Designing Training Virtual Environments Supported by Cognitive Agents

Jean-Paul Barthès, Gregory Moro Puppi Wanderley, Rerni Lacaze-Labadie, Domitile Lourdeaux

Designing Training Virtual Environments Supported by Cognitive Agents

Jean-Paul A. BARTHÈS, Gregory Moro Puppi WANDERLEY, Rémi LACAZE-LABADIE,
Domitile LOURDEAUX
Sorbonne Universités, Université de Technologie
de Compiègne
Heudiayc UMR CNRS 7253
CS 60 319, F-60203 Compiègne cedex
Email: {jean-paul.barthes,gregory.wanderley,rlacazel,domitile.lourdeaux}@hds.utc.fr

***Abstract*—This paper discusses the design of training environments using a virtual reality space populated by virtual characters participating in collaborative work. We are particularly interested in training a single human person who needs to organize and run a team doing collaborative work. Many problems have to be solved, due to the requirements: e.g. virtual characters must display a human behavior (possibly doing mistakes), be pro-active according to the situation and their physical, psychological and social profile; all exchanges of information must be done using natural language.**

***Index Terms*—Collaborative training platform, Cognitive agents**

As mentioned by Bosse et al. [1] "VR-based training offers a solution to many of the problems of real world training. In particular, VR-based training is less costly and time consuming, easier to set up, manipulate, and repeat, less dependent on place and time, and involves fewer risks and ethical difficulties." In Collaborative Virtual Environments for Training, virtual characters and the trainee have to work together to achieve team goals. Examples of such environments include the SecuReVi application for firefighters training [2] or the 3D Virtual Operating Room for medical staff training [3] or training personnel in industrial environments with high risks [4]. In such environments, agents must display a plausible human-like behavior. In many cases one must use natural language.

Although the technology exists for building such environments, we would agree with Bosse et al. when they write: "[...] the design of a training system as sketched above is a non-trivial task." In this paper we report and discuss some of the problems we encountered when developing a prototype of such an environment.

I. INTRODUCTION AND CONTEXT

The work reported in this paper has been inspired by the VICTEAMS project directed by D. Lourdeaux. The goal of the project is to train a medical leader to organize a medical post in a combat area so that all incoming wounded people can be taken care of efficiently prior to their evacuation. The training is not meant to improve medical behavior, but focuses on how the trainee should manage and supervise the medical team. The environment consists of a 3D virtual space representing the medical post in which a nurse and a paramedic must currently attend two wounded people (Fig.1). The scene must reflect what happens in real conditions. We will use this context as an example in the paper.

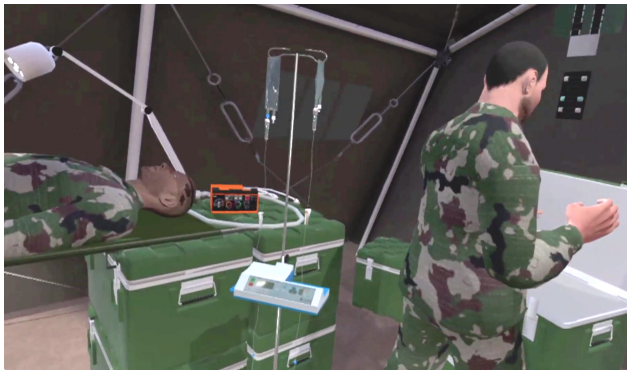

Fig. 1. Virtual reality view of the VICTEAMS medical post

II. REQUIREMENTS

We considered the following requirements: (i) the medical leader is immersed in the virtual 3D space (by wearing a VR Head-Mounted Display); (ii) the medical staff and the wounded people are represented by virtual characters; (ii) each medical virtual character has its own behavior, according to a physical and psychological profile, and has its own perception of the world; (iv) behaviors are not scripted but depend on what must be done and on what the medical leader orders; they must be human-like; (v) communications are done using natural language; (vi) communications may occur between the medical leader and the staff, or among the medical staff, or between the victims and the staff; and (vii) an external trainer, deus ex machina, can modify the situation dynamically by creating emergencies.

