Computing and Informatics, Vol. 30, 2011, 829–855

ONTOLOGIES SUPPORTING INTELLIGENT AGENT-BASED ASSISTANCE

Emerson Cabrera Paraiso, Andreia Malucelli

Post-Graduate Program on Informatics
Pontifcia Universidade Catlica do Paraná
Av. Imaculada Conceio, 1155
Curitiba, Brazil
e-mail: {paraiso, malu}@ppgia.pucpr.br

Communicated by Huajun Chen

Abstract. Intelligent agent-based assistants are systems that try to simplify peoples work based on computers. Recent research on intelligent assistance has presented significant results in several and different situations. Building such a system is a difficult task that requires expertise in numerous artificial intelligence and engineering disciplines. A key point in this kind of system is knowledge handling. The use of ontologies for representing domain knowledge and for supporting reasoning is becoming wide-spread in many areas, including intelligent assistance. In this paper we present how ontologies can be used to support intelligent assistance in a multi-agent system context. We show how ontologies may be spread over the multi-agent system architecture, highlighting their role controlling user interaction and service description. We present in detail an ontology-based conversational interface for personal assistants, showing how to design an ontology for semantic interpretation and how the interpretation process uses it for semantic analysis. We also present how ontologies are used to describe decentralized services based on a multi-agent architecture.

Keywords: Ontology, personal assistant agents, intelligent assistance, semantic analysis, conversational interfaces, multi-agent systems

Mathematics Subject Classification 2000: 68T30, 68T50, 68U35

1 INTRODUCTION

As real world problems become more and more complex, intelligent artificial assistants have the potential to simplify people's professional activities. Recent research has focused on individual agents or multi-agent system (MAS) that assist humans in several and different situations, such as: in the office, at home, in medical care and in many other daily activities [1, 2, 3, 4, 5, 6] and [7]. Such agents must often monitor the evolution of a process or state over time and make periodic decisions based on such monitoring [1]. A particular case of such agents is the personal assistant (PA).

Personal Assistants help users reduce the ever-growing load of information, events and various commitments they need to handle, for instance by learning how to organize and keep track of relevant items [8]. A PA is a specialized intelligent artificial agent that helps human users to do their daily work. The interaction with an PA extends from a simple command line to a natural conversation with a human like digital actor. According to Maes [9], a PA may assist users in different ways:

- by hiding the complexity of difficult tasks;
- by performing tasks on behalf of the user;
- by training the user;
- by helping different users to collaborate;
- by monitoring events and procedures.

In many projects, PAs are also used to interface a user to a MAS. Building such a system is a difficult task that requires expertise in numerous artificial intelligence and engineering disciplines [10]. Those systems are built around powerful processing cores (reasoning, learning, scheduling and planning) connected with separate components in charge of user interaction (language processing, dialogue management, user interface controlling).

A key point in this situation is knowledge handling. How to represent knowledge is one of the most important questions for many research domains. The way an agent represents its world and the domain of its application influences how it learns, how knowledge is transferred across tasks, and how knowledge is communicatedeither between agents or to a human [11]. From the several ways to represent knowledge, ontologies are one of the most used.

The use of ontologies for representing domain knowledge and for supporting reasoning is becoming wide-spread. In the area of supporting users with specialized agents they play a key role, ranging from allowing descriptions of components of a computational workflow for earthquake simulation [12] to representing user profiles in a recommender system [13]. A PA is one of these specialized agents that can take advantage of such powerful tools [14]. In this type of situation, ontologies have been used to help interpret the context of messages sent by other agents, and to keep a computational representation of knowledge useful at inference time. The ontologies, however, may also be used for facilitating the interaction between user

and the PA. The information they contain is a source of knowledge for conversational interfaces, enhancing the quality of the assistance the PA and the MAS as a whole can offer.

In this paper we present how ontologies can be used to support intelligent assistance in a MAS context. We show how ontologies may be spread over the MAS architecture, highlighting their role controlling user interaction and service description. The paper begins by describing the MAS paradigm for intelligent assistance. Then, we present an ontology-based conversational interface for PAs. In this section we show how to design an ontology for semantic interpretation and how the interpretation process uses it for semantic analysis. After that, we present how ontologies are used to describe decentralized services based on an MAS architecture. Finally, we mention some related work, we offer a conclusion and indicate some perspectives.

2 AGENT-BASED INTELLIGENT ASSISTANCE

There has been a growing interest over the last few years in systems in which multiple agents (a community of agents or a MAS) attempt to cooperatively perform a common task. Current MASs mostly work in complex, dynamic and open environments. MAS consist of a group of possibly heterogeneous and autonomous agents that share a common goal and work cooperatively to achieve it [15]. A MAS is defined as a loosely coupled network of problem solvers working together to solve problems that are beyond the individual capabilities or knowledge of each problem solver [16].

During the past few years, different projects have been developed involving MAS for intelligent assistance ([17, 18, 19]). In the field of improving computer-supported cooperative work (CSCW), for instance, MAS has proven success in many projects [20, 21] or [22]. The application of such an approach may potentially improve the exchange of information among the participants, provide support, improve workflows and procedure controls, and provide convenient user interfaces in CSCW [23].

We have been using PAs coupled with MAS in intelligent assistance projects ([24, 23]). In our systems, all agents are cloned from a generic agent, first proposed by Ramos and Barths [25]. The generic agent contains all the basic structure that allows an agent to exist. The agent runs on a platform specially designed to implement MAS called OMAS (Open Multi-Agent System) [26], forming a community of agents called a coterie. Each agent has a kernel with basic functionalities. In OMAS, agents have a private Agent Communication Language (ACL) similar to KQML [27].

An OMAS MAS contains two types of agents: Service Agent (SA) providing a particular type of service corresponding to specific skills, and, PAs in charge of interfacing humans to the system. The particular skills of PAs are devoted to understanding their master (i.e. the user which is its owner) and to presenting the information intelligently and in a timely manner. Only PAs have the user interface

since in our approach only PAs interface with users. The user interface components will be described in Section 3.

In OMAS, MAS designers are free to implement their favourite agent coordination mechanism. In intelligent assistance projects, we have been using a coordination mechanism called Center of Services. The Center of Services architecture is characterized by the distribution of services description on each SA. This means that the PA does not know, a priori, the services (and finally the tasks) available on the MAS. To each user demand, the PA sends a consultation query (in broadcast) to the community of SAs. From the user's point of view, the system is a service center and the PA is its service provider. In this decentralized architecture, a service description is placed in each SA. It means that each SA has a task ontology describing its skills. A task model was defined allowing each SA to describe its services (further details in Section 4). This task model is known by every agent in the MAS. It is important to highlight that the PA and the SAs are continuously aware of and ready to perform those services.

In our work, ontologies are used in different ways depending on their place in the MAS. Basically, we are using a set of task and domain ontologies, separating domain and task models for reasoning. Ontologies are mainly used by PAs to interpret messages from other agents (human and artificial) and by SAs to describe their skills (tasks or services). Usually, task and domain ontologies are handcrafted using Protǵé [28]. The PA's domain ontology is obtained by manually integrating all service agents' domain ontologies and some extra information useful to control the interaction, as will be shown in details in Section 3.

In the next sections, we present the different ways we use ontologies. In Section 3, we show how ontologies are used by the PA, and in Section 4 how ontologies are used to describe agent skills.

