Interfacing Users and CSCW Applications using a Web-Based Embodied Conversational Assistant

Emerson Cabrera Paraiso, Yuri Campbell

Pontifícia Universidade Católica do Paraná, Post-Graduate Program on Informatics
CEP: 80.215-901, Curitiba, Paraná, Brazil
{emerson.paraiso, yuri.campbell}@pucpr.br

Abstract: WebAnima is a web-based embodied conversational personal assistant agent. It is an interface agent specially designed to assist team members of a CSCW application during their daily work based on computers. In WebAnima, the intelligent behavior is guaranteed thanks to a conversational interface and ontologies that support semantic interpretation. We believe that embodied conversational assistants will improve the quality of assistance and increase collaboration between project members. We describe the design of the agent, highlighting the role of ontologies for semantic interpretation and the dynamic behavior of the embodied animated agent.

1 Introduction

The use of multi-agent systems (MAS) for improving cooperative work based on computers (CSCW) has become very popular these past few years ([1], [2] and [3]). In general, for this kind of application, the MAS architecture contains two types of agents: the ones responsible for executing a specific task and the ones aimed to interface users with the system. The latest are commonly named interface agents [4]. In this sense, an interface agent is a semi-intelligent system which assists users with daily computer-based tasks [5]. We agree that, for the effective use and success of this approach, the interface agent must have a user interface specially designed for CSCW applications. The interface agent and its interface must take into account that users are doing many tasks and using several different applications at the same time (browsers, word processors, CADs, etc.). The user interface should captivate users to keep using their agent. To achieve this goal, we have been developing a new personal interface agent called WebAnima. WebAnima involves the use of conversational animated personal assistants (CAPA) coupled with the MAS. A CAPA is the result of mixing personal assistants and embodied conversational agents. Embodied conversational agents are animated anthropomorphic interface agents that are able to engage a user in real-time, multimodal dialogue, using speech, gesture, gaze, posture, intonation, and other verbal and nonverbal behaviors to emulate the experience of human face-to-face interaction [6]. They are designed to converse like a human as much as their intelligence allows [7]. In WebAnima, the intelligent behavior is guaranteed thanks to a conversational interface [8] and ontologies that support semantic interpretation.

Each team member (community of users connected to the CSCW application) has a WebAnima agent that behaves according to its user profile built on the fly. WebAnima can potentially improve the exchange of information among the participants, provide support, improve workflows and procedure controls, and provide convenient user interfaces in MAS-based CSCW applications [9].

A WebAnima agent can be used in different domains, since its knowledge about the domain and tasks to be performed are represented as ontologies. As the result of this approach we expect:

- to improve the quality of assistance;
- to improve collaboration between members;
- to improve users interest on using the system; and
- to reduce the user's cognitive load.

In this paper, we present the WebAnima architecture and how it can centralize and control user interaction in an MAS application. In order to contextualize, examples are based on an MAS that supports a research and development team on its daily activities. The paper begins by describing the WebAnima architecture (section 2). After that, we present the conversational interface controller in section 3. In the section 4, we present the embodied animated interface. Finally, we offer a conclusion and indicate some perspectives for forthcoming work.

2 The WebAnima Agent Architecture

Before describing what a WebAnima agent is, it is important to define a personal assistant (PA). A PA is an interface agent in charge of interfacing humans to the system. The particular skills of a PA are devoted to understanding its master and presenting the information intelligently and in a timely manner. We have applied PAs in CSCW applications, where they play a major role [10]. Firstly, they are in charge of all exchanges of information among team members. Secondly, a PA is able to organize the documentation of its master with the help of a service agent. Finally, PAs must capture and represent the team members' operations, helping them in the process of preserving and creating knowledge. Thus, our main goal is to provide a system, that supports collaborative work and helps to capture and to organize experiences without overloading the team members with extra-work.

WebAnima is a web-based embodied conversational personal assistant agent, as shown in Fig 1. WebAnima is, in fact, an evolution of SpeechPa: an intelligent speech interface for PAs in research and development projects [10].