III. OVERALL APPROACH

It seems that the easiest way to develop such environments is to use a multi-agent system (MAS), due to its modularity and extensibility, coupling it with a virtual reality environment. In practice and to test the approach, we developed a prototype

that could apply to the VICTEAMS project. It will serve to illustrate the problems we encountered and the proposed solutions.

A. Global Architecture

It is quite straightforward to represent virtual characters by cognitive agents. Fig.2 represents a multi-agent architecture. Each virtual character (victim, medical staff) is controlled by an agent, the medical leader has a personal assistant agent to interface him with the system, a special agent represents the state of the world. In addition, one needs several technical agents to run the simulation, interface the virtual reality environment or interface external services. We will refer to the agents representing the medical staff and the victims as Non Player Agent or NPAs.

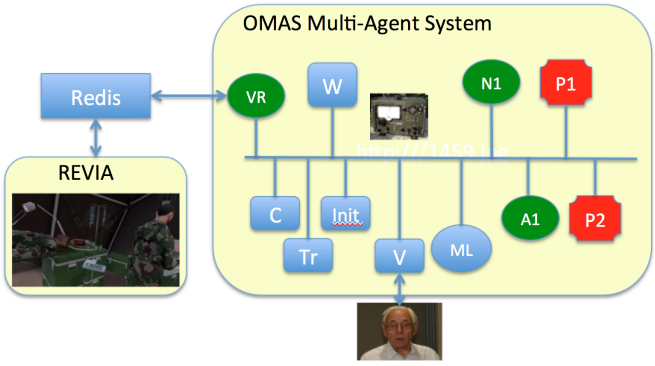

Fig. 2. A multi-agent architecture for the training system. A1, N1: medical staff; P1, P2: victimes; ML: Medical Leader; V: vocal interface; W: World; Init, Tr, C, VR: technical agents; REVIA: virtual reality interface.

An interest of such a platform is that new features can be added by adding more agents.

Now, although the use of the term NPA has a strong flavor of video game, the system has not the same characteristics. Indeed, in a video game the main problem for the system is to be efficient and able to handle situations as fast as possible, which is why most of the scenarios are scripted. In our case, the simulation is that of a medical post and the real time is relatively slow. The training scenarios cannot be scripted because the behavior of the NPAs is both reactive and pro-active, thus not predictable, with the consequence that the possible states are too numerous to be taken into account. In addition it is not possible to anticipate what the Medical Leader is going to do or to order. Finally, an external trainer can inject changes at any time in the state of the world, leading the NPAs to modify their behavior. Thus, a special behavior engine must be developed to handle NPAs.

B. Cognitive Agents

In this kind of simulations NPA agents are representing humans, thus their behavior must be plausible or as stated by Mittal et al. [5] they should be adaptive believable agents displaying a *strong emergence*. We thus decided that each NPA should have the following characteristics:

- it has a profile containing physical, psychological and social data (example in Table I)
- it has its own description of the world
- it has a representation of the perceived state of the other NPAs, victims and medical staff
- because it represents a professional, it has a set of procedures for actions corresponding to goals
- it has goals that decompose into tasks and actions
- it can do several actions at the same time
- it interacts using natural language

Physical data include things like energy or stress. Stress is important, because the level of stress can alter what an agent is doing, and lead to mistakes. Psychological data include a description of the agent personality. Social data govern the relations with other agents. Social data however, do not play a prominent role in the particular example of the medical post. Describing the profiles and the beliefs about the world requires developing an ontology, a knowledge base and rules to interpret the information they contain.

TABLE I
AGENT PROFILE (ATT: ATTRIBUTE; REL: RELATION)

Prop.	Name	Comment
att	name	name of the agent (an index)
rel	ability	capacity of doing a task
rel	action history	list of "Action Events"
att	action result	to keep the result of last action
att	behavior profile	lazy, active, hyperactive, dull,...
att	color	(default blue) used by the execution window
att	conclusions	set of conclusions to return to Medical Leader
att	energy level	current energy level
rel	focus	victim id on which the agent is currently focused
rel	goals	initial list of goals used by init
rel	goal history	list of completed goals
att	integrity	Is agent cooperative or rather selfish?
rel	liking	agent likes or dislikes other agents
rel	location	agent location
rel	objects	objects the agent has currently
att	observe scope	scope for observations
rel	patient	value is a list of victim taken care of
att	physical state	e.g. "has blood on hands"
att	qualification	"SC1" "SC2" or "SC3" last is doctor
att	skills	list of skills we have: diploma, certificates, position,...
att	stress	level of stress
rel	tools	list of tools that the agent has currently

