutor to a formal representation that can be stored in the working memory. By using the model of communication action presented in section 3.1 and its execution through an ECA (section 3.2), it is possible to get the semantic content of the sentence uttered by the tutor and to instantiate this content in the learner's memory. The working memory stores and manipulates information based on the content of the sensory memory and the long-term memory (prior knowledge). The level of complexity of stored information in the working memory depends on the learner's prior knowledge (by complexity of

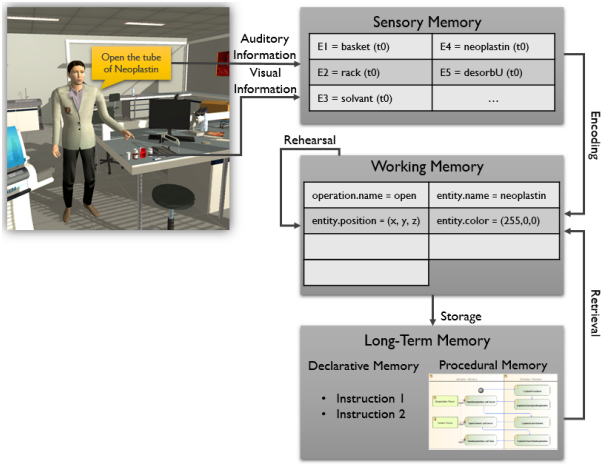

Figure 4: Formalization of the encoding and structuring of instructions in the learner's memory.

information we mean the level of the formal representation in MASCARET hierarchical formalism). This prior knowledge is retrieved from the long-term memory. The transfer of some knowledge from the working memory to the long-term memory, takes place when the learner completes an action.

This student model is used as an input in the tutor behavior which takes into account the actions done by the learner and the inferred student model to adapt the execution of the pedagogical scenario. This adaptation can be a modification of the student model (modification of the memory content) and/or the execution of a pedagogical action.

Our tutor behavior categorizes the actions done by the learner, based on two types of actions: (1) related to the domain model: an action can be either a domain action on a specific object or an answer to the tutor's questions (the tutor relies on the domain model to check if these actions are considered as errors or not), (2) related to the interaction: actions done by the learner can also be a feedback to the tutor's action (e.g. a facial expression, a question, observing the environment or an inaction). In this case, instead of using the domain knowledge, the tutor evaluates whether this feedback is negative or not. If the learner's action is considered as an error or as a negative feedback, this means that this action is unexpected in the context of the executed scenario. In this case a new pedagogical action is needed and the content of the learner's memory must be reevaluated. For example, if according to the pedagogical scenario the tutor explains the next action that the learner has to do, we instantiate two chunks in the working memory, one for the `Action` and the other one for the `Entity`. If the learner realizes an unexpected action (for example he/she shows a negative facial expression), then the tutor behavior considers that the learner does not know the object position, contrary to what the tutor had inferred. In this case the tutor remedies this situation by re-evaluating the content of the learner's working memory and then realizes a new pedagogical action to highlight the object.

5 CONCLUSION

In this paper we have integrated SAIBA compliant embodied conversational agents to MASCARET. The ECA integration is based on the formalization of communication actions between agents. These actions can be either triggered by an agent behavior or scheduled in a pedagogical scenario. They are executed on two levels: (1) agent level, where the content is automatically interpreted, (2) ECA level, where content is transmitted through human-like communication channels.

MASCARET has been used in several huge industrial projects (aerial activities management on aircraft carrier, maintenance of windmill turbines,...). Our model has been integrated in two industrial projects: learning procedures on a blood analysis automate and learning procedures on an aircraft radar. To validate the usability of our proposition, we also developed an adaptive tutor behavior that uses the communication behavior and actions, and executes a predefined pedagogical scenario through an ECA.

REFERENCES

[1] Richard C. Atkinson and Richard M. Shiffrin. 1968. Human memory: A proposed system and its control processes. *In K. W. Spence and J. T. Spence (Eds.), The Psychology of learning and motivation: Advances in research and theory (vol. 2).* (1968), 89–105.