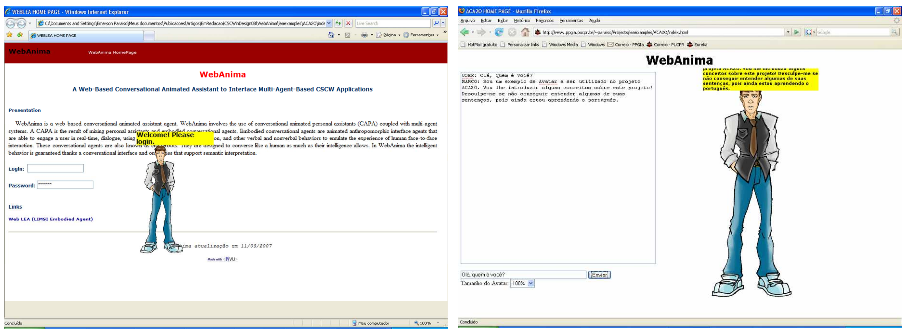
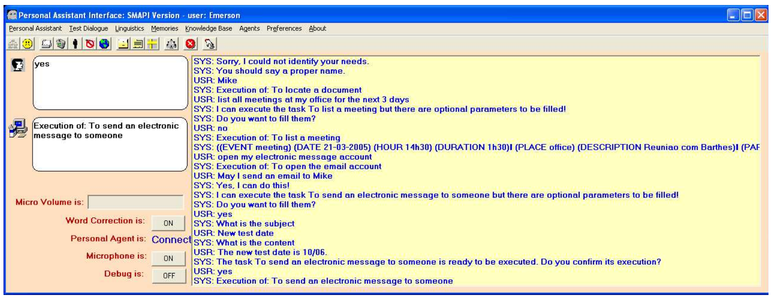

WebAnima
A Web-Based Conversational Animated Assistant to Interface Multi-Agent-Based CSCW Applications
Welcome! Please login.
Login:
Password:
WebAnima
Enviar

Fig 1. Some WebAnima snapshots.

As shown in Fig 2, SpeechPA handles dialogs in natural language and was used to interface team members in a R&D project prototype. Even if SpeechPA is a mixed-initiative conversational interface, its “static” behavior limits its acceptance. Due to some assumptions defined at the beginning of the project, SpeechPA follows the strategy of treating only directive speech acts [11], reducing the number of turn-takings since some speech acts, like acknowledgement acts (“Thank you” or “Have a nice trip”) are not used by the PA. To overcome these limitations, in the WebAnima agent, the conversational module accepts and uses a wider set of speech acts, giving more flexibility to the agent.



yes

Execution of: To send an electronic message to someone

Micro Volume is:
Word Correction is: ON
Personal Agent is: Connect
Microphone is: ON
Debug is: OFF

```
SYS: Sorry, I could not identify your needs.
SYS: You should say a proper name.
USR: Mike
SYS: Execution of: To locate a document
USR: list all meetings at my office for the next 3 days
SYS: I can execute the task To list a meeting but there are optional parameters to be filled!
SYS: Do you want to fill them?
USR: no
SYS: Execution of: To list a meeting
SYS: ((EVENT meeting) (DATE 21-03-2005) (HOUR 14h30) (DURATION 1h30)I (PLACE office) (DESCRIPTION Reuniao com Barthes)I (PAF
USR: open my electronic message account
SYS: Execution of: To open the email account
USR: May I send an email to Mike
SYS: Yes, I can do this!
SYS: I can execute the task To send an electronic message to someone but there are optional parameters to be filled!
SYS: Do you want to fill them?
USR: yes
SYS: What is the subject
USR: New test date
SYS: What is the content
USR: The new test date is 10/06.
SYS: The task To send an electronic message to someone is ready to be executed. Do you confirm its execution?
USR: yes
SYS: Execution of: To send an electronic message to someone
```

Fig 2. The original SpeechPA interface.

In order to animate the interface and to test the acceptability of animated characters by users, we added to SpeechPA an avatar (human-like figure) (as shown in Fig 1).

The WebAnima agent structure is shown in Fig 3. The design and implementation of such agent is a hard task and involves many different components: dialogue controllers, natural language parsers, speech recognizers and synthesizers, knowledge manipulators, to list a few. For the design of WebAnima we made some assumptions related to the agent and its operation, described in detail in [9].

A WebAnima agent is a rather complex system. Among the many types of agent models and systems that have been proposed, we selected cognitive agents. The main advantage of cognitive agents is the possibility of designing intelligent behaviors by specifying a set of skills. In addition, in our case, such agents run independently of any particular task to solve.

