WebAnima: A Web-Based Embodied Conversational Assistant to Interface Users with Multi-Agent-Based CSCW Applications

Emerson Cabrera Paraiso[1] Yuri Campbell[1] Cesar A. Tacla[2]
[1] *Pontifical Catholic University of Paraná, Post-Graduate Program on Informatics*
CEP: 80.215-901, Curitiba, Paraná, Brazil
paraiso@ppgia.pucpr.br, yuri.campbell@pucpr.br
[2] *Universidade Tecnológica Federal do Paraná*
CEP:80.230-901, Curitiba, Paraná, Brazil
tacla@utfpr.edu.br

Abstract

WebAnima 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. In this paper, we present the embodied conversational assistant and its insertion into a multi-agent system designed for research and development projects. We describe the design of the agent, highlighting the role of ontologies for semantic interpretation and the dynamic behavior of the embodied animated agent.

Keywords: Personal Assistants, Animated Agents, CSCW, Knowledge Management, Ontology.

1.Introduction

During the past few years, different projects have been developed involving the use of multi-agent systems (MAS) for improving cooperative work based on computers ([15], [16] and [17]). The first reason to employ MAS is that, like in a team, an MAS consists of a group of possibly heterogeneous and autonomous agents that share a common goal and work cooperatively to achieve it. We have been using personal assistants (PAs) coupled with MAS with success (see [5], [10] and [16] for further details). In this architecture, the MAS contain two types of agents: Service Agents (SA) providing a particular type of service corresponding to specific skills, and PAs in charge of interfacing humans to the system. In our approach, the particular skills of a PA are devoted to understanding its master and presenting information intelligently and in a timely manner. The main goal of such an agent is to reduce the user's cognitive load. The PA can be developed so as to adapt to their owner, providing the necessary semantic glue to access external services (provided by service agents) in a uniform fashion [5]. Information can be captured on the fly, improving knowledge management. This architecture was proven in many projects ([3], [5], [11] and [16]) related to cooperative design based on computers. In our approach, the PA is the only interface the system has. It may be viewed as a "portal of services", since it can delegate to other agents (service agents) the execution of tasks. Also, every agent that needs exchange information with users does it by negotiating messages with the PA. As a consequence, the interaction between agents (artificial or humans) is very intensive and should not waste users' time. Since we are considering professional applications, where users have many tasks to do and where users are using several different applications at the same time (browsers, word processors, CADs, etc.), the PA interface should captivate users to keep using their assistant. To achieve this goal, we have been developing a new personal assistant 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 [4]. They are designed to converse like a human as much as their intelligence allows [14]. In *WebAnima* the intelligent behavior is guaranteed thanks a conversational interface [8] and ontologies that support semantic interpretation. Each team member 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 [11]. A *WebAnima* agent can be used in different domains, since its knowledge about the domain and tasks to perform 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 MAS architecture. After that, we present the *WebAnima* agent. We then describe how ontologies are used for syntactic and semantic interpretation and for task representation. We also describe the embodied animated interface. Finally, we offer a conclusion and indicate some perspectives for forthcoming work.

2.Overview of the MAS Architecture

In order to better understand the MAS platform we use, this section describes an MAS architecture for knowledge management (KM) systems in research and development (R&D) projects [16]. R&D teams have no time to organize project information nor to articulate the rationale behind the actions that generate project related information. 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. Some general requirements for a KM system that guided the project are: the system must cover as much of the R&D activity as possible; it must save time by helping the user in the day-to-day activities; it must support users in creating and sharing knowledge; it must be reliable, secure and persistent.

Initially, there are two types of agents: Service Agents that provide a particular type of service corresponding to specific skills and PAs 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.

In this architecture, the PAs play a major role in the KM system. 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, as R&D members have to deal with knowledge intensive tasks, they are supposed to construct their own work methods, and in this process they should remember their past experiences, and if possible have access to other members' past experiences. Consequently, PAs must capture and represent the team members' operations, helping them in the process of preserving and creating knowledge.

The architecture contains several service agents (Figure 1). A service agent, called *Project Agent*, holds all the values shared by the group, and among them the ontologies; and the project tasks ontology (e.g. construction of a prototype). The team members can only extend the ontologies (i.e. add new subclasses). In this way, users can express their preferences, for instance, by refining a document category from the documentation ontology. So, the first thing a new user does when s/he integrates the project is to download the existing ontologies.

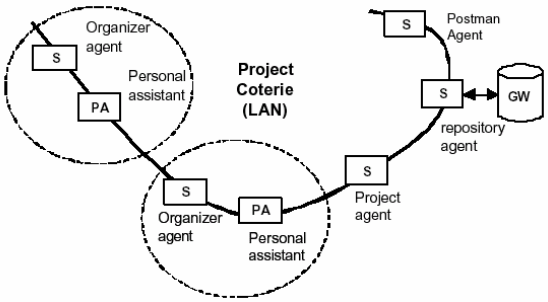


Figure 1. The MAS architecture for KM

Each PA is supposed to help its master to organize her documentation. In order to do that, each PA works jointly with its *Organizer agent*, a service agent able to build representations of the documents for later retrieval. As the architecture should be independent of any specific software tool, it integrates a service agent called a *Repository Agent*, encapsulating the groupware that must be part of the architecture. The referred agent offers services for saving and retrieving documentation but, depending on the tool it encapsulates, it can offer other kinds of services like WEB searches, or e-mail management. The agents run on a platform called OMAS (Open Multi-Agents System) [3].