Goals relate to actions that an NPA can do. For example "undress the victim, check for wounds, set a drift, inject morphine, report, etc." Such goals are composed of tasks like "go fetch a tourniquet, apply the tourniquet to the wounded limb, get a syringe, etc." Tasks are described in a graph, inspired by the ACTIVITY-DL model [6], and corresponding to what a professional of the same qualification as the NPA normally does to fulfill a goal (Fig.3). It must be noted that some of the tasks may require a joint effort from several virtual characters.

Thus, in order to determine the behavior of an NPA, we need a behavior engine that will organize the goals and traverse the corresponding task trees. We developed such an engine based on task priorities, and detailed elsewhere [7]. Each task

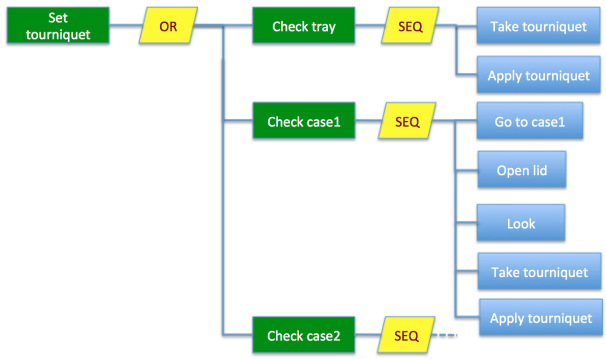

Fig. 3. An AND/OR graph for organizing tasks and actions. Here installing a tourniquet requires first to look whether there is one on the tray and if so apply it; if not, try to pick one from Case1 and apply it; if none there, go and pick one from Case 2 and apply it.

is assigned a priority that depends on the goal and the state of the agent. The task/action selected is then the one with the highest priority. Priority computation are done with each agent at each clock tick of the simulation. A task/action can be interrupted unless it is marked as non-interruptible. However, when coupled with a VR environment, the duration of some actions like moving to a specific location is imposed by the VR environment.

Discussion

In such an approach there is no planning since goals are represented by AND/OR graphs. The priority mechanism implements a way of traversing the graph, with possible interruptions. This approach seems adequate in our context since the medical staff is composed of trained professionals who do not plan from scratch each time a medical gesture must be done, but rely on their experience.

The key point in the behavior engine is the computation of the priority that must take into account the priority assigned to the current goal, the state of the agent, the state of the world, what the other agents (NPA and human) are doing and the social relations among agents. Details of how this is done is described in [7].

We also need rules to relate a given action to the physical or psychological parameters of the agent, as well as rules independent from actions, but depending solely on time, to modify the parameters of an agent. This is particularly important for the victims whose state can change rapidly.

In addition, some actions can occur in parallel, for example "reporting" can be done while setting up a tourniquet, or "reading the blood pressure" from the equipment, or "holding the head," or interacting with other agents. Some actions like "observing" must be triggered periodically to update the perceived state of the world, or specifically, for example to assess the content of a case after the lid has been pulled open.

Finally, mistakes can be introduced in the selection process by introducing rules related to the stress or the level of energy of the agent or by the trainer (see Section III-F).

Consequently, one needs a multi-agent platform where agents can do several actions at the same time, which they can do if they are multi-threaded. Agents must also be able to set up actions that are triggered periodically, and they must be able to react to changes in the world. Reactions can be taken into account during actions like "Observe" or indirectly through the computation of the priorities.

C. User Interface

The interface with the trainee, i.e. the Medical Leader, is easier if implemented vocally, using earphones and a microphone associated with the VR Head-Mounted Display (Fig.4). The trainee gives order or asks questions that are forwarded to the proper NPA. The answer comes through the earphones as a speech using the voice assigned to each NPA.