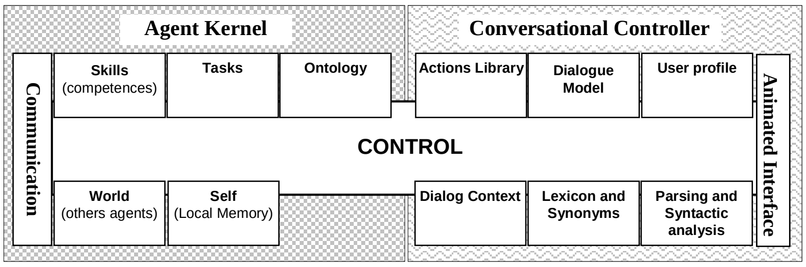


Fig 3. The WebAnima agent structure.

Our agent is built around three main blocks: the user interface (a web-based animated interface implemented using the toolkit WebLEA [12] – see section 4 for details), the ontology-based conversational interface controller, mainly responsible for controlling the dialogue (the same used in SpeechPA), and a fixed body, called the Agent Kernel. The Agent Kernel block contains all the basic structure that allows an agent to exist. Further information on the Agent Kernel can be founded in [10]. The next two sections describe in detail the other two main WebAnima blocks: the conversational interface controller and the animated user interface.

3 The Conversational Interface Controller

To produce a more attractive interface agent, from the user interface point of view, WebAnima incorporates a conversational interface. Conversational interfaces as defined by Kölzer [8], let users state what they want in their own terms, just as they would do, speaking to another person.

Whenever the user says something, this is known as an utterance. It can be a single word, or contain several words (a phrase or a sentence). For example, “email,” “email account,” or “I’d like to open my email account” are utterances. The utterances are captured using a commercial automatic speech recognition engine that returns the recognized result for each word. The Utterance Capturing module concatenates all the words forming an utterance.

Like in most dialogue systems, we process each utterance sequentially. The process of interpreting an utterance is done in two steps: (i) parsing and syntactic analysis; and (ii) ontology application. The results are sent to the dialogue manager continuously, or back to the user when they do not make sense.

The parsing algorithm replaces each utterance stem with its syntactic category (verb, noun, adverb, etc) with the help of a lexicon file and a set of grammar rules. In our application, a typical utterance could be: “I need a list of all project participants.” According to our taxonomy this is an order utterance and can be processed by the grammar rules. If a sentence is not well formed, according to the grammatical structure, or if it is out of the domain, then it is classified as a nonsensical utterance. In this case the user is invited to reformulate her sentence.

The mixed-initiative and task-oriented dialogue mechanism is coordinated by the dialogue manager. It is capable of choosing a dialogue model appropriate to the beginning of a session. Each dialogue session is conducted as a task with sub-tasks. When the user requests an action, the dialogue manager tries to execute it, creating a task that is dispatched by the Action Looping module. However, if the initial utterance lacks crucial information—e.g., an action parameter—it starts sub-tasks to complete the action list, asking additional information from the user. The Action Looping handles GUI events and also receives calls from the dialogue manager. It is also responsible for merging all modalities (e.g., button click and speech).

In the context of an open conversation, the problem of understanding is complex, demanding a well structured knowledge base. Domain knowledge is used here to further process the user's statements and for reasoning. To this effect, we are using a set of task and domain ontologies. The main purpose of an ontology is to enable knowledge sharing and reuse. The key components that make up an ontology are a vocabulary of basic terms and a precise specification of what those terms mean [13].

In this ontology-based conversational interface, we are using a set of task and domain ontologies, separating domain and task models for reasoning. As suggested by Allen [14], this is interesting for domains where task reasoning is crucial. Besides, using domain knowledge separately reduces the complexity of the linguistic modules, and allows a better understanding of statements. In one of their works, Milward and Beveride [15] describe how scripted dialogue systems are moving to a new generation of practical systems based on domain knowledge and task descriptions.

Ontologies play two main roles in our PA: (i) they help an agent to interpret the context of messages sent by others agents or by the user (utterances); and (ii) they keep a computational representation of knowledge useful at inference time. The design of such ontologies must cover the user's world, in terms of entities and their relationships. In addition, the ontologies must also facilitate the process of semantic interpretation, supplying the parser with linguistics elements, like noun synonyms, or hyponyms/hyperonyms.

Given this overview about the ontologies and their role in the process, let's focus our attention on the semantic interpretation mechanism. The approach to the semantic interpretation presented here is based on the notion that the meaning of utterances can be inferred by looking for concepts and their attributes. Precisely, the module responsible for applying the ontology to the utterance is interested in finding the list of verbs that indicate the task to be executed and the domain concepts. The corresponding keywords are concepts of the ontology directly related to a list of actions.