In the next sections we present *WebAnima* in details.

3.The WebAnima Agent

WebAnima is a web-based embodied conversational assistant agent. *WebAnima* is, in fact, an evolution of *SpeechPa*: an intelligent speech interface for PA in research and development projects [10]. As shown in Figure 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 [13], 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.

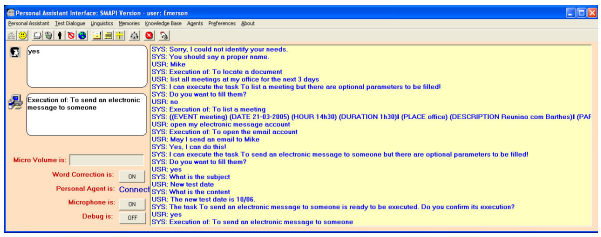


Figure 2. The *SpeechPA* interface

In *WebAnima* agent, the conversational module accepts and uses a wider set of speech acts, giving more flexibility to the agent. It also has an avatar (human-like figure) that contributes to animate the PA's use (Figure 3). Potentially, the interface becomes more agreeable to pilot.

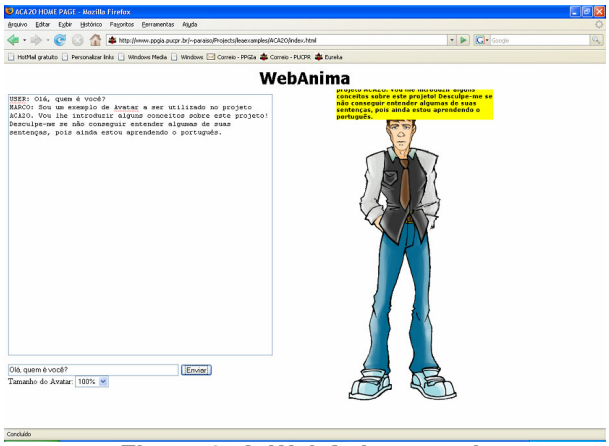


Figure 3. A *WebAnima* snaphot

The *WebAnima* agent structure is shown in Figure 4. Each agent in the MAS has a kernel with basic functionalities. Only PAs have the user interface module since in our approach only PAs interface with users.

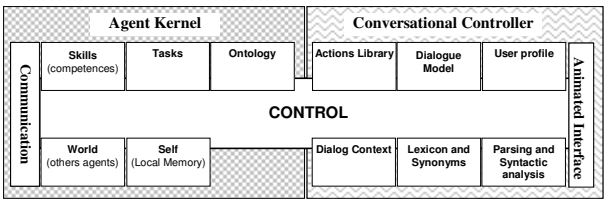


Figure 4. *WebAnima* structure

The role of the *WebAnima* agent is crucial for the effectiveness of our approach. 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 details in [11].

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.

Our agent is built around three main blocks: the user interface (a web-based animated interface implemented using the toolkit WebLEA [12] – see section 5 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 details the other two main *WebAnima* blocks: the conversational interface controller and the animated user interface.

4.The Ontology-Based Conversational Interface Controller

To produce a PA more attractive, 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 contains 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 [7].

4.1. Knowledge Handling: the Ontologies

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 [2], 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 [9] 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.

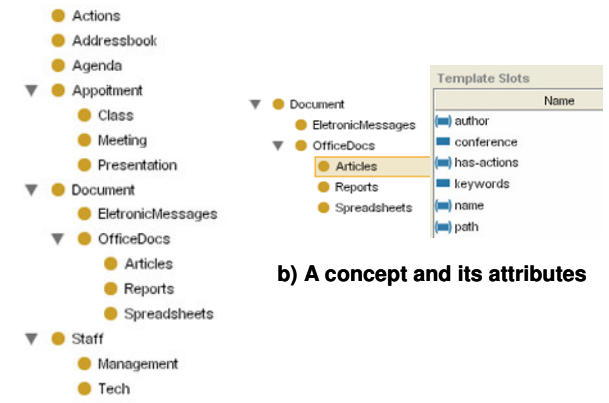
4.2. Semantic Interpretation: A Case Study

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 Figure 5a (we used Protégé [6] 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 (Figure 5b). Note that a set of actions (e.g., read, list, erase, shown in Figure 5c) may be applied to each concept, as shown Figure 5d.



a) Excerpt of the ontology

b) A concept and its attributes

c) Actions instances

d) Actions list for ***Articles***

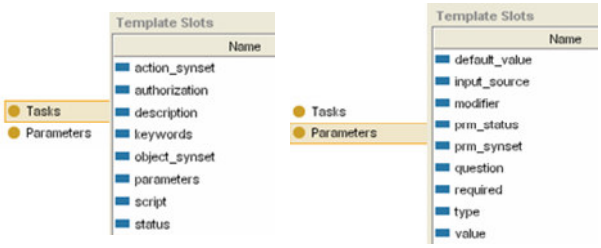
Figure 5. An excerpt of the ontology *Project*

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 (Figure 5d). 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 its property. Next, the dialogue manager takes control of the dialogue.

