An Intelligent Speech Interface for Personal Assistants Applied to Knowledge Management

Emerson Cabrera Paraiso and Jean-Paul A. Barthès
Laboratoire Heudiasyc - Université de Technologie de Compiègne
BP 20.529; 60205 – Compiègne – France; Tel.: +33 3 44 23 44 66; FAX: +33 3 44 23 44 77
Email: {eparaiso, barthes}@hds.utc.fr

Abstract: This paper describes the design of an intelligent speech interface for Personal Assistants applied to knowledge management. We believe that the use of speech facilitates the interaction between human and agent, since the user may speak with the agent using her own terms, increasing the quality of the assistance. One of the difficulties is to handle spoken natural language, understanding its actual context. To do so, we restrict the exchanges to the following Directives Speech Act classes: inform, request, or answer. In addition, for effective knowledge handling, we are using a set of task and domain ontologies, separating domain and task models for reasoning. We present such an architecture in a multi-agent system and apply it to knowledge management. From the user's point of view, the system is a *Service Centre* and the personal assistant is its *Service Provider*. As a result of this conversational speech interface, we expect an increase in the quality of assistance and a reduction of the user's cognitive load.

An Intelligent Speech Interface for Personal Assistants Applied to Knowledge Management

Emerson Cabrera Paraiso and Jean-Paul A. Barthès
Laboratoire Heudiasyc - Université de Technologie de Compiègne
BP 20.529; 60205 – Compiègne – France; Tel.: +33 3 44 23 44 66; FAX: +33 3 44 23 44 77
Email: {eparaiso, barthes}@hds.utc.fr

Abstract: This paper describes the design of an intelligent speech interface for Personal Assistants applied to knowledge management. We believe that the use of speech facilitates the interaction between human and agent, since the user may speak with the agent using her own terms, increasing the quality of the assistance. One of the difficulties is to handle spoken natural language, understanding its actual context. To do so, we restrict the exchanges to the following Directives Speech Act classes: inform, request, or answer. In addition, for effective knowledge handling, we are using a set of task and domain ontologies, separating domain and task models for reasoning. We present such an architecture in a multi-agent system and apply it to knowledge management. From the user's point of view, the system is a *Service Centre* and the personal assistant is its *Service Provider*. As a result of this conversational speech interface, we expect an increase in the quality of assistance and a reduction of the user's cognitive load.

Keywords: Intelligent speech interface, dialogue systems, personal assistants, knowledge management.

1. Introduction

Personal Assistants (PA) are agents that help human users (often referenced as *masters*) do their daily work. The particular skills of a PA are devoted to understanding its master and presenting the information intelligently and in a timely manner. The main goal of such an agent is to reduce the user's cognitive overload. In such a context, the agent displays autonomy while working with the user cooperatively as shown by Maes [28]. Interfacing humans to computer systems using a PA may improve the quality of interaction, especially when the user is involved in several simultaneous tasks. According to Maes, a PA can 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 general, a PA belongs to a community of agents (artificial and human agents). Often, there are two types of artificial agents: *Service Agents* that provide a particular type of services corresponding to specific skills, and *Interface Agents* responsible for interfacing the user with the system [28].

One can evaluate the efficiency of the global system by measuring its quality of assistance: the higher the quality of assistance, the higher the efficiency. In an agent-based system as in any software application, an appropriate user interface is crucial. For many types of applications, the cooperation between user and agent requires

interaction through a user-friendly interface. Traditionally, developers use graphical-oriented interfaces, which implies the use of menus, sub-menus, dialogue boxes, and so on. Often this approach is inappropriate or not very appealing, leading to an assistance of poor quality. To increase the quality of assistance, we are developing conversational voice interfaces ([32] and [33]). Conversational interfaces as defined by Kölzer [25], let users state what they want in their own terms, just as they would do when speaking to another person. Of course the interaction becomes more complex, but the complexity is handled by the system. Conversational interfaces let the user concentrate on her main activity and, once in a while, exchange spoken words with the PA.

Obviously, this approach does not fit all situations. The combined use of PAs and conversational interfaces is more suitable for applications in which:

a) the domain is limited and well defined, as for instance, knowledge management projects or learning environments;
b) user's actions are complex and require previous knowledge of the domain;
c) the user needs to be guided to execute a task;
d) a traditional graphical user interface may be too cumbersome, due to too many menu and sub-menus options, dialogue boxes and so on;
e) the user needs to memorize past actions or navigation steps;
f) a display is not available or the user's hands are busy.