Fig. 4. Trainee (Medical Leader) wearing a VR Head-Mounted Display

Discussion

When developing the application, setting up the virtual environment takes some time and is not really necessary for testing the system. Because the Medical Leader is interfaced with a personal assistant, it is easy to display a master/PA interface window (Fig.5) showing the history of the different conversations and allowing to interact directly using typed text when the speech-to-text module has problems!

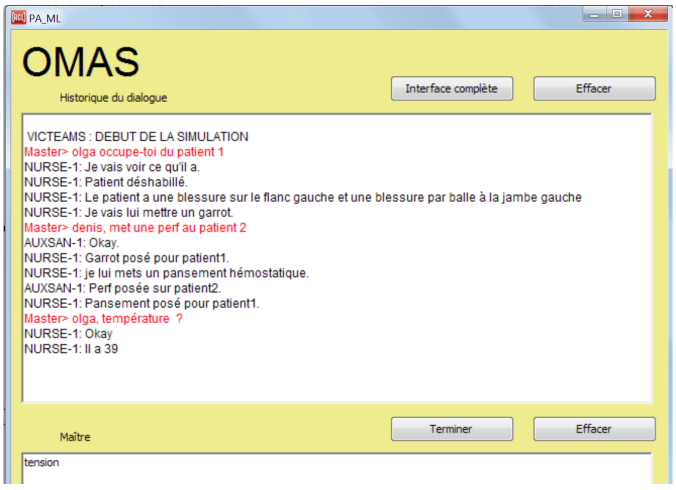

Fig. 5. Master/PA window for the Medical Leader showing interactions with Nurse-1 and Auxsan 1

D. Natural Language Interaction

Another interesting point in this type of simulation is the requirement to use natural language (Barange et al. [8]). This can be done by hooking a speech-to-text and text-to-speech module. However, one has still to handle the text resulting from the conversion.

There are two types of conversations: those between an NPA and the Medical Leader, and those between two NPAs (for example a nurse and a victim). The first type of conversation is asymmetric since on the side of the Medical Leader all utterances are sent to the NPA without any change, the only work the ML personal assistant agent has to do is to decide to what NPA it must send it, which is not so difficult. The NPA has then to analyze the text and decide whether to answer the question if it is a question, or to set up a goal if it is an order, eventually arguing about its usefulness.

Addressing several people at the same time requires running several conversations in parallel, creating new ones if new people ask questions (e.g. a victim). In this application we do not consider multiparty conversations, although some orders or questions may be broadcast.

Discussion

Because we are in a context of selecting actions, the agents are not requested to understand the text they receive but rather use it to select a possible goal from a library of goals[1]. Once a goal is selected, a specific conversation pertaining to this goal can be executed to obtain the necessary values of missing parameters. Then, the goal can be posted with an adequate priority, to be processed by the behavior engine.

A simple mechanism of conversation graph can be used as shown in (Barthès [10]) in which conversations are represented by state graphs and transitions between the different states, with possible recursion (sub-conversation) or interruption in case of a digression (new conversation). Fig.6 shows the very simple "undress patient" conversation graph. Since the input text might not contain enough information to determine which patient is targeted, a sub-conversation is needed to resolve this point (Fig.7).

E. Vocal Interaction

Vocal interaction between the trainee (Medical Leader) and an NPA is rather easy to implement using speech-to-text and text-to-speech modules, although recognition is sometimes difficult due to ambient noise. More difficult is conversation between two NPAs since they should also be heard by the trainee.

A problem with vocal interaction is the poor quality of the speech-to-text conversion. Two approaches are usually offered: (i) free speech; or (ii) grammar based speech. The two extremes are on one side letting the speaker say anything she wants, and on the other side test her utterances against a list of predefined possible phrases. The latter is more efficient but compels the designer to produce a list of expected utterances,

[1]Readers interested in conversations should read Winograd & Flores [9])

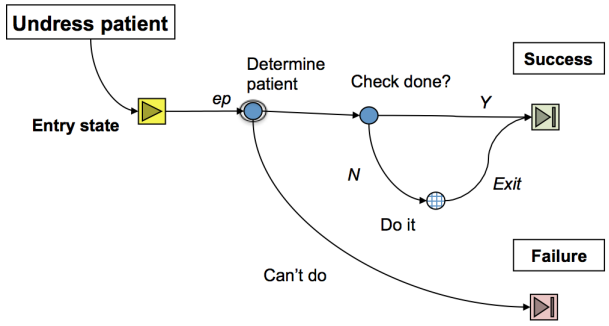

Fig. 6. Undress dialog. From the input one determines which patient to undress by calling a sub-conversation, then check if the patient is already undressed; if so exit with a success; otherwise set up a goal to undress him; then exit with success.