3 AN ONTOLOGY-BASED CONVERSATIONAL INTERFACE FOR PAS

In an agent-based system, as in any other software application, an appropriate user interface is crucial. Traditionally, developers propose graphics-oriented interfaces involving the use of menus, sub-menus, dialogue-boxes, and so on. Often this approach is inappropriate or at least not very appealing, leading to an assistance of poor quality [14]. To produce a PA more attractive, from the user interface point of view, we developed a conversational interface called SpeechPA [14]. Conversational interfaces as defined by Klzer [29], let users state what they want in their own terms, just as they would do, speaking to another person. Of course, the interaction is more complex, but the complexity is handled by the agent. SpeechPA allows the PA to handle real dialogues with users, as the one shown in Table 1. This real dialogue was captured during a conversation session between a user and his PA, in a CSCW project we developed [23]. In [23] the reader can find the evaluation of SpeechPA in an experimentation.

In the context of this mixed-initiative dialogue, there is a group of engineers that cooperatively develop a new software component. Each of them has a PA. To better understand the SpeechPA capabilities, we will define a short set of competences for the PA: list all project participants, give information about a member, send messages or open the email account. This conversation is taken during John's (one of the participants) work (where PA means personal assistant and USR means the user).

PA (9h05):	*Welcome to the Voice Personal Assistant.*
USR (9h05):	Open the report on agents.
PA (9h27):	*You have a meeting scheduled for today: participant: Mike, time: 11:00, place: office, description: agent platform.*
USR (9h27):	Send an email to Mike.
PA (9h27):	*What is the subject?*
USR (9h27):	Our meeting today.
PA (9h27):	*What is the content?*
USR (9h28):	We have a meeting today. Please bring your report on agent platforms.
PA (9h35):	*You have a new email: sender: Mary.*
USR (9h35):	OK, could you open my email account?
PA (9h35):	*Yes.*
USR (9h41):	List all meetings in my office tomorrow morning.
PA (9h41):	*You have no meetings scheduled for tomorrow morning.*
USR (9h41):	What is the email address of Mary?
PA (9h41):	*The email is: mary@company.com.*
USR (9h41):	Send an email to her and to Mike.
PA (9h42):	*What is the subject?*
USR (9h42):	Setting a meeting for tomorrow.
PA (9h43):	*What is the content?*
USR (9h44):	Dear Mike, Mrs. Mary Smith is working on personal assistant agents and would like to talk with us on her projects. Are you available for a meeting in my office tomorrow at 09:00 AM?
PA (10h19):	*You have a new email: sender: Paul.*

Table 1. An excerpt of dialogue

The design and implementation of such interface is a difficult task that involves many different modules: dialogue controllers, natural language parsers, speech recognizers and synthesizers, knowledge manipulators, to list a few. Adding this type of interfaces to a PA is also a task that may require a significant effort as reported by Milward and Beveride [30].

The challenge in handling this kind of open conversation is to understand users' statements and to keep the history of objects referenced in late statements (reference resolution). To do so, we are using domain ontologies. The design of such

ontologies must cover the world surrounding the user, in terms of entities and of their relations. In addition, the ontologies must also facilitate the process of syntactic interpretation, supplying the parser with linguistic elements, such as noun synonyms, hyponyms/hyperonyms and so on.

The next section presents some principles to follow when writing ontologies for semantic interpretation. These principles arise from our experience in writing ontologies for intelligent assistance.

3.1 Domain Ontologies for Semantic Interpretation

The design of an ontology depends on its intended function (the reader is referred to [31] and [28] for a quick review of the domain.). Eriksson in [32] addresses the issues of designing ontologies for dialogue interaction and information extraction. In her work, she explores the requirements of an ontology specially designed for dialogue systems. In [32], Eriksson states that ontologies provide a common vocabulary that can be used to state facts and to formulate questions about the domain. Dzikovska et al. [33] present a method for customizing a broad-coverage parser to different domains by maintaining two ontologies (one that is generalized for language representation, and another that is customized to the domain), and for defining mappings between them.

Many researchers have applied semantic-driven approaches (e.g. [33]), searching keywords or phrases in the utterance (user statement). Our approach to semantic interpretation, however, is based on the notion that the meaning of utterances can be inferred by looking for concepts and their attributes (more details in Section 3.2).

To interpret users' statements and to manage the dialogue between user and the PA, some simple principles should be respected when writing the ontology. The domain ontology[1] shown in Figure 1 was elaborated following the principles described in the next paragraphs.

In its first version, this ontology was written in the context of a research and development MAS application. It is composed by 108 concepts.

3.1.1 Definition of Concepts and Their Relations

Concepts and their relations (binaries) are the basic elements that constitute an ontology. The use of hyponymy and hyperonymy to link concepts (is-a relation) and meronymy to describe dependency (has-a) may help interpret incomplete or unexpected statements. For instance, the user demands to the PA: List all articles on agents. According to the ontology (Figure 1), finding conference articles or journal articles in the statement could be easier to interpret. However, thanks to the hyperonymy relation between articles and conference/journal, a formal representation would be obtained as well. Acting in the same way, the reference resolution may be guaranteed as shown in the following short dialogue fragment:

[1] We intentionally reduced the ontology to a bare minimum, extracting some concepts, properties, relations and labels.

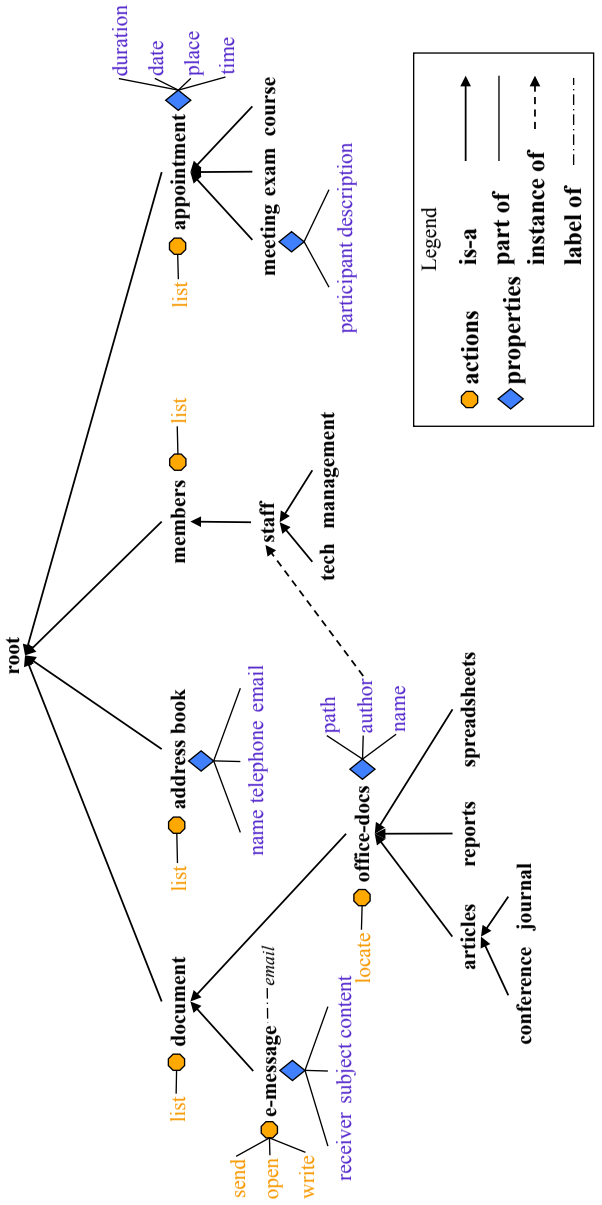

Fig. 1. An excerpt of the domain ontology

PA: Which documents do you want to list?
USR: Articles.
PA: Do you mean conference articles or journal articles?
The semantic analyser identified articles as a hyponymy of documents (is-a relation).
It is also helpful to define a list of synonyms to each concept.