The design and implementation of such components is a difficult task that involves many different modules: dialogue controllers, natural language parsers, speech recognizers and synthesizers, knowledge manipulators, to list but a few. Knowledge handling is a very good example of such complexity. The way knowledge is spread and represented is far from ideal from a computational point of view. Electronic and physical documents, relational databases, tacit knowledge are examples of sources of knowledge in an interconnected system. To be able to offer a real and effective system, specialized agents must be coupled to the different sources of knowledge in order to share information. To do so, researchers in the domain of multi-agent systems are using ontologies ([11], [34] and [43]) for representing and sharing information among agents. Ontologies provide a structured representation of domain knowledge for applications. They may also be used to better understand the users' intentions and to control interaction [30].

In this paper we propose a system that takes into account most of these elements, and present a conversational speech interface for PAs designed specially for professional activities. We are applying our approach to a knowledge management (KM) multi-agent system (MAS) used in the context of research and development projects, as explained by Tacla and Barthès in [43]. The MAS has been developed to support cooperative projects, where each participant shares documents, exchanges information, and contributes to building a distributed organizational memory. The aim of the research is to design a computer environment in which project members can make their knowledge explicit and share their experience without excessive extra-work. To this purpose, each user is given a PA and may speak with it in English in order to control it or to ask it to perform tasks, like retrieving a document from a Lotus Notes® database or looking for knowledge in the organizational memory. The user and her PA use practical dialogues—which means that they are pursuing specific goals or tasks cooperatively—as defined by Allen et al [2]. The dialogue system is task-oriented. Tasks range from simple tasks like "locate a document" to more complex tasks that must be decomposed into subtasks.

As the result of this approach we expect:

- to improve the quality of assistance; and
- to reduce the user's cognitive load.

The rest of the paper is organized as follows. First, we state the main assumptions we made for developing our system. Then, we describe our speech interface and how it works. Then, we present how the interface has been implemented into a PA. After that, we describe briefly how the system is being used for KM. Finally, we mention some related work, we offer a conclusion and indicate some perspectives.

2. Main assumptions

To develop our system we had to make some assumptions related to the behavior of the PA and to the constraints coming from the speech technology.

2.1. Main operational assumptions

The main assumptions related to the PA are the following:

- the PA works concurrently with all the user's activities; the user is doing her job and, once in a while, needs to interact with her PA;
- the PA may ask questions or start a dialogue if a previous user command cannot be executed, or was misunderstood, or else if the agent needs additional information to solve a problem;
- the PA may alert the user when an event has occurred (e.g. incoming electronic message, or incoming response to a search request).

Some assumptions concern the interaction:

- the nature of the application leads us to a master-slave relationship where the user commands her PA;
- the user makes statements that may be questions or declarative sentences;
- the user may change the context of the conversation after a few utterances, introducing a break in the chain of discourse;
- emotion and affective states are considered less important since we are working with daily professional activities.

2.2. Constraints from speech recognition

The speech recognition technology has advanced quickly during the last decade and is now used in commercial projects (see Kotelly [26] for further details). Many applications already have voice capabilities. Speech recognizers can handle simple inquiries about bank balance, movie schedules, and phone call transfers. However, if a spoken interface may appear as a definite improvement, it is important to realize that speech by itself does not automatically produce a user-friendly interface. Simply replacing menu selections by predefined spoken phrases may aggravate the problem, since one is limited to a list of unknown predefined commands. Most commercial projects use the standard VoiceXML (Voice Extensible Markup Language). VoiceXML is an XML-based markup language for distributed voice applications allowing developers to create web-based voice applications that users can access by telephone or other pervasive devices [41]. VoiceXML however, is a limited tool and real open conversation treatment is not covered.

Speech recognition systems can be characterized by many dimensions: speaking mode (isolated word to continuous speech), enrollment (speaker-dependent to speaker-independent), vocabulary (small to large) and many others [7]. Recognition is generally more difficult when vocabularies are large or have many similar-sounding words. A commonly used recognition paradigm is known as the Hidden Markov Models (HMM). An HMM is a doubly stochastic model, in which the generation of the underlying phoneme string and the frame-by-frame, surface acoustic realizations are both represented probabilistically as Markov processes.