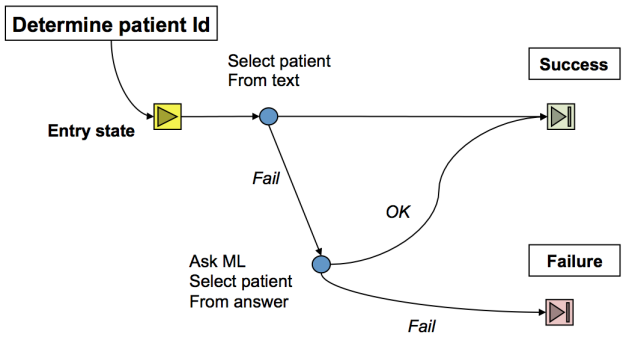

Fig. 7. Determine patient dialog. Try first to get the patient reference from the input text; if it fails, ask a question to the Medical Leader and analyze the answer.

which is both tedious and difficult. Indeed, the list can be quite long because the number of ways one can express the same content can be quite high.

Discussion

Although some problems may arise from poor recognition from the speech-to-text module, it does not constitute a critical point since it may introduce mistakes that the trainee is expected to correct.

A more difficult point comes from cases when several agents speak at the same time, which, as can be checked when listening to a radio program, leads to garbled results difficult to understand.

The possibility of an NPA misunderstanding some order and starting doing some irrelevant action requires implementing the possibility for the trainee to stop this action. This must be taken into account and adds some complexity to the behavior engine.

F. Trainer

The goal of the trainer is to dynamically create emergency situations, either by changing the difficulty of the ongoing situation or by re-orienting the situation toward new (training) objectives. The agent in charge of planning and executing

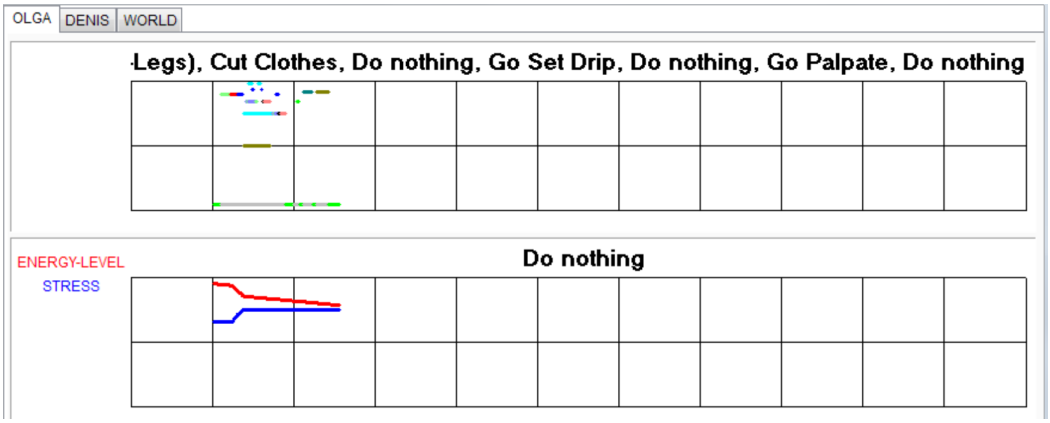

Fig. 8. Trace of the actions undertaken by the nurse on the top diagram showing the internal computed priority of the actions, and of the evolution of stress and energy levels of the nurse in the second diagram.

events starts from a given state of the world and uses a list of goals selected by the trainer. An event is generated in the form of a sequence of actions that can be applied in that state. These actions are system-controlled actions that we oppose to actions executed by the trainer or by NPAs. For example the event "victim dies" can be executed as the following sequence: "a wound of the victim gets infected," "the victim's health condition gets bad" and "the victim stop breathing."