[2] Pierre Chevaillier, Thanh-Hai Trinh, Mukesh Barange, Fréderic Devillers, Julien Soler, Pierre De Loor, and Ronan Querrec. 2012. Semantic modelling of virtual environments using MASCARET. In *Proceedings of the Fourth Workshop on Software Engineering and Architectures for Realtime Interactive Systems, IEEE VR, Singapore.*

[3] Arthur C. Graesser and Sidney D'Mello. 2012. Emotions during the learning of difficult material. *The psychology of learning and motivation* 57 (2012), 183–226.

[4] Arno Hartholt, David Traum, Stacy C. Marsella, Ari Shapiro, Giota Stratou, Anton Leuski, Louis-Philippe Morency, and Jonathan Gratch. 2013. All Together Now: Introducing the Virtual Human Toolkit. In *13th IVA*. Edinburgh, UK.

[5] Dirk Heylen, Stefan Kopp, Stacy C. Marsella, Catherine Pelachaud, and Hannes H. Vilhjálmsson. 2008. The next step towards a function markup language. In *Proceedings of 8th International Conference on Intelligent Virtual Agents (LNCS)*, Vol. 5208. Springer, 270–280.

[6] Hazaël Jones and Nicolas Sabouret. 2013. TARDIS - A simulation platform with an affective virtual recruiter for job interviews. In *IDGEI: Intelligent Digital Games for Empowerment and Inclusion.*

[7] Amol Kokane, Hitesh Singhal, Subhayan Mukherjee, and G. Ram Mohana Reddy. 2014. Effective E-learning using 3D Virtual Tutors and webRTC Based Multimedia Chat. In *International Conference on Recent Trends in Information Technology (ICRTIT).* 1–6.

[8] Robert Koper and Colin Tattersall. 2005. *Learning Design: A Handbook on Modelling and Delivering Networked Education and Training.* Springer International Publishing.

[9] Stefan Kopp, Brigitte Krenn, Stacy C. Marsella, Andrew N. Marshall, Catherine Pelachaud, Hannes Pirker, Kristinn R. Thórisson, and Hannes H. Vilhjálmsson. 2006. Towards a common framework for multimodal generation: The behavior markup language. In *Proceedings of 67th International Conference on Intelligent Virtual Agents (LNCS)*, Vol. 4133. Springer, 205–217.

[10] Fréderic Le Corre, Charlotte Hoareau, Franck Ganier, Cédric Buche, and Ronan Querrec. 2014. A Pedagogical Scenario Language for Virtual Environment for Learning based on UML Meta-model. Application to Blood Analysis Instrument.. In *CSEDU.* Spain, 301–308.

[11] James C. Lester, Sharolyn A. Converse, Susan E. Kahler, Todd Barlow, Brian A. Stone, and Ravinder S. Bhogal. 1997. The persona effect: affective impact of animated pedagogical agents. In *ACM SIGCHI Conference on Human factors in computing systems.* 359–366.

[12] Radoslaw Niewiadomski, Elisabetta Bevacqua, Maurizio Mancini, and Catherine Pelachaud. 2009. Greta: an interactive expressive ECA system. In *8th International Conference AAMAS.* 1399–1400.

[13] Roger Nkambou, Jacqueline Bourdeau, and Riichiro Mizoguchi. 2010. *Advances in Intelligent Tutoring Systems* (1st ed.). Vol. 308. Springer-Verlag Berlin Heidelberg.

[14] Hannes H. Vilhjálmsson, Nathan Cantelmo, Justine Cassell, Nicolas Ech Chafai, Michael Kipp, Stefan Kopp, Maurizio Mancini, Stacy C. Marsella, Andrew N. Marshall, Catherine Pelachaud, Zsòfia Ruttkay, Kristinn R. Thórisson, Herwin van Welbergen, and Rick J. van der Werf. 2007. The Behavior Markup Language: Recent Developments and Challenges. In *7th IVA (LNCS)*, Vol. 4722. Springer, Paris, France, 99–111.