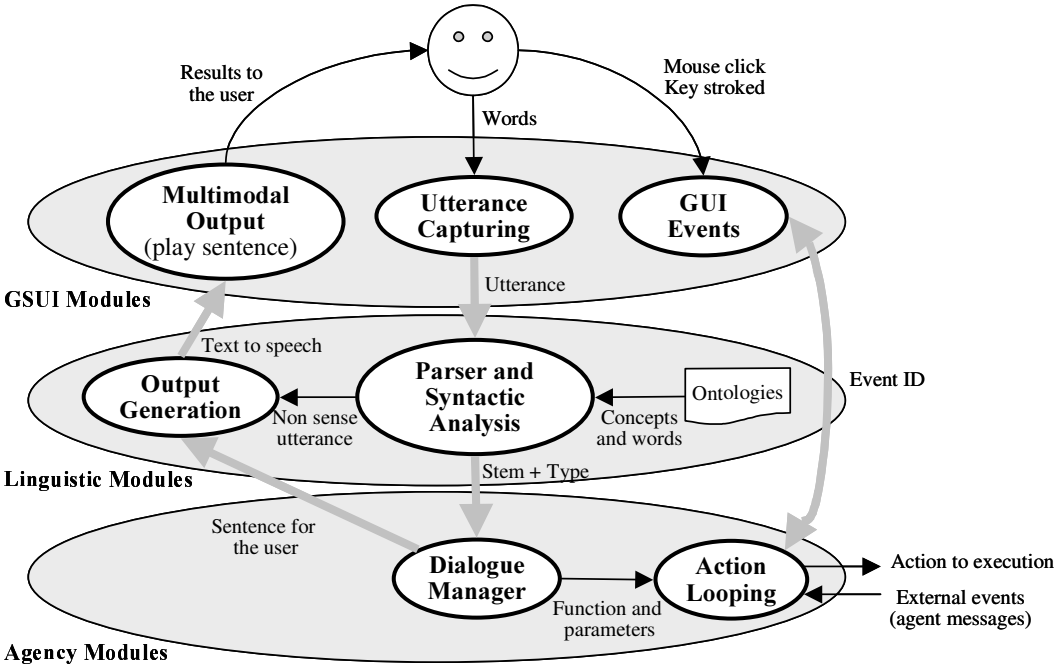


Fig. 1. Interface diagram.

Speech recognition is an extremely complex process, quite error prone, and cannot be solved today without a great deal of knowledge about what the utterances are likely to be. Thus, the following constraints must be taken into account:

- the results from an automatic speech recognition engine may be quite far from what the user actually said; the differences may be lexically and syntactically significant;
- people address computers differently than they address other humans, trying to adapt to what they perceive as limitations of the machine [22].

Regarding the last point, we believe that if we could make the computer interact with people in a more realistic way, we could reduce the cognitive load.

Since using isolated spoken commands does not seem to be efficient, we propose to add a mechanism to control a real dialogue between the user and the PA. Of course, adding voice to a dialogue increases its complexity. Indeed, the development of a spoken dialogue system requires the integration of several components of the spoken language technology, like speech recognition, natural language processing, dialogue modeling or speech synthesis. However, the result is an interaction that approaches human performance.

The next section proposes an architecture for an intelligent speech interface, the design of which results from the considerations that were just developed.

3. Architecture of the intelligent speech interface

The global architecture is shown in Fig. 1. It has three parts: (i) graphical and speech user interface (GSUI) modules; (ii) linguistic modules; and (iii) agency modules. *GSUI modules* produce outputs or collect the user's inputs, like capturing voice and handling GUI events. *Linguistic modules* are responsible for lexical and syntactical analysis and context verification. *Agency modules* are directly connected to the agent kernel, which can "intelligently" manage the dialogue and the interface through the use of ontologies. The

architecture is detailed in the following paragraphs.

3.1. GSUI modules

Although talking is a privileged mode, the agent interface is multimodal. Hence, graphical events such as mouse clicks or key strokes may occur at the same time. In particular, the main window may contain buttons or graphical elements displaying general information like trees of documents. The corresponding events are directly sent for execution.

Whenever the user says something, this is known as an utterance. An utterance is defined by Kemble as any stream of speech between two periods of silence [23]. 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 an automatic speech recognition engine that returns a list of words. The *Utterance Capturing* module concatenates the resulting list of words. To do so, it looks for a period of silence to determine when the end of an utterance is reached. A process running independently analyzes each utterance in parallel with the Utterance *Capturing* process.

Due to local noise interference or bad pronunciation, the utterance may be lexically and/or syntactically different from the words actually said. Initially, we are using the utterance as it is, extracting a list of known disfluencies. However, we intend to later study a mechanism to enrich and/or modify the utterance.

When using verb phrases, the verb is directed linked to an action list. In the sentence "I would like to send an email" the verb *send* may fire the action of opening an email client for *sending* a message. Syntactic markers such as "the" are less important.

3.2. Linguistic modules

Like in most dialogue systems, we process each utterance sequentially.

Table 1. **Example of utterances**

User's utterances	Action fired by the PA
"Please, search Agents.doc in the database."	Send request to *Document Manager Service Agent*
"Retrieve all definitions for Bluetooth."	Search knowledge item in all individual memories
"Book a flight ticket to Paris."	Out of context: not executed

Table 1 shows some possible utterances and the specific actions fired thereafter. Each utterance is processed as described in the next section. The nature of the application, i.e., knowledge management, allows us to restrict the space of dialogues to those containing only Directives Speech Acts statements (e.g., inform, request, or answer) ([1] and [40]). Currently, we are not taking into account emotions or affective states. In recent works, however, some researchers have been studying such effects (see [19] and [29] for further details).