Discussion

Selected training goals may not always be satisfied in the current state, for example if the trainer calls for the death of a victim but no victim has any serious wound or bad health condition at that moment. In this case, the planner tries to predict the future ongoing situations by including expected actions in its action-space. An expected action is a trainer or an NPA action that the Trainer agent cannot trigger but only expects to happen. When an expected action happens in the virtual environment, it is marked as done in the plan. When all the predecessors of a system-controlled action are marked as done, the action is triggered. For example, let us imagine that the trainer selects the goal "medical material breaks" to raise the difficulty of the situation, but not medical material can break in the current state. In that case, the planning system detects in the state that a victim has a wounded limb, it plans the expected action "apply the tourniquet to the wounded limb" and finally plans the event "the tourniquet breaks." Therefore, the goal of the trainer can be achieved in the environment as soon as the tourniquet is applied.

G. Virtual Reality Interface

Interfacing the virtual reality system (VRS) is usually not difficult when its API accepts messages with a high semantic content. However, several features have to be taken into account. If the VRS is autonomous, then it has its own clock. The VRS returns information about the end of an action (e.g. moving a virtual character to some place), about possible errors (an action cannot be done). It should also be possible to suspend then resume it.

Simulating the VRS when developing the application is not so trivial, but the possibility of customizing an agent to do so simplifies the approach.

H. Debugging

Designing the type of environment we discuss is difficult but debugging them is worse. We are in a complex environment with many interactions, asynchronous events, parallel conversations, which makes it impossible to replicate errors when they occur during execution. Typically a session will run 20 to 30 different asynchronous threads at any time. One must then devise some ways of displaying what happens during execution of the simulation.

Tracking the conversation with the Medical Leader has been mentioned in Section III-C and shown in Fig.5 using the default window for personal assistant agents. However, conversation states must be followed when proposing new conversations. This can be done by using a specific window devoted to conversations. Another interesting point consists in posting the sequence of actions that an NPA undertakes together with the variation of some of its parameters. Fig.8 displays the actions taken by the nurse plotted as a function of their computed priority and in the graph underneath the variation of her energy level and stress level in a specific tag. The second tag refers to the second medical NPA, here called Denis, and the third tag displays a page showing the messages sent to the virtual reality environment.

It is also interesting to trace the actions when they are done as shown in Fig.9.

Of course it is also necessary to be able to display and edit the ontology for each agent, to be able to suspend the simulation and later resume it, to accelerate or slow the clock or to step through each clock tick. However, in the last cases one must simulate the virtual reality environment because it runs on its own clock.

If anything gets very bad, one can use the tracing tools of the agent platform like monitoring the messages exchanged by agents.

Fig. 9. Trace of the nurse actions on the left column while remaining pending actions are shown in the right column of the page.

Discussion

All the debugging tools mentioned in this section are extremely useful for developing the simulation system. Selective traces of a combination of agents is quite efficient. However, the number of parallel threads is indeed a problem. The main consequence is that the designer has a problem getting a full picture of the application at a given time, which would call for additional more efficient debugging tools. Such tools must also be provided for maintenance when the application is installed.

IV. CONCLUSION AND FUTURE WORK

Designing and developing an agent-based prototype coupled to virtual reality for training purposes taking into account agent profiles, pro-active behaviors, asynchronous external interventions, simply proved that we currently have the technology for doing so. But as Roger Schank once remarked: "*Artificial intelligence is 5% fun and 95% hard work.*"

Given that the prototype is a VR-training system and not a cognitive platform for running social simulation experiments as proposed by Smart and Sycara [11], we reached two main conclusions from this work. First, an MAS architecture provides a good basis for building such complex systems thanks to its modularity and nearly unlimited possibilities of extensions. It must however offer cognitive agents with their ontology, their behavior rules, their profiles, be flexible enough to allow parallel execution, and integrate natural language interfaces with vocal interaction. Second, the main effort for developing an application concerns the debugging phase. Debugging tools, as well as various handles through the code, must be built from the beginning and should be part of the early design. Debugging tools offered by multi-agent platforms are usually not sufficient. Without some good debugging tools, building such applications is extremely difficult.