3.1.2 Definition of Properties

To describe each property of a concept, one should define its type, a list of synonyms, its cardinality and a domain restriction. Domain restriction should be one of the following:

- Time: for temporal attributes (e.g. appointment: time, duration or hour);
- Space: for geographic localization;
- People: for describing people;
- General: for others attributes.

A restriction is especially useful when interpreting questions (see examples in Section 3.2).

3.1.3 Definition of Actions

An action is a token used to describe operational tasks applied to a concept. An action token is used by service agents to identify whether it is related to its domain or skills or not. Each action token has a skill associated. A skill can be expressed as a procedure or a rule. Actions are inherited as shown in Figure 2: list, linked to document is an action inherited by articles and reports.

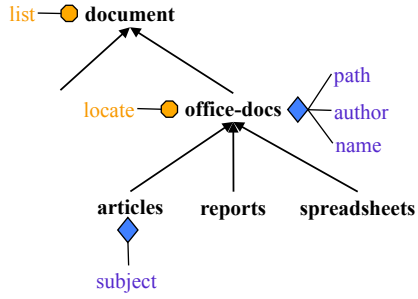

Fig. 2. An excerpt of the ontology: concept document

3.1.4 Definition of Multiple Instances

Since ontologies should be able to incorporate several views of a domain, e.g. user and system, or several different information sources, as advocated by Eriksson [32],

multiple instantiation is important. Multiple instantiation is useful for defining the list of tasks in the task ontology, as well.

The resultant ontology contains domain information but also lexical information and information on synonyms. To avoid any misunderstanding, it is important to highlight that the main ontology structure (concepts and their relations) construction is guided by the domain application and not by its utilization in the semantic interpretation system.

3.2 Semantic Interpretation Process

The process of interpreting a user statement is carried out in two steps:

- parsing and syntactic analysis; and
- semantic interpretation (domain ontology application).

The results of this process are sent to the dialogue manager continuously or back to the user when they do not make sense.

The parsing algorithm works top-down, replacing each utterance stem with its syntactic category (verb, noun, adverb, etc) with the help of lexicon files and a set of grammar rules.

In order to reduce the interaction and consequently avoid wasting time, we limited the space of dialogue utterances to directive speech act classes [34] – inform, request, or answer – since such classes define the type of expected utterances in a master-slave relationship. A speech act is an act that a speaker performs when making an utterance. The idea is to do something by looking for the related acts in the utterance. The strategy of treating directive speech acts reduces the number of turn takings since some speech acts, like acknowledgement acts ("Thank you" or "Have a nice trip") will not be used by the PA. In our applications, a typical utterance could be: "Look for a document on agents in the database." According to our taxonomy this is an order utterance and can be processed by the grammar rules. If a sentence is not well formed, or if it is out of the domain, then it is classified as a nonsensical utterance. In such a case, the user is invited to reformulate his/her sentence.

The approach to semantic interpretation presented here is based on the notion that the meaning of user statements can be inferred by looking for concepts and their attributes. More precisely, the module responsible for applying the ontology to the statement searches for domain concepts and the list of verbs that indicates the task to be executed. The corresponding keywords are concepts of the ontology directly related to a list of actions. We believe that this approach is ideal for applications where the domain is well known and restricted. In contrast, statistical models as proposed by [35] bring the analysis to the parser level, leaving useful domain information out of the process.

After interpreting a user entry, we have a formal representation of it. Thus, this formal representation is the semantic analysis result. The formal representation

is a well-formed computational formula. This formula is a list built respecting the BNF (BackusNaur Form) specification shown in Table 2.

<formula> ::= '(' <action> 1*<ontology-components> 1*')'
<action> ::= <token>
<ontology-components> ::= '(' 0*<attribute-name> 0*'(' <concept> ')'
<attribute-name> ::= <token>
<concept> ::= <concept-name> 0*<attribute-list>
<concept-name> ::= <token>
<attribute-list> ::= '(' ':' <attribute-name> <attribute-value> ')'
<attribute-value> ::= <token> — 'nil'
<token> ::= string

Table 2. BNF specification of the formal representation

To exemplify the semantic interpretation process, let us introduce the following entry:
USR: *Do I have a meeting with Mike in my office?*
Firstly, the statement is parsed and a syntactic tree is obtained (Figure 4) according to the algorithm shown in Figure 3.

To interpret the given utterance, first the parser checks the context of the input, verifying that it is a question related to the domain. To do so, it uses the domain ontology and the lexicon. The lexicon contains thousands of words extracted from WordNet [36] and enriched with the list of all concepts and attributes of the ontology. Additional information is collected, by identifying relations between pairs of words, based on a link grammar adapted from Link Parser [37].

The statement is classified as a question so as to be able to follow the conversation, the PA must answer the given question. In order to answer the question, the PA builds the formal representation of it, before sending it to the concerned service agent:

```
(list (Meeting (:participant "Mike") (:place "office")))
```

This list was obtained from an algorithm that searches for domain concepts. The VO (verb-object) link between Have and Meeting points the phrase object. The algorithm searches an equivalent concept in the domain ontology (Figure 1). It finds Meeting, a hyperonymy of Appointment. Comparisons between words and concepts (and their synonyms) are made with help of a word similarity mechanism.

After selecting the candidate concept, the algorithm starts a process to try to fill up all concept properties. The Meeting concept has six properties: duration, date, place, time, participants and description. Note that the parser identified two constituents indicated between bracts ('[' and ']'). Each constituent has a potential concept property on it. Each instance of a concept is kept in memory for future use (stack of instances). For this type of question, the action list is systematically used.