3.2.1. Parser and syntactic analysis

The process of interpreting an utterance, shown in Fig. 2, 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 works bottom-up. It replaces each utterance stem with its syntactic category (verb, noun, adverb, etc) with the help of a lexicon file. The parser is guided by the *Grammar Verification* module that tries to match each grammar rule against the utterance. It also identifies and isolates eventual constituents presented in the utterance, thanks to the mechanism used in the Link Parser [42].

As reported by Jurafsky and Martin [22], there are a number of ways utterances differ from written English sentences. Spoken sentences have many more pronouns than written sentences. They are shorter, consisting

of fragments or phrases. Another peculiarity is the presence of breaks, e.g. hesitations, "uh," or "um." Thus, we designed grammar rules to handle such cases. Although our interface uses a list of specialized grammars, the latter are not restrictive. Popescu et al. in their natural language interface for databases [35], restricted the grammar to *wh-questions* (what, where, etc), which would be too restrictive in our context.

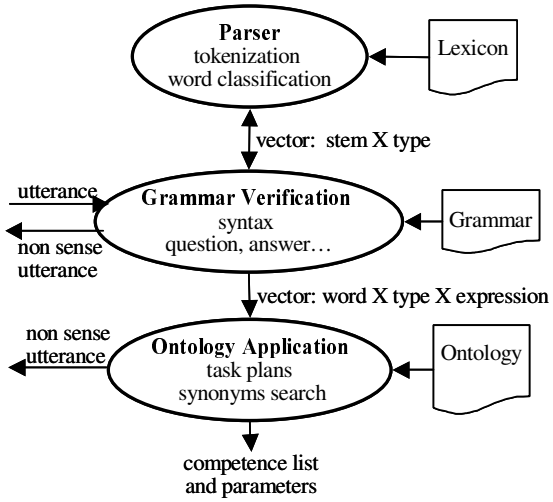


Fig. 2. Linguistic treatment.

We limited the space of dialogue utterances however to Directives Speech Act classes—inform, request, or answer—since they define the type of expected utterances in a master-slave relationship. The grammar rules were divided in order to classify an utterance into one of three categories: order, interrogation or answer. In our application, a typical utterance could be: "*Give me the list of all project participants.*" According to our taxonomy this is an order utterance and can be processed by the grammar rules.

Once an utterance is classified, the domain knowledge is taken into account with the help of WordNet [12] and of a domain ontology. The treatment is made by the *Ontology Application* module and is described later in this paper.

We are using WordNet to improve the linguistic treatment. WordNet is a lexical reference system whose design is inspired by current psycholinguistic theories of human lexical memory [12]. English nouns, verbs, and adjectives are organized into synonym sets, each representing one underlying lexical concept. WordNet also defines a large number of domain-independent lexical relationships that can hold between synsets, e.g., *is-a*, *part-of*, or *member-of*. We are using the set of synonyms to help the linguistic modules match tokens with ontological concepts.

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 occurrence of nonsensical utterances is rare, since our system tries to act with minimal information, but nevertheless may occur.

3.2.2. Output generation

The generation mechanism is limited to creating utterances from parameters listed in a dialogue model. Such utterances may vary from pre-formatted questions to warning messages.

3.3. Agency modules

The PA is composed of three main blocks: the *Speech Interface*, the *Agency modules* and the *Kernel* (Fig. 5). The speech interface is directly related to a second block, the *Assistancy* module, responsible for controlling the dialogue and assistance. The *Dialogue Manager* links the *Speech Interface* and the *Assistancy* module. The mixed-initiative and task-oriented dialogue mechanism is coordinated by the *Dialogue Manager* module. The *Dialogue Manager* is capable of choosing a dialogue model appropriate to a session that has just begun. 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*. However, if the initial utterance lacks crucial information, e.g., a task parameter, it starts sub-tasks to complete the

action list, asking for additional information from the user. To do that, it uses an action library. Once the action list is complete, the PA executes it, with support from other agents.

The *Action Looping* handles GUI events and also receives calls from the *Dialogue Manager*. After choosing a function to be executed, the *Dialogue Manager* sends it the chosen function and a list of parameters. The function will be executed locally or remotely depending on the type of required task. *Action Looping* is also responsible for merging all modalities (e.g. button click and speech).

4. Domain knowledge and task representation

In this section we present our strategy for handling knowledge and modeling tasks. Although knowledge and tasks are modeled by ontologies, it is important to highlight that domain knowledge and task models are handled separately.

4.1. Domain knowledge

Knowledge handling is crucial for the effectiveness of our architecture. Domain knowledge is used here to further process the user's statements and for reasoning. For that, we use 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 [16]. 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 [24], to representing user profiles in a recommender system [30].