In this paper, the ontologies are simple and short enough to understand the semantic interpretation mechanism. The concepts and their properties are organized to map the world but also to help processing natural language (by adding a list of applicable actions to each concept of the ontology). To illustrate how the mechanism works, consider the utterance:

USER: Could you list all articles about Agents?

A very simple piece of ontology is shown in Fig 4a (we used Protégé [16] for a simplified representation), describing concepts that model a project. A project, according to the ontology, may have different types of documents, an address book, an agenda, and a list of members. A set of actions are related to each concept. Each concept may have some attributes (Fig 4b). Note that a set of actions (e.g., read, list, erase, shown in Fig 4c) may be applied to each concept, as shown Fig 4d.

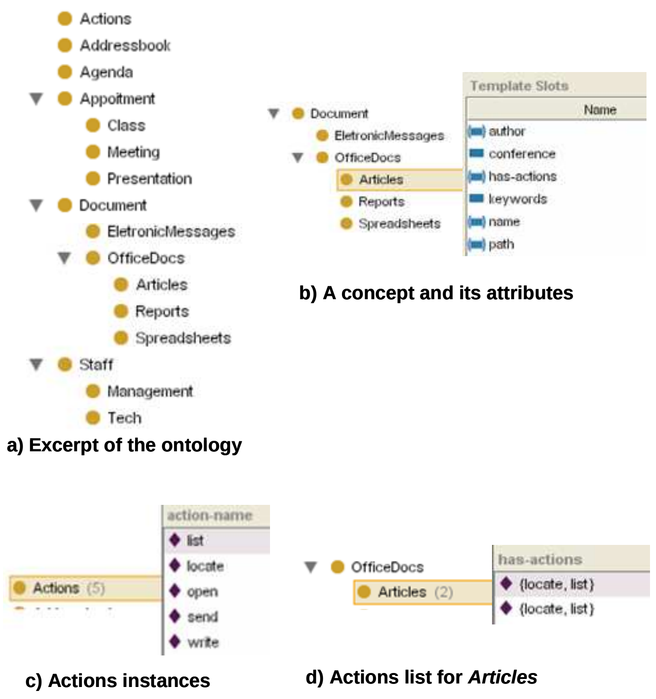
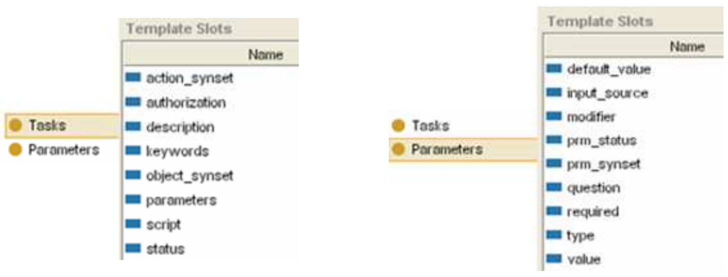


Fig 4. An excerpt of the ontology *Project*.

To interpret the given input, the parser checks its context. It verifies that it is a question related to the domain. To do so, it uses the domain ontology and the lexicon. Since it is a question and since it is related to the application domain, the Grammar Verification module returns a matrix containing the list of tokens and their syntactic classification. By looking up the tokens in the ontology, it finds that the token *list* is an action (Fig 4d). Note that it uses a list of synonyms (e.g. “list” and “enumerate” are synonyms in this context). It finds also that *articles* is an object and *Agents* is one of its properties (to define each property of a concept, one should give its type, a list of synonyms, its cardinality and a domain restriction). Next, the dialogue manager takes control of the dialogue.

Tasks in our system are represented as shown in Fig 5. A task has a set of parameters that are filled during a dialogue session. The dialogue manager will push a task onto the stack of tasks when an utterance related to the task is given. Many tasks may be handled simultaneously (even tasks of the same type), for instance:

USER: I need to send an email to Mike Palmer.



a) Task parameters **b) Parameters' fields**

Fig 5. Task model example.

After the parsing and semantic analysis, the dialogue manager is able to start a new task, since it is related to the domain (according to our first ontology presented in Fig 4). The task *To Send an Electronic Message* has some parameters to be filled before the agent is able to execute it (each task has the structure shown in Fig 5a). One of the parameters may be the subject of the message. Since the given utterance does not contain this information, the dialogue manager will request it from the user, asking her the question defined in the appropriate question field (as listed Fig 5b). The dialogue manager changes the task status to pending and waits for a response from the user. When all fields are filled, the dialogue manager sends the task for execution.