A second example is shown in Figure 5.

```
Data: utterance utti
Data: syntactic tree treei
Ontology: R&D ontology ont
Lexicon: lexicon lx
Result: list luti

treei = parseUtterance(utti, ont, lx);
if treei <> Ø then
    Verify which speech act is applicable to utti;
    switch speechAct do
        case "order":
            Search the verb to be assigned as the action acti;
            acti = searchVerb(treei);
        case "question":
            Assume acti = "list";
        case "answer":
            Send information to the Dialogue Manager for treatment ;
            return;
    end-switch;
    Search the set of concepts Cpti in treei;
    for each Cptk do
        Create an instance in memory and fill up all found attributes;
        push(Cptk); //Stack of instances
    end-do;
    luti = buildList(acti, Cpti);
else
    luti = "out of the domain";
end-if
```

Fig. 3. The algorithm to produce the formal representation

To interpret this statement, the extract of ontology shown in Figure 5 is used. Three main elements are easily selected: an object (email, label of e-message) linked to the ontology concept e-message, one parameter (receiver), and one action (send). Following the process just explained, and using the syntactic tree shown in Figure 6, the formal representation can be obtained:

```
(send (e-message (:receiver "Mike")))
```

Note that the specific action will be executed by a service agent. The way it is done is shown in Section 4. The treatment of user questions is similar. After the syntactic analysis, the questions are classified according to two different types: direct or conditional question. Direct questions are those known as *wh-questions* (what, where, when, who, etc.) leading directly to an action execution. Conditional questions (can/could, may/might, do/does, etc.) lead the dialogue manager to interpret the result before being able to respond them to the user. The process of interpreting a direct question is particular, since the dialogue manager does not search for an action to be executed. Instead, it searches for an object desired by the user. For instance, assume the following question:
USR: *Who is the author of the article entitled agents?*

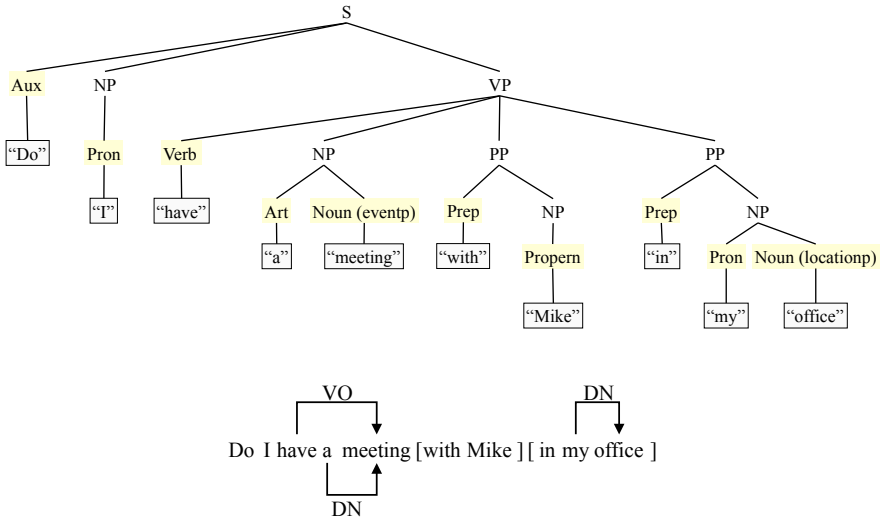

Fig. 4. The syntactic tree and some additional information for *'Do I have a meeting with Mike in my office'*

Fig. 5. Another example of semantic interpretation

This question is composed by one complement, containing the concept "article" and one attribute (name). One important link (verb-object) is also found (be-author):

In this case, the action list is systematically used, generating the formal representation below:

```
(list (author (Article (:title "agents"))))
```

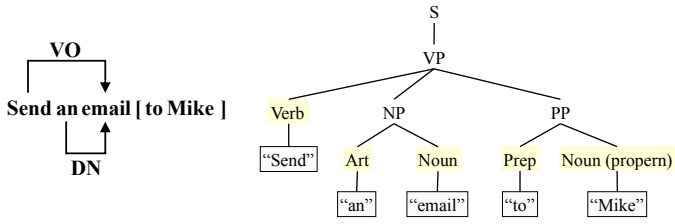

Fig. 6. The syntactic tree and some additional information for 'Send an email to Mike'

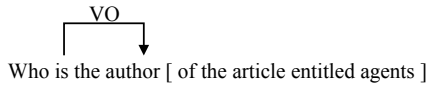

Fig. 7

The direct question presented below is more complicated to be treated:
USR: *Where is the meeting with Paul?*

The information which the user looks for is indirectly referenced. When the user uses *where*, s/he actually wants to known where the meeting will take place. In this case, the semantic analyser will use the restrictions of each found concept attribute. The attribute *place* has the restriction *space*, indicating the geographic location of the meeting. Thus, the formal representation may be obtained:

```
(list (place (Meeting (:participant "Paul"))))
```

The last example of direct question is shown here:
USR: *When is the meeting at my office with Paul?*

This question presents a statement without direct identification of a desired object. Also, when searching for attributes compatibles with *when* (restriction *time*), one can find three different attributes: *date*, *duration* and *time*. Formal representation of the result contains the three attributes as unknown information:

```
(list (date duration time (Meeting
(:place "office"
 :participant "Paul"))))
```

The interpretation process for conditional questions has a different workflow: first the PA tries to find an answer for the question and, when successful, it answers the question before presenting the received result. For instance, let us introduce the question below:
USR: *Do I have a meeting today?*
The following formal representation is obtained:

```
(list (Meeting (:date "28-03-2007")
               :participant "John")))
```

Before presenting the answer, the dialogue manager will present a "yes" if it has a meeting or only a "no" if it does not have a meeting.

Once the formal representation has been obtained, in all cases, one or several service agents will receive it and a response may be received by the PA if at least one of them can solve it.

The ontologies are also used to solve another important problem on dialogue management: the reference.

3.3 Reference Resolution

Reference resolution is a complex problem being object of active research on linguistics (see [38] and [39] for details). The reference in dialogue systems is the study of how objects from the real world are presented inside the utterances, how they should be stored and how they can be used for keeping the conversation context. The simplest form of reference appears when a new object or a past introduced object is found in a statement. To better understand, let us consider the excerpt of dialogue between a user and his/her PA shown in Table 3.

USR (1)	Give me the list of articles written by Mike Palmer.
PA (2)	*(PA performs the task: list of articles from Mike Palmer)*
USR (3)	Is he a participant of the project?
PA (4)	*Yes*
USR (5)	Do you known his telephone number?
PA (6)	*Yes. The telephone number is: 041 3271 1515*
USR (7)	and his address?

Table 3. An excerpt of dialogue between a user and his/her PA

In the first statement (1), the user cites the name of Mike Palmer for the first time. When the system wants to keep the context of the conversation, it should store that information. Then, in the next statement (3), the user cites Mike again, through the pronoun *he*. This phenomenon is known as anaphora, e.g., the fact of to reference an entity already introduced in the dialogue.

Another well known phenomenon in a dialogue is the occurrence of an ellipsis. The detection of an ellipsis is done by analysing a statement that is incomplete at a first glance, but where the "missing parts" were presented in previous statements. In the dialogue, at the statement (5) the user demands the telephone number of Mike and in the statement (7) s/he adds a demand of his address. Here, the statement (7) can be interpreted as a complement of the statement (5).

In the actual version of our system, we only treat the presence of an anaphora. To do so, after interpreting each user statement, the dialogue manager creates or updates the instances used to interpret the given statement. For example, see the short dialogue below:

USR (1): What is the email address of Mary?
PA (2): The email is: mary@company.com.
USR (3): Send an email to her.

By interpreting the statement (1), the semantic analyser builds the formal representation:

```
(list (address (AddressBook (:name "Mary"))))
```

At the same time, the dialogue manager stores in a stack of instances of concepts, the instance below:

```
(e-message (:receiver "Mary")
           (:subject nil)
           (:content nil))
```

The dialogue manager may now to solve the reference to Mary in (3) thanks to the instances stored in the stack, since probably the object just referred appeared in the latest statements. The same occurs in the next short dialogue:

USR (1): What is the starting time of the meeting with Mike?
PA (2): Starting time is: 14h00
USR (3): Where is it planned to be?