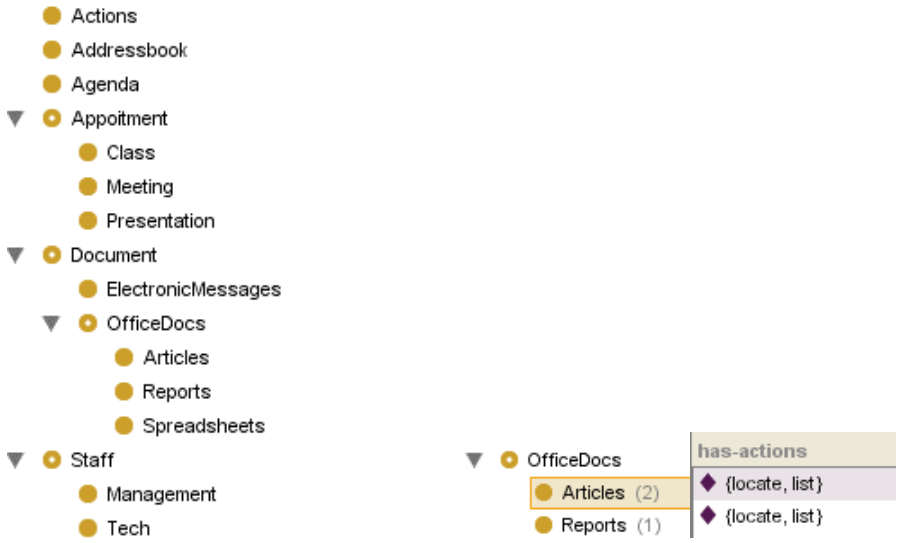
In a recent paper, Milward and Beveride [31] describe how scripted dialogue systems are moving to a new generation of practical systems based on domain knowledge and task descriptions. Eriksson in [11] addresses the issues of designing ontologies for dialogue interaction and information extraction. In her work, she explores the requirements of an ontology specially designed for dialogue systems. In [11], Eriksson states that ontologies provide a common vocabulary that can be used to state facts and formulate questions about the domain.

Another example of ontology application in the domain of dialogue support is presented in [36], where Purver et al. propose an ontology-based discourse understanding mechanism for a persistent meeting assistant. In their approach, a central ontology of multimodal discourse is used to describe all communicative actions performed at physical and conceptual levels.

Ontologies may be used during the parsing process. Dzikovska et al. [8] 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.

In our PA, ontologies play two main roles: (i) they help interpret the context of messages sent by others agents or by the user; and (ii) they keep a computational representation of knowledge useful at inference time.

We are using a set of task and domain ontologies, separating domain and task models for reasoning. As suggested by Allen, 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.



a) Excerpt of the ontology.

b) Actions list for *Articles*.

c) Actions instances.

d) A concept and its attributes.

Fig. 3. An excerpt of the ontology *Project*.

4.1.1. Semantic interpretation

In the context of an open conversation, the problem of understanding is complex. Semantic processing involves extracting meanings from recognized word strings and inferring the user's dialogue acts or information goals based on the recognized semantic concepts [17]. However, one does

not require a full understanding of the user's utterances to act in the right direction as stated by Popescu et al [35]. Schank and Rieger [39] once wrote: "it is not necessary to understand in order to act." The design of our spoken dialogue system follows this principle.

The approach to 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 *Ontology Application* module is interested in finding the list of tokens 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.

We believe that this approach is ideal for applications where the domain is well known and restricted. In contrast, statistical models as proposed by He and Young [17] bring the analysis to the parser level, leaving useful domain information out of the process.

Designing an ontology involves several decisions on various levels of details, which will not be discussed here. In this paper, the ontologies are simple and short enough to understand the semantic interpretation mechanism we are presenting. The concepts and their properties are organized so as 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. 3a (we used Protégé [15] for an easier 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. Each concept may have some attributes (Fig. 3d). Note that a set of actions (ex: read, list, erase, shown in Fig. 3c) may be applied to each concept, as shown in Fig. 3b.

First, the parser verifies the context of the input. It verifies that it is a question related to the domain. To do that, it uses the domain ontology and the lexicon. The lexicon contains thousands of words extracted from WordNet enriched with the list of all concepts and attributes of the ontology.

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. 3c). Note that it uses a list of synonyms from WordNet (e.g. "list," "enumerate" or "name" are synonyms in this sense). It uses others mechanisms like Trigrams to evaluate word similarity [18]. It also finds that *articles* is an object and *agents* is its property. At this point, we have an action list with its parameters. Next, the *Dialogue Manager* module takes control of the dialogue.