Considering now the prototype we have built, it appears that it could be improved by adding actions, more rules, developing conversations. In other words we think that this architecture can scale up easily. In addition, the multi-agent approach makes it possible to add more features without having to rewrite everything all the time. However, the prototype has not yet been tested by actual medical people and when it is, the testing could uncover hidden weaknesses. We are eager to start doing this.

In the near future we are also considering improvements like installing a better analysis of natural language interactions (to allow for example complex sentences like "Prepare to do this as soon as this other thing has been checked"). Also, currently we do not take into account the exact location of agents for distributing messages. It can be done and might be interesting to consider it. Finally some real situations require multi-lingual interaction in the context of international cooperating teams.

ACKNOWLEDGEMENTS

We would like to acknowledge the help of all members of the VICTEAMS project (ANR-14-CE24-0027) funded by the French National Research Agency (ANR), the French Defense Procurement Agency (DGA) and labeled by the Labex MS2T.

REFERENCES

[1] T. Bosse, J. de Man, and C. Gerritsen, "Agent-based simulation as a tool for the design of a virtual training environment," in *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), Warsaw, Poland, August 11-14, 2014 - Volume III*, 2014, pp. 40–47.

[2] R. Querrec, C. Buche, E. Maffre, and P. Chevaillier, "Scurvi: virtual environments for fire-fighting training, 5th virtual reality international conference," in *2003 5th virtual reality international conference (VRIC03)*, 2003, pp. 169–175.

[3] M. Sanselone, S. Sanchez, C. Sanza, D. Panzoli, and Y. Duthen, "Constrained control of non-playing characters using monte carlo tree search," in *2014 IEEE Conference on Computational Intelligence and Games, CIG 2014, Dortmund, Germany, August 26-29, 2014*, 2014, pp. 1–8. [Online]. Available: https://doi.org/10.1109/CIG.2014.6932886

[4] L. Edward, D. Lourdeaux, and J. A. Barthès, "Virtual autonomous agents in an informed environment for risk prevention," in *Intelligent Virtual Agents, 9th International Conference, IVA 2009, Amsterdam, The Netherlands, September 14-16, 2009, Proceedings*, 2009, pp. 496–497.

[5] S. Mittal, M. J. Doyle, and E. Watz, "Detecting intelligent agent behavior with environment abstraction in complex air combat systems," in *2013 IEEE International Systems Conference (SysCon)*, 2013, pp. 662–670.

[6] C. Barot, D. Lourdeaux, J. Burkhardt, K. Amokrane, and D. Lenne, "V3S: A virtual environment for risk-management training based on human-activity models," *Presence*, vol. 22, no. 1, pp. 1–19, 2013.

[7] J. A. Barthès, L. Callebert, and D. Lourdeaux, "Priority-based contextual local decision making in multi-agent systems," in *20th IEEE International Conference on Computer Supported Cooperative Work in Design, CSCWD 2016, Nanchang, China, May 4-6, 2016*, 2016, pp. 186–191. [Online]. Available: https://doi.org/10.1109/CSCWD.2016.7565986

[8] M. Barange, A. Kabil, and P. Chevaillier, "The C2BDI agent architecture for teamwork coordination using spoken dialogues between virtual agents and users," in *Advances in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection - 12th International Conference, PAAMS 2014, Salamanca, Spain, June 4-6, 2014. Proceedings*, 2014, pp. 315–318.

[9] T. Winograd and F. Flores, Eds., *Understanding Computers and Cognition*. Norwood, NJ, USA: Ablex Publishing Corp., 1985.

[10] J.-P. A. Barthès, "Improving human-agent communication using linguistic and ontological cues," *Intl. J. Electronic Business*, vol. 10, no. 3, pp. 207–423, 2013.

[11] P. R. Smart and K. Sycara, "Cognitive social simulation and collective sensemaking: An approach using the act-r cognitive architecture," May 2014. [Online]. Available: https://eprints.soton.ac.uk/364297/