The presence of the pronoun *it* in (3) leads the dialogue manager to search for an instance of a compatible concept. In the top of the stack of instances it will find the instance:

```
(Meeting  (:date "17-05-2006")
  (:time "14h00")
  (:place "office")
  (:duration "1h00")
  (:participant "Mike")
  (:description nil))
```

The response is then posted to the user since all needed information is already known.

3.4 Impossible Demands

One last interesting advantage of using ontologies to support intelligent dialogue is to handle impossible demands like the one shown here:

PA (1): You have a new email: sender Mike Palmer.
USR (2): I would like to read it.
PA (3): (PA opens the email client)
USR (4): Erase this email.
PA (5): I can't do what you want. However, I can: send an electronic message or open the email account. These tasks are related to email.

According to the extract of ontology shown in Figure 5, "erase" is not a valid action. The only possibility the PA has is to refuse this demand. It can, however, make some suggestions based on tasks related to the actions *send* and *open*. In order to inform the user the correct information, the PA sends out, in broadcast, a message called REQUEST-HELP-INFO. Receiving this message, each SA formats an INFO HELP-REQUESTED message containing its services description (in free text) related to a specific concept.

We have shown in this section how ontologies may support knowledge in a conversational interface in the context of intelligent assistance. The next section presents the use of ontologies to describe services performed by service agents.

4 DECENTRALIZED ONTOLOGY-BASED DESCRIPTION SERVICES

As we mentioned in Section 2, our systems are developed using OMAS [26]. In OMAS, MAS designers have the freedom to implement their favourite agent coordination mechanism. In intelligent assistance projects, we have been using the *Center of Services architecture*. Figure 8 shows schematically how this architecture works.

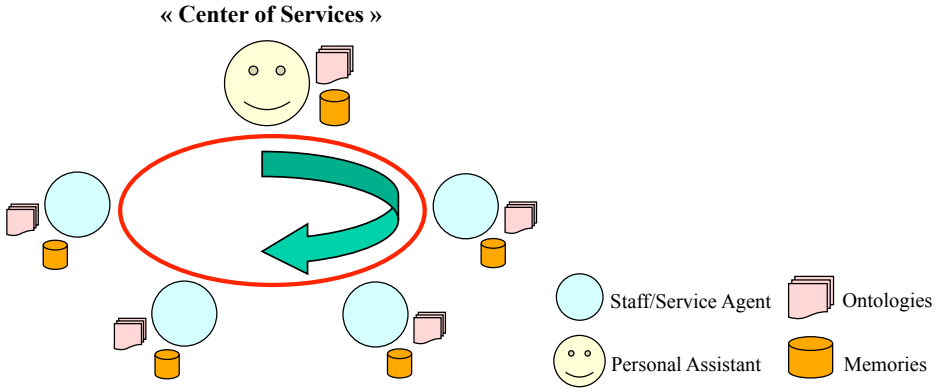

Fig. 8. Coordination architecture: Decentralized

The Center of Services architecture is characterized by the distribution of services description on each SA. This means that the PA does not know, a priori, the services (and finally the tasks) available on the MAS. To each user demand, the PA sends a consultation query (in broadcast) to the community of SAs. From the user's point of view, the system is a Service Center and the PA is its Service Provider.

In this decentralized architecture, a service description is placed in each SA. It means that each SA has a task ontology describing its skills. Using the description of a service, the service agent may start the execution of a task, negotiating further information with other agents, if needed. In order to be able to do that, the task ontology is crucial, its role being evidenced by the example shown in Figure 11. Ontologies were chosen to represent tasks because their expressiveness allows a structured representation of each service. A task model was defined allowing each SA describe its services, as shown in Figure 9.

The process of selecting and executing a task is presented in the sequence diagram shown in Figure 10: a negotiation process must be started in order to select a SA (and a task), to fill out all task parameters and to execute a selected task. SAs are able to interpret a query and to format a description message containing the task description. In order to facilitate agent communication, a content language was defined. This language has a set of 10 messages allowing task controlling and distribution (main messages in Table 4).

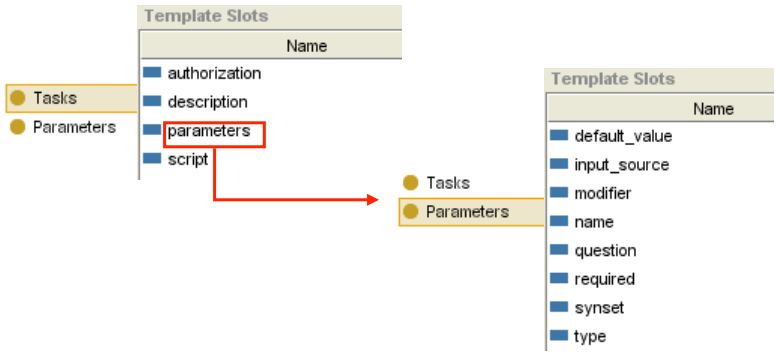

Fig. 9. The task model and its attribute parameters in details

(TASK-SEACRH script msg-id)	Message sent by a PA to look for a SA able to accomplish a specific task
(TASK-FORMAT task-description msg-id)	Message sent by a SA to a PA containing a task description (missing information description to be collected from the user)
(TASK-REQUEST script msg-id)	Message sent by an agent to another agent demanding a task execution
(TASK-RESULT msg-id response context)	Message containing the result of a task execution

Table 4. Some content messages

Once the PA receives a task description, it can demand the missing information from the user. The information is sent to the SA allowing the task execution.

The main advantage of this architecture is that new agents (and their services) may easily enter and quit the MAS without interrupting the PA work. This architecture is more suitable to the kind of projects we have developed. Its flexibility allows the offer of new services "on the fly".

The next paragraphs present a simple example to illustrate the whole process. After interpreting the user statement:
USR: Send an email to Mike
The PA is able to construct the formal representation of it:

```
(send (e-message (:receiver "Mike")))
```

The PA formats a TASK-SEARCH message and sends it (in broadcast) to the community of agents:

```
(TASK-SEARCH (send (e-message (:receiver "Mike"))) 25)
```

By sending the TASK-SEARCH message, the PA intends to either receive the task result (only if a SA is capable to send it using only the information already