Let us take another example. Given the user utterance:

USER: *I want to modify the article about agents.*

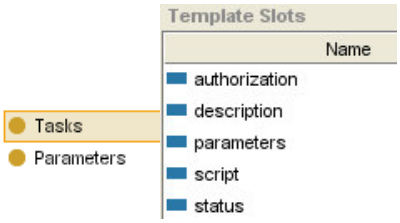
In this case the system figures out that the action *modify* is not in the list of allowed actions. It will enumerate the possible actions applicable to *articles* and ask for a new input.

4.2. Task representation

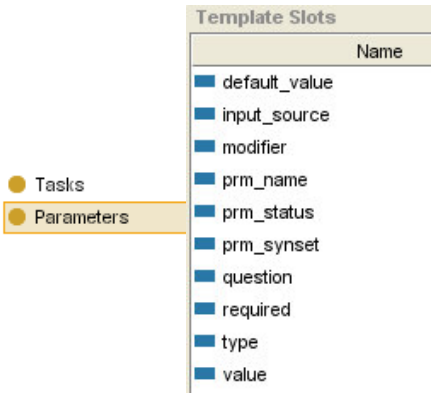
Tasks in our system are represented as shown in Fig. 4. A task has a set of parameters that are filled during a dialogue session. The *Dialogue Manager* module will push a task onto the stack of tasks when an utterance related to the task is given. Many tasks may be handled simultaneously (even tasks of the same type).

Let us take an example. The user says:

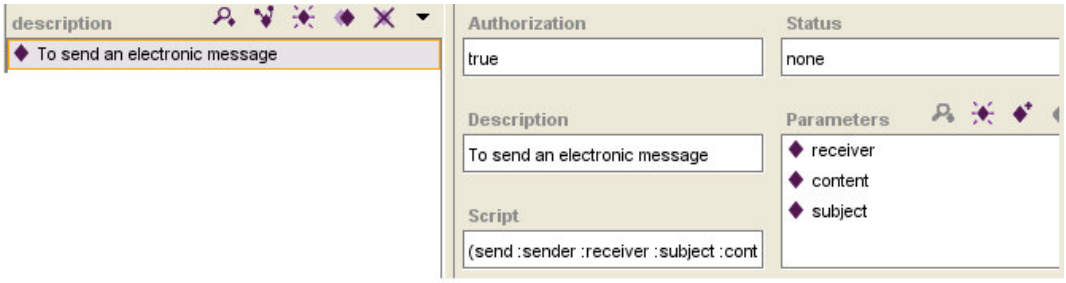
USER: *I need to send an email to Mike and to Palmer*.



a) Task parameters.



b) Parameters' fields.



c) A task instance.

Fig. 4. 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 Fig. 3). The task *To Send an Electronic Message* (Fig. 4c) has some parameters to be filled (Fig. 4a) before the agent is able to execute it. One of the parameters is 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. 4b). 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. Normally, the task execution is performed by other agents, as described later in this paper.

In the next section, we present the structure of our PA.

5. Structure of the personal assistant

Our Personal Assistant (PA) is a rather complex system, shown in Fig. 5. The *User Interface* module has already been described in this paper. For the problem under consideration and because of the distributed nature of knowledge, we naturally think of agents as a possible paradigm as discussed by Barthès [3]. 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. Also, such agents run independently of any particular problem solving task. A large number of papers and books have been published about agents and the reader is referred to Jennings et al [20] for a quick review of the domain.

Our agent is built around three main blocks: the user interface, described previously, an *Assistancy* module and a fixed body, called the *Agent Kernel*.

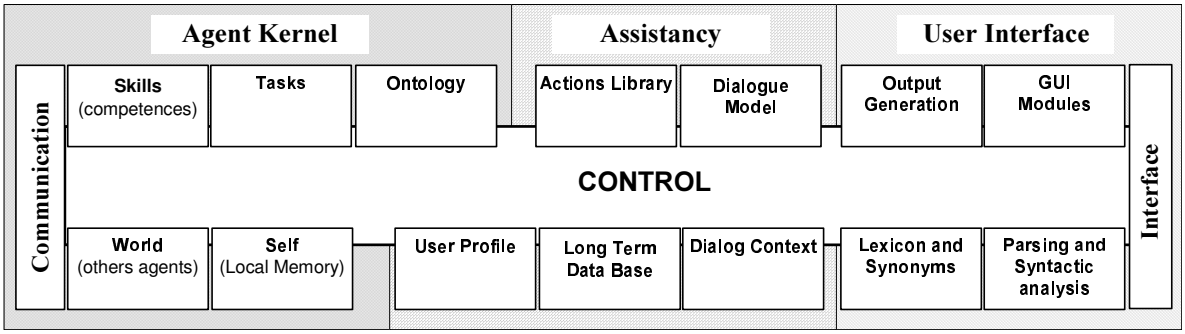


Fig. 5. PA structure.