Our platform runs in a Microsoft Windows™ environment, using the default automatic speech recognition and text to speech engines. Ontologies are XML files.

4. The Embodied Animated Interface

The third main block that composes WebAnima is its embodied animated interface. In WebAnima we used a toolkit called WebLEA. WebLEA is a technology of 2D cartoon-like simple graphic characters that can be displayed and animated on web pages. WebLEA is a toolkit dedicated to the display and animation of embodied characters on web pages using the JavaScript technology in full client mode. The 2D cartoon-like characters (Fig 6) which are used for embodying the assistant enable to display various postures, facial expressions and gestures [17].

In WebAnima, the character behavior is driven by two set of parameters: a static set and a dynamic set. The static set of parameters determines general behaviors, such as: the use of politeness and/or humor when formulating responses, general movements (allowing or not the character to walk on the screen), use of gestures, size, speech language (English or Portuguese), etc. These parameters are set by each user. They can be changed at any time.

Fig 6. The Three different WebLEA characters.

The other set of parameters are related to specific behaviors and may impact in task execution or limit autonomy. These parameters are defined dynamically. As much as the agent is used, more accurate this set is. They are initialized when a new user is created (e.g. a new engineer starts working in the project) with default values. As mentioned at the beginning of this article, each team component has his/her own PA. Three main behavior parameters compound this set: degree of autonomy, presentation policy and help policy.

As a cognitive agent, the PA may assume some responsibilities regarding the execution of tasks. The degree of autonomy depends on the pattern defined by the user. Every time a task is selected to be executed, the PA will verify if it should demand (to its user) authorization to fire it. When writing a task (in the task ontology) the designer of the application must define if the PA must or not request authorization before execute it (parameter "authorization" as shown in Figure 6a). For each task that requires authorization, at the first time the PA will ask the user if it may assume the responsibility to execute the task without authorization the next time. Tasks like: to erase emails classified as spam or to charge a new spreadsheet just uploaded in the spreadsheet database are good examples. The PA will use the same strategy to determine when interrupt the user to present a message or to request an information needed to accomplish a task.

A PA that constantly interrupts its user with boring questions or messages may drive the user to ignore it. To avoid that, a presentation policy must be defined. The presentation policy defines how information is presented and how the user is warned. Since the PA is the only interface the user has with the system, the interaction may be stimulated by the PA or by a request made by a service agent. For instance: a service agent may inform that a printing is finished or that an email just came in. The PA will classify each message in two categories: the ones that may be stored and displayed later and the ones that should be displayed immediately. Every message or query generated by the PA itself will be displayed immediately. Queries from service agents will be displayed as soon as possible (when the user is not busy giving information for executing a specific task). Warning messages or the confirmation of a task execution will be postponed and printed in a log window.

The help policy is related to how often the PA will suggest help when the user uses the system. For new users, the agent will propose to guide them in order to accomplish a specific task (for instance: send and compose an email or search a document in the document's database). To do that, the PA will consult a base of procedures, fed by the community of PAs. Each time a user accomplishes a task, her PA will register the steps needed to accomplish it (if not registered) in this centralized database. For a new user, the PA will suggest to guide her until she accomplishes the specific task for the first time. After that, the PA will assume that the user may herself accomplish the task.

5. Conclusions and Future Work

In this paper we presented a web-based embodied conversational assistant to interface users with CSCW applications. We described the design of the agent, highlighting the role of ontologies for semantic interpretation and the dynamic behavior of the embodied animated agent.

Since the application is a PA, an essential feature of the user interface was respected: predictability. It was an assumption stated in the beginning: to provide correct responses and act according to the user's command. Impossible requests, such as those out of context, are easily handled since the system uses a competence list described as an ontology.

WebAnima illustrates the enormous potential for task-oriented collaboration between team members and conversational agents in CSCW applications. As advocated by Rickel and Johnson [18], although verbal exchanges may be sufficient for some tasks, we expect that, with WebAnima, many domains will benefit from an agent that can additionally use gestures, facial expressions and locomotion.

The actual version of WebAnima does not support multiples languages. This is a special challenge in natural language based interfaces (special grammars, etc.). By now, we work with English and Portuguese only.