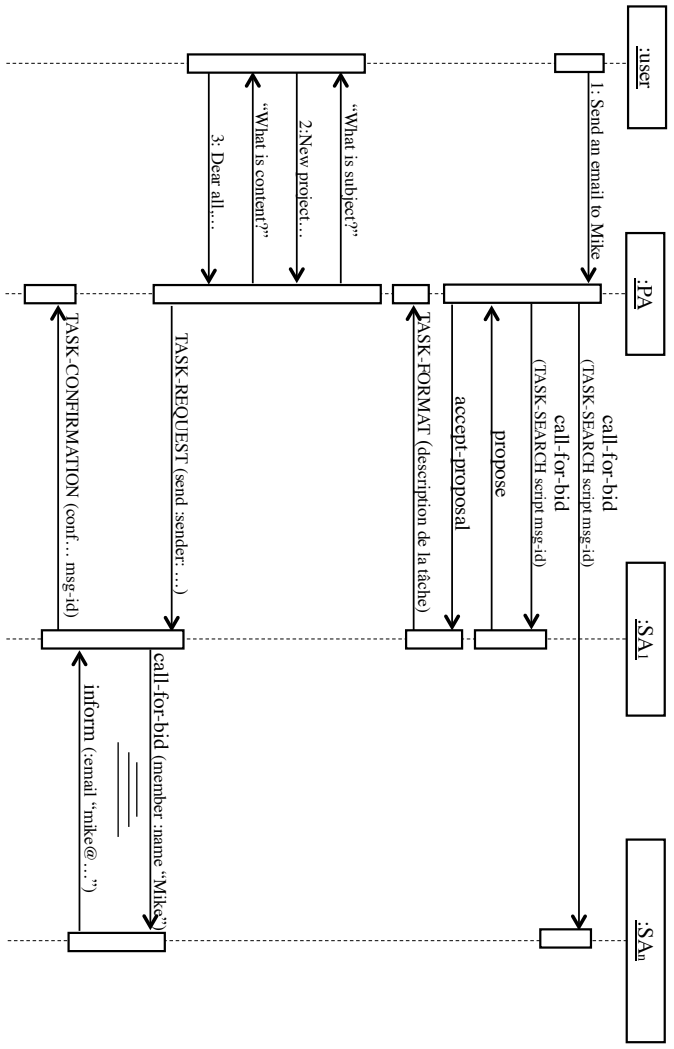

Fig. 10. Sequence Diagram for the task filling process

known) or a TASK-FORMAT message specifying the information that should be collected from the user. The process of sending and collecting messages is controlled by OMAS. Figure 11 shows the XML representation of the task *Send an Electronic Message*.

After receiving a TASK-SEARCH message, a SA analyses its content (field *script*), using its local ontology. The concerned SA formats a TASK-FORMAT message and sends it to the PA:

```
- <Task>
    <description>To send an electronic message to someone</description>
  - <prm prm_id="1">
      <name>RECEIVER</name>
      <synset>receiver to</synset>
      <question>Who is the receiver</question>
      <type>propern</type>
      <modifier>none</modifier>
      <required>true</required>
      <input_source>vocal</input_source>
      <default_value>none</default_value>
    </prm>
  - <prm prm_id="2">
      <name>SUBJECT</name>
      <synset>subject about</synset>
      <question>What is the subject</question>
      <type>text</type>
      <modifier>none</modifier>
      <required>false</required>
      <input_source>vocal</input_source>
      <default_value>none</default_value>
    </prm>
  - <prm prm_id="3">
      <name>CONTENT</name>
      <synset>content text</synset>
      <question>What is the content</question>
      <type>text</type>
      <modifier>none</modifier>
      <required>false</required>
      <input_source>keyboard</input_source>
      <default_value>none</default_value>
    </prm>
    <script>@SENDEMAIL</script>
    <authorization>true</authorization>
  </Task>
```

Fig. 11. A task example: "Send an Electronic Message"

```
(TASK-FORMAT
          (<Task>
    <prm prm_id="2">
       <name>SUBJECT</name>
       <synset>subject about</synset>
       <question>What is the subject</question>
       <type>text</type>
       <required>true</required>
       <input_source>vocal</input_source>
       <default_value>none</default_value>
    </prm>
    <prm prm_id="3">
       <name>CONTENT</name>
       <synset>content text</synset>
       <question>What is the content</question>
```

```
       <type>text</type>
       <required>true</required>
       <input_source>keyboard</input_source>
       <default_value>none</default_value>
    </prm>
       <script>@SENDEMAIL</script>
       <authorization>true</authorization>
          </Task>)
25)
```

Once the PA receives the message, it sends it to the dialogue manager that starts the process of filling the task parameters. Note that to obtain the subject or the *content* parameter, the PA will present the question field content: *What is the subject* or *What is the content*. Also note that an instance of the concept *e-message* is progressively updated. After filling the parameters, the PA writes the script field and sends it to the SAs using a TASK-REQUEST message:

```
(TASK-REQUEST (@SENDEMAIL (: receiver "Mike")
   (: subject "New project meeting")
   (: content "Dear Paul ")) 25)
```

One SA may interrogate another SA in order to execute a task. As shown in the sequence diagram (Figure 10), the SA agent responsible for sending an electronic message may send a TASK-REQUEST to the community of SAs to obtain the electronic address of Mike:

```
(TASK-REQUEST (list (address (AddressBook (:name "Mike")))) 25)
```

After receiving the response, the SA may effectively execute the task (to send the electronic message) and to confirm its execution by sending a TASK CONFIRMATION message to the PA:

```
(TASK-CONFIRMATION 25 "The email was sent with success")
```

A SA may decide with a TASK-REQUEST or a TASK-SEARCH matter for it by inspecting its domain ontology and matching with the field *script*. Each SA has the ability to interpret each message that arrives to its mailbox.

In this section, we presented the way ontologies are used to describe agent's skills (tasks). By describing an example, we intended to show in practice how intelligent assistance is enhanced thanks to ontologies.

5 RELATED WORK

In the past years there had been increasing activity in the area of MAS and intelligent assistance. In this discussion we focus on how researches and their projects deal with knowledge when providing service description and when designing the user interface.

The Smart Personal Assistant (SPA) is a research project focused on natural language interaction with personal assistant systems for use on mobile devices such as PDAs (Personal Digital Assistants) and mobile phones [40]. The current SPA is a personal information management assistant that provides users with integrated access to e-mail and calendar information. Their conception of personal assistants is different from own ones. In our architecture PAs interface users with the system and services are executed by SAs. In their architecture, a user may have many personal assistants and "wrapper" agents for task providing. In SPA, there is also a coordinator agent. The coordinator is built using a BDI (Belief, Desire, Intention) agent architecture (see [41] for more details on this architecture) in which both dialogue management and coordination of the task assistants are encoded in the agent's plans. Domain knowledge is placed in the Coordinator agent, centralizing all interpretation. The centralization is very dangerous since an interruption in the Coordinator agent may halt the whole system.

Guzzoni et al. [42] propose an approach called Active Ontologies and a tool called Active, to model all aspects of an intelligent assistant: ontology-based knowledge structures, service-based primitive actions, composite processes and procedures, and natural language and dialogue structures. They define Active Ontologies as a processing formalism where distinct processing elements are arranged according to ontology notions. The Active tool provides several components to developers, among them a component to write natural language enabled interfaces. For service description, they choose a dynamic service broker approach, where a service registry accepts a set of facts from each service provider, containing its capabilities and attributes.

CommomKADS is a methodology proposed by Laclavik [43] used for knowledge modelling in a MAS context for KM. The work's aim is to make a connection between KM and MAS, mainly by bringing work done in the semantic web area to MAS. In this work, an agent knowledge model is presented, which can model agents environment, agent context and agent resources obtained as result of agent behaviour. Agents are based on a behavioural architecture, with a memory formed by sets and description logic, subsequently implemented using RDF/OWL (Resource Description Framework/Web Ontology Language). This model is based on the JADE (Java Agent DEvelopment Framework) [44] ontology model. As the authors explain, the JADE Ontology model has some limitations. A model based on Java classes can not support multiple inheritance, inverse concepts and other features of semantic ontology representations.