5.1. Assistancy

A personal assistant needs a mechanism for controlling the dialogue and for keeping track of the conversation. The *Dialogue Manager* controls the dialogue, receiving and sending the utterances from and to the user. To do so, it uses a library of actions to follow during a dialogue session. It also uses a set of dialogue models for executing each action. The context of the dialogue is kept by storing the syntactic tree corresponding to each utterance for each dialogue session that was started.

The dialogue strategy is cooperative, which means that the PA takes the user goals and helps him achieve them. This strategy is implemented by the *Dialogue Manager* [5] to:

- evaluate the situation and to propose explanations, help or pertinent arguments for making the right choice;
- search the best path in the space of possibilities;
- follow the user until the solution is found.

The PA is also capable of learning the user's activities. To learn them, it uses machine-learning algorithms. In [27], Liu et al. propose an adaptive user interface based on personalized learning. We use a centroid-based method. This method called ELA (Entropy-based Learning) works dynamically, using the incremental learning principle. The reader is referred to [10] for more information.

5.2. Agent kernel

In our system all agents are cloned from a generic agent, first proposed by Ramos in [37]. The generic agent contains all the basic structure that allows an agent to exist. Let us describe each component briefly:

- *Communication*: responsible for sending and receiving messages to or from the environment;
- *World*: contains an internal representation of other agents and of the environment;
- *Skills*: contains the set of services performed by an agent. A particular skill can be expressed as procedures or as rules. If the skill is complex, then a plan for executing it can be created, and the derived tasks are spawned and broadcast on the network;
- *Tasks*: is a representation of what tasks are currently active in the system;
- *Ontology*: specification of a vocabulary; or a taxonomy (domain), used to interpret messages;
- *Self*: stores the memory, goals and skill descriptions;
- *Control*: controls the different modules.

The agent runs on a platform specially designed to implement multi-agent systems, called OMAS (Open Multi-Agents System) [4]. In OMAS, agents have a private Agent Communication Language (ACL) similar to KQML [13]. The platform is FIPA compliant [14], which allows external interaction with others platforms if needed.

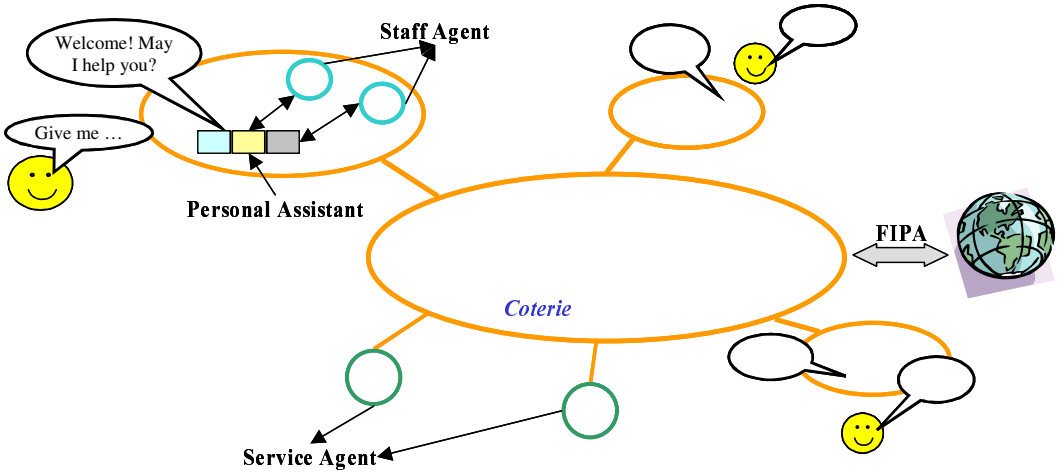


Fig. 6. MAS architecture.

A variety of platforms and agent architectures have been proposed. A good reference is the work of d'Inverno and Luck [9] that presents a formal approach to dealing with agents and agent systems.

6. MAS for knowledge management

We are embedding our speech interface into a PA that is part of a knowledge management multi-agent system, shown in Fig. 6. In this architecture, agents are totally independent but belong to a cluster called a "coterie." Communication occurs thanks to several protocols (basic and Contract-Net), specified at the message level. Agents can dynamically enter or leave the coterie.

A coterie contains two types of agents: *Service Agents* that provide a particular type of service corresponding to specific skills, and PA*s*. We assign a PA to each human participant. It is in charge of all information exchanges among participants. A PA can use some service agents locally. They constitute its *staff* (Fig. 7). The staff agents may also delegate sub-tasks to other agents in order to accomplish a complex task.