Tasks in our system are represented as shown in Figure 6. 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**

Figure 6. Task model

After 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 Figure 5). 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 Figure 6a). 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 Figure 6b). 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.

5. 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 (Figure 7) which are used for embodying the assistant enable to display various postures, facial expressions and gesture [1].

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 (allow or not the character "walks" 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.

Figure 7. 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 demand authorization before execute it (parameter "authorization" as shown in Figure 6a). For each task that requiring 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 demand 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, feed by the community of PAs. Each time a user accomplishes a task, her PA will register the steps needed to accomplish it (if it is 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.

6. Conclusions and Future Work

In this paper we presented a web-based embodied conversational assistant to interface users in a multi-agent-based CSCW application. With this approach, even beginners users can achieve useful and fast treatment to their requests.

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.

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.

References

[1] S. Abrilian, S. Buisine, C. Rendu, and J.-C. Martin, "Specifying Cooperation between Modalities in Lifelike Animated Agents", in International Workshop on Lifelike Animated Agents: Tools, Functions, and Applications, Tokyo, Japan, pp. 3-8, 2002.

[2] J. Allen, G. Ferguson and A. Stent, "An Architecture for More Realistic Conversational Systems", *Proceedings of Intelligent User Interfaces 2001*, Santa Fe, NM, 2001.

[3] J.-P. A. Barthès, "MASH Environments for corporate KM", *Proceedings of the Knowledge Management Workshop of IJCAI 2003*, Mexico, August 2003.

[4] T. Bickmore and Justine Cassell, "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.

[5] F. Enembreck and J.-P. A. Barthès, "Personal Assistants to improve CSCW", *Proceedings of CSCWD 2002*, Rio de Janeiro – Brazil, 2002.

[6] J. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubézy, Eiksson, N. Noy, W. Tu, *The Evolution of Protégé: An Environment for Knowledge-Based Systems Development*. 2002.

[7] N. Guarino, "Formal Ontology in Information Systems", In Proceedings of FOIS'98, Italy, IOS Press, pp. 3-15, 1998.

[8] A. Kölzer, "Universal Dialogue Specification for Conversational Systems", Proceedings of IJCAI – Knowledge and Reasoning in Practical Dialogue Systems, 1999.

[9] D. Milward, and M. Beveride, "Ontology-Based Dialogue Systems", *Proceedings of International Joint Conference on Artificial Intelligence IJCAI – 03,* Acapulco, Mexico, 2003.

[10] E. C. Paraiso, and J.-P. A. Barthes; "An Intelligent Speech Interface for Personal Assistants in R&D Projects". In: CSCWD 2005 - The 9th IEEE International Conference on CSCW in Design, Coventry - UK, v. 2. pp. 804-809, 2005.

[11] E. C. Paraiso, and J.-P. A. Barthes, "An Intelligent Speech Interface for Personal Systems in R&D Projects", in Expert Systems with Applications, v. 31, pp. 673-683, 2006.

[12] J-P. Sansonnet, J-C. Martin and K. Leguern, "A Software Engineering Approach Combining Rational and Conversational Agents for the Design of Assistance Applications", T. Panayiotopoulos et al. (Eds.): IVA 2005, LNAI 3661, pp. 111 - 119, 2005.

[13] J. R. Searle, "A Taxonomy of Illocutionary Acts", *Proceedings of Language, Mind and Knowledge*, Vol. 7, University of Minnesota Press, 1975, pp. 344-369.

[14] G. Sing, K. Wong, C. Fung and A. Depickere, "Towards a more natural and intelligent interface with embodied conversation agent", In Proceedings of International Conference on Game Research and Development, pp. 177-183, 2006.

[15] L. M. Spinosa, C. O. Quandt and M. P. Ramos, "Toward a Knowledge-based Framework to Foster Innovation in Networked Organisations", *Proceedings of CSCWD 2002*, Rio de Janeiro – Brazil, 2002.

[16] C. A. Tacla and J.-P. A. Barthès, "From desktop operations to lessons learned", *Proceedings of The Seventh International Conference on CSCW*D, Rio de Janeiro, 2002.

[17] S. WU, H. Ghenniwa and W. Shen, "User Model of a Personal Assistant in Collaborative Design Environments", *Agents in Design*, MIT, Cambridge, USA, 2002, pp. 39-54.