ASWAD (Agent-Supported Workflow in Public Administration) [45] is a European funded project that aims at providing public administrations with a unified and flexible Internet application for organizing cooperative work practices. The ASWAD tool is based on a groupware system with built-in workflow management and PAs. The system includes components for calendaring, contacts management, email handling and document management. The PA enables users to filter and process information, and to automatically delegate tasks to others. The PA's interface comprises an animated avatar (a companion), accepting very simple control

commands such as open, cancel, erase, etc. The use of such animated agents may, however, disturb the whole system performance and they are often ignored by users, as argued by Kramer et al. in [46].

Purver et al. [47] present a personal office assistant capable of understanding multi-party discourse. The assistant is not capable of direct interaction with the user, but they intend to develop a system capable of understanding, describing and automatically participating in the discussion during meetings. Their approach is also based on ontologies, used to describe communicative actions (concepts related to meetings).

Bai [48] studied how to coordinate and manage agents' behaviours and make them work cooperatively in a MAS. In this work, authors firstly classify the different kinds of knowledge in MASs into different categories; then, they define an ontology format to represent knowledge; finally, they introduce a framework for agent coordination through ontology management. Ontologies are used in this coordination framework to face dynamic and changeable environments.

A growing number of researchers, like Quesada et al. [49] or Yates et al. [50], are working with interfaces to household appliances. They are proposing speech interfaces in their projects, but they do not consider a very important issue: intelligent support. None of them uses an intelligent and specialized mechanism to support the user, as we did using the assistant agent approach. A PA is a piece of software devoted to understanding its master and presenting the information intelligently and in a timely manner. Thus, the union of a speech interface and an assistant agent is a good solution for user assistance.

6 CONCLUSIONS

In this paper we presented how ontologies may be used to support intelligent assistance in a MAS context. We described how ontologies are spread over the MAS architecture, highlighting their role controlling user interaction and service description. Using ontologies as the support to implement conversational interfaces, one can develop intelligent user interfaces through which even inexperienced users can have their requests treated in a fast and useful manner, guarantying predictability. Rather, the PA provides 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 list of applicable tasks described in the ontology.

We have performed some experiments to evaluate the conversational interface and its dialogue system, in order to measure, among others, how fast and useful users' requests are treated (further details in [23] and [51]). The average time elapsed to start a task was 17.4 seconds (to select a query from the user's utterance, to fill its respective parameters and to query a service agent to perform the recognized task). To accomplish five different tasks, users needed about 32 turns.

Our goal with this paper is to show those ontologies provide a semantic framework for knowledge management, a key point in a MAS. The use of ontologies

for intelligent assistance is an interesting and not exhausting explored field of research.

We are currently working in an intelligent agent architecture for on-line assistance [24]. The intent of this architecture is to provide a framework to develop web-based intelligent agents that give useful information on a specific domain. In this context, ontologies are used to interpret users' demands and also to generate natural language responses.

We are also working on improving the reference resolution treatment. The actual version only deals with anaphora. For the next version, we intend to add the treatment of ellipsis as well.

Future work also includes improving 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.

REFERENCES

[1] Varakantham, P.—Maheswaran, R.—Tambe, M.: Exploiting Belief Bounds: Practical Pomdps for Personal Assistant Agents. In Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '05), New York, NY, USA 2005, ACM, 2005, pp. 978–985.

[2] Tacla, C.—Barthès, J. P.: A Multi-Agent Architecture for Knowledge Acquisition. Agent Mediated Knowledge Management at AAAI Spring Symposium, 2003.

[3] Schreckenghost, D.—Martin, C.—Bonasso, P.—Kortenkamp, D.—Milam, T.—Thronesbery, C.: Supporting Group Interaction Among Humans and Autonomous Agents. In Proceedings of the AAAI02 Workshop on Autonomy, Delegation, and Control, AAAI Press 2002, pp. 361–369.

[4] Pollack, M. E.—Brown, L.—Colbry, L.—McCarthy, C. E.—Orosz, C.—Peintner, C.—Ramakrishnan, S.—Tsamardinos, I.: Autominder: An Intelligent Cognitive Orthotic System for People With Memory Impairment. 2003.

[5] Magni, P.—Bellazzi, R.—Locatelli, F.: Using Uncertainty Management Techniques in Medical Therapy Planning: a Decision-Theoretic Approach. In Applications of Uncertainty Formalisms, Springer, 1998, pp. 38–57.

[6] Wong, A. K. Y.—Yip, F.—Ray, P. K.—Paramesh, N.: Towards Semantic Interoperability for It Governance: an Ontological Approach. Computing and Informatics, Vol. 27, 2008, No. 1, pp. 131–155.

[7] Wang, P.—Xu, B.: Debugging Ontology Mappings: A Static Approach. Computing and Informatics, Vol. 27, 2008, No. 1, pp. 21–36.

[8] Richard, N.—Yamada, S.: An Adaptive, Emotional, and Expressive Reminding System. 2007.

[9] Maes, P.: Agents That Reduce Work and Information Overload. Commun. ACM, Vol. 37, 1994, pp. 30–40.

[10] Winikoff, M.—Padgham, L.—Harland, J.: Simplifying the Development of Intelligent Agents. In Proceedings of the 14th Australian Joint Conference on Artificial

Intelligence: Advances in Artificial Intelligence, AI '01, London, UK, Springer-Verlag 2001, pp. 557–568.

[11] Beeson, P.—Macmahon, M.—Modayil, J.—Morarka, A.—Kuipers, B.—Stankiewicz, B.: Integrating Multiple Representations of Spatial Knowledge for Mapping, Navigation, and Communication. Interaction Challenges for Intelligent Assistants at AAAI Spring Symposium, 2007.

[12] Kim, J.—Spraragen, M.—Gil, Y.: An Intelligent Assistant for Interactive Workflow Composition. In Proceedings of the 9th International Conference on Intelligent User Interfaces (IUI '04), New York, NY, USA, ACM 2004, pp. 125–131.

[13] Middleton, S. E.—De Roure, D. C.—Shadbolt, N. R.: Capturing Knowledge of User Preferences: Ontologies in Recommender Systems. In Proceedings of the 1st International Conference on Knowledge Capture (K-CAP '01), New York, NY, USA, ACM, 2001, pp. 100–107.

[14] Paraiso, E. C.—Barthès, J. P. A.: Speechpa: an Ontology-Based Speech Interface for Personal Assistance. Intelligent Agent Technology, IEEE/WIC/ACM International Conference, 2005, pp. 657–663.

[15] Sycara, K.: Multi-Agent Systems. AI Magazine, Vol. 19, No. 2.

[16] Durfee, E. H.—Lesser, V. R.: Distributed Artificial Intelligence. Vol. 2, Ch. Negotiating task decomposition and allocation using partial global planning, pp. 229–243, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA 1990.

[17] Enembreck, F.—Barthès, J.-P.: Personalization in Multi-Agent Systems. Intelligent Agent Technology, IEEE/WIC/ACM International Conference 2005, pp. 230–233.

[18] Tavares, T. A.—Oliveira, S. A.—Canuto, A.—Gonalves, L. M.—Filho, G. S.: A Multi-Agent System for 3D Media Spaces Assistance. 2005 Web Congress, Latin American, pp. 166–175.

[19] Molina, M.—Blasco, G.: A Multi-Agent System for Emergency Decision Support. In Lecture notes in computer science, Springer, 2003, pp. 43–51.