The next step is to improve the agent behaviour by adding a learning module to it in order to keep a more sophisticated user profile. This will allow clustering users and better adapting the PA behavior. This is the first step to treat special multi-cultural situations. We are also studying others alternatives to implement the embodied agents, such as Java 3D. We hope to implement a faster solution since WebLEA is too slow in some platforms.

Acknowledgments
Yuri Campbell would like to thanks Fundação Araucária–Brazil that partially supported him in this research.

References

1. Spinosa, L. M., Quandt, C. O., Ramos, M. P.: Toward a Knowledge-Based Framework to Foster Innovation in Networked Organizations. In: 7th CSCWD 2002, Rio de Janeiro, Brazil, (2002).
2. Tacla, C. A., Barthès, J.-P. A.: From Desktop Operations to Lessons Learned. In: The 7th CSCWD, Rio de Janeiro, (2002).
3. Wu, S., Ghenniwa, H., Shen, W.: User Model of a Personal Assistant in Collaborative Design Environments. In: Agents in Design, MIT, pp. 39-54, Cambridge, USA. (2002).
4. Jennings, N., Sycara, K., Wooldridge, M.: A Roadmap of Agent Research and Development. In: Autonomous Agents and Multi-Agent Systems 1, pp. 7-18. (1998).
5. Maes, P., Kozierok, R.: Learning Interface Agents. In: The Eleventh National Conference on Artificial Intelligence, pp. 459-465. Washington D.C.: AAAI Press.
6. Bickmore, T., Cassell, J.: Social Dialogue with Embodied Conversational Agents. In: J. van Kuppevelt, L. Dybkjaer, and N. Bernsen (eds.), Natural, Intelligent and Effective Interaction with Multimodal Dialogue Systems. New York: Kluwer Academic. (2004).
7. Sing, G., Wong, K., Fung, C., Depickere, A.: Towards a More Natural and Intelligent Interface with Embodied Conversation Agent. In: International Conference on Game Research and Development, pp. 177-183. (2006).
8. Kölzer, A.: Universal Dialogue Specification for Conversational Systems. In: IJCAI – Knowledge and Reasoning in Practical Dialogue Systems (1999).
9. Paraiso, E. C., Barthes, J.-P. A.: An Intelligent Speech Interface for Personal Systems in R&D Projects. In: Expert Systems with Applications, v. 31, pp. 673-683. (2006).
10. Paraiso, E. C., Barthes, J.-P. A.: An Intelligent Speech Interface for Personal Assistants in R&D Projects. In: The 9th IEEE International Conference on CSCW in Design, pp. 804-809, Coventry – UK. (2005).
11. Searle, J. R.: A Taxonomy of Illocutionary Acts. In: Language, Mind and Knowledge, Vol. 7, University of Minnesota Press, pp. 344-369. (1975).
12. Sansonnet, J-P., Martin, J-C., Leguern, K.: A Software Engineering Approach Combining Rational and Conversational Agents for the Design of Assistance Applications. In: T. Panayiotopoulos et al. (Eds.): pp. 111 – 119, IVA 2005, LNAI 3661. (2005).
13. Guarino, N.: Formal Ontology in Information Systems. In: FOIS'98, Italy, IOS Press, pp. 3-15. (1998).
14. Allen, J., Ferguson, G., Stent, A.: An Architecture for More Realistic Conversational Systems. In: Intelligent User Interfaces 2001, Santa Fe, NM. (2001).
15. Milward, D., Beveride, M.: Ontology-Based Dialogue Systems. In: International Joint Conference on Artificial Intelligence IJCAI – 03, Acapulco, Mexico. (2003).
16. Gennari, J., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubézy, Eiksson, M., Noy, N., Tu, W.: The Evolution of Protégé: An Environment for Knowledge-Based Systems Development. (2002).
17. Abrilian, S., Buisine, S., Rendu, C., Martin, J.-C.: Specifying Cooperation between Modalities in Lifelike Animated Agents. In International Workshop on Lifelike Animated Agents: Tools, Functions, and Applications, pp. 3-8, Tokyo, Japan. (2002).
18. Rickel, J., Johnson, W.L.: Task-Oriented Collaboration with Embodied Agents in Virtual Worlds, J. Cassell, J. Sullivan, and S. Prevost (Eds.), Embodied Conversational Agents. Boston: MIT Press, pp. 95-122, (2000).