[20] Shen, W.—Wang, L.: Web-Based and Agent-Based Approaches for Collaborative Product Design: An Overview. International Journal of Computer Applications in Technology, Vol. 16, 2003, pp. 103–112.

[21] Spinosa, L. M.—Quandt, C. O.—Ramos, M. P.: Toward a Knowledge-Based Framework to Foster Innovation in Networked Organisations. Proceedings of CSCWD 2002.

[22] Wu, S.—Ghenniwa, H.—Zhang, Y.—Shen, W.: Personal Assistant Agents for Collaborative Design Environments. Computers in Industry, Vol. 57, 2006, No. 8–9, pp. 732–739, Collaborative Environments for Concurrent Engineering Special Issue.

[23] Paraiso, E. C.—Barthès, J. P. A.: An Intelligent Speech Interface for Personal Assistants in Rd Projects. Expert Systems with Applications, Vol. 31, 2006, No. 4, pp. 673–683, 2006, Computer Supported Cooperative Work in Design and Manufacturing.

[24] Paraiso, E. C.—Campbell, Y.—Tacla, C. A.: Webanima: A Web-Based Embodied Conversational Assistant to Interface Users With Multi-Agent-Based Cscw

Applications. 12th IEEE International Conference on CSCW in Design, 2008, Vol. 1, pp. 337–342.

[25] Ramos, M. P.: Structuration et Evolution Conceptuelles d'Un Agent Assistant Personnel Dans Les Domaines Techniques. 2000.

[26] Barthès, J. P. A.: Omas v 1.0 Technical Reference. Tech. rep., HEUDIASYC UMR 6599, Université de Technologie de Compiègne 2000.

[27] Finin, T.—Fritzson, R.—McKay, D.—McEntire, R.: KQML as an Agent Communication Language. In Proceedings of the Third International Conference on Information and Knowledge Management (CIKM '94), New York, NY, USA 1994, ACM, 1994, pp. 456–463.

[28] Gennari, J. H.—Musen, M. A.—Fergerson, R. W.—Grosso, W. E.—Crubzy, M.—Eriksson, H.—Noy, N. F.—Tu, S. W.: The Evolution of Protégé: an Environment for Knowledge-Based Systems Development. International Journal of Human-Computer Studies, Vol. 58, 2002, pp. 89–123.

[29] Kölzer, A.: Universal Dialogue Specification for Conversational Systems.

[30] Milward, D.—Beveride, M.: Ontology-Based Dialogue Systems. In International Joint Conference on Artificial Intelligence, Acapulco, Mexico 2003.

[31] Pinto, H. S.—Martins, J. P.: Ontologies: How Can They Be Built? Knowledge and Information Systems. Vol. 6, 2004, pp. 441–464, 10.1007/s10115-003-0138-1.

[32] Flycht-Eriksson, A.: Design of Ontologies for Dialogue Interaction and Information Extraction. 2003.

[33] Dzikovska, M. O.—Allen, J. F.—Swift, M. D.: Integrating Linguistic and Domain Knowledge for Spoken Dialogue Systems in Multiple Domains. 2003.

[34] Searle, J.: A Taxonomy of Illocutionary Acts. Minneapolis, University of Minnesota Press, 1975, pp. 334–369.

[35] He, Y.—Young, S.: Semantic Processing Using the Hidden Vector State Model. Computer Speech and Language, Vol. 2005, pp. 8–106.

[36] Fellbaum, C.: Wordnet: an Electronic Lexical Database. 1998.

[37] Grinberg, D.—Lafferty, J.—Sleator, D.: A Robust Parsing Algorithm for Link Grammars. In In Proceedings of the Fourth International Workshop on Parsing Technologies, 1995.

[38] Issco, A. P. B.—Popescu-Belis, A.: Reference Resolution over a Restricted Domain: References to Documents. In ACL 2004 Workshop on Reference Resolution and its Applications, 2004, pp. 71–78.

[39] Harabagiu, S. M.—Maiorano, S.: Three Ways to Customize Reference Resolution. Proceedings of International Symposium on Reference Resolution for Natural Language Processing 2002.

[40] Nguyen, A.—Wobcke, W.: An Agent-Based Approach to Dialogue Management in Personal Assistants. In Proceedings of the 10th International Conference on Intelligent User Interfaces (IUI '05), New York, NY, USA, ACM 2005, pp. 137–144.

[41] Rao, A. S.—Georgeff, M. P.: BDI Agents: from Theory to Practice. In ProceedAings of The First International Conference on Multi-Agent Systems (ICMAS-95), pp. 312–319.

[42] Guzzoni, D.—Baur, C.: Modeling Human-Agent Interaction With Active Ontologies.

[43] Laclaviik, M.—Gatial, E.—Balogh, Z.—Habala, O.—Nguyen, G.—Hluchý, L.: Experience Management Based on Text Notes (EMBET). In: Proc. of eChallenges 2005 Conference, 19–21 October 2005, Ljubljana, Slovenia, Innovation and the Knowledge Economy, Volume 2, Part 1: Issues, Applications, Case Studies; Edited by Paul Cunnigham and Miriam Cunnigham; IOS Press, pp. 261–268. ISSN 1574-1230, ISBN 1-58603-563-0.

[44] Bellifemine, F.—Rimassa, G.: Developing Multi-Agent Systems With a Pacompliant Agent Framework. Softw. Pract. Exper., Vol. 31, 2001, pp. 103–128.

[45] Agent-supported workflow in public administration project. Available on: `http://www.aswad-project.org/index.html`, 2002.

[46] Krmer, N. C.—Tietz, B.—Bente, G.: Effects of Embodied Interface Agents and Their Gestural Activity. Proceedings of 4th International Working Conference on Intelligent Virtual Agents 2003.

[47] Purver, M.—Niekrasz, J.—Peters, S.: Ontology-Based Multi-Party Meeting Understanding. 2005.

[48] Bai, Q.—Zhang, M.: Agent Coordination Through Ontology Management. Artificial Intelligence and Applications 2004.

[49] Quesada, J. F.—Garcia, F.—Sena, E.—Bernal, J. A.—Arnores, G.—Julietta, G. D. I.: Dialogue Management in a Home Machine Environment: Linguistic Components over an Agent Architecture. 2001.

[50] Yates, A.: A Reliable Natural Language Interface to Household Appliances. In Proceedings of the 8th international Conference on Intelligent User Interfaces, ACM Press 2003, pp. 189–196.

[51] Paraiso, E. C.—Tacla, C.: Using Embodied Conversational Assistants to Interface Users With Multi-Agent Based Cscw Applications: The Webanima Agent. Vol. 15, 2009, No. 9, pp. 1991–2010.

Emerson Cabrera Paraiso is an Associate Professor in the Postgraduate Program on Informatics at Pontifical Catholic University of Paraná in Brazil. His research interests include natural language processing, text mining, information retrieval and ontologies. He received a Ph. D. in Computer Science from Universit de Technologie de Compiègne (France).

Andreia Malucelli received her Ph. D. in Electrical and Computer Engineering from the Faculty of Engineering, University of Porto (FEUP), Portugal, and her Master in Electrical Engineering from the Federal Technological University of Paraná (UTFPR). She is currently a Professor at the Pontifical Catholic University of Paraná (PUCPR). Her research interests comprise: software engineering, organizational learning, ontologies, multi-agent systems and information systems in healthcare.