Our approach is bottom-up and aims at recording the user's behavior automatically whenever possible, building a library of cases. It comprises several steps: capturing and representing an action or operation, augmenting an operation representation, clustering operations, indexing and classifying the results. All this is done locally (by the PA and its staff) and the results are stored in a distributed memory. Actions or operations are mainly related to communications (e.g. sending emails), or documents (e.g. searching for documents). They can be activated from the user interface provided by the PA. The PA builds a representation of each operation.

The approach presents two advantages: (i) it is easy to implement; and (ii) the information is qualified according to the needs of a particular specialist. Thus, in this approach, PAs are important agents rather than *Service Agents*. The net result is a distributed knowledge system in which the information has been organized locally as a function of the particular interests of a given user.

6.1. Speech interface for KM systems

The need for a speech interface is clear when we study the complexity of a user's action in a KM system. The central point here is to decrease her cognitive overload.

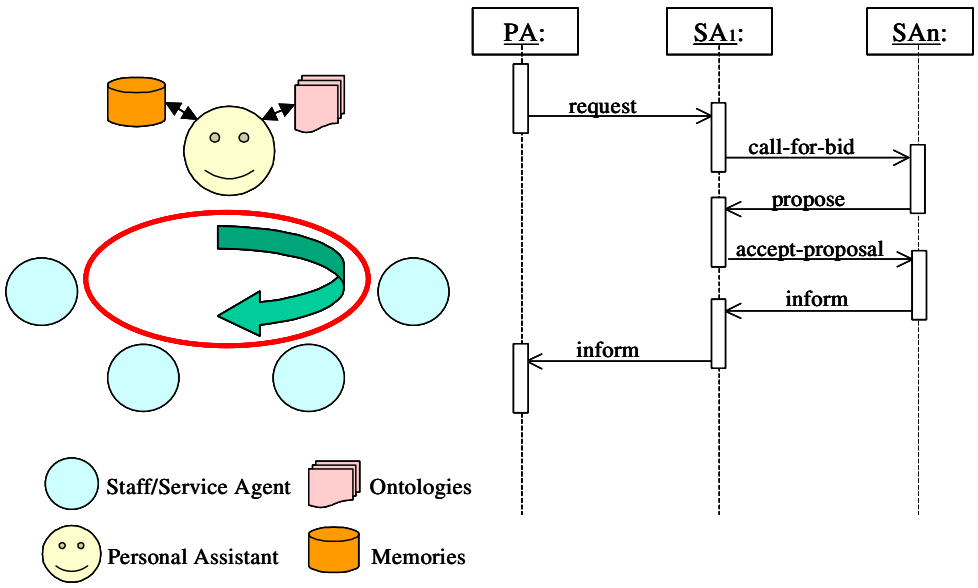


Fig. 7. PA delegating to other agents the execution of tasks.

A KM system is a very specialized piece of software and it should not waste the user's time. Knowledge management involves experienced and less experienced users. Since one of the sources of information for the system is the user input, whenever the user describes her intentions with greater details, the system will produce better results. In addition, the user should obtain the same results regardless of her experience in the domain. Our main strategy is to simplify the interaction between the user and her PA in order to reduce extra work.

From the user point of view, the whole system is a *Service Centre* and the PA is its *Service Provider*. All negotiation is made through the PA.

6.2. A dialogue excerpt

To give an example of our approach let us introduce the situation shown in Table 2. Imagine a group of engineers that cooperatively develop a new software component, working with their PA. To better understand the system capabilities, we defined a short set of competences for the PA: list all project participants, give information about a member, send messages or open the email account. At the beginning, the PA presents itself and produces a welcoming message. This first statement shows that the agent selected a directive strategy for the dialogue system. In fact, after the opening message, it switches to a cooperative strategy until the end of the session.

The user starts by clarifying what the PA can do (*input 2*). Line *4*, the user orders the PA to do an action that it cannot do. The user gives another input (*input 6*).

Later, the user needs to know who is in charge of updating the project database (*input 10*). Initially, the PA cannot produce any action since it does not have a competence associated with the first request. The PA may give information about a member, but only if this member is identified, i.e. named. As it is not the case, the PA warns the user. The user then orders the PA to list all project members (*input 12*). The list shows Mike as the database manager. The user then breaks the conversation thread and asks for an incompatible action (*input 16*). Note, that the PA may now list all attributes for the action *send* to be applicable, since *send* is an action described in the ontologies.

Table 2: **An excerpt of a dialogue**

1	PA: Welcome to the Voice Personal Assistant.
2	USR: What can you do?
3	PA: I can: list all project participants, or give information about a project member, or send electronic messages, or open email account. What do you want to do?
4	USR: Give a list of all cities near Paris.
5	PA: I cannot do what you want.
6	USR: Give the address of Barthes.
7	PA: Jean-Paul Barthes, Professor at UTC-France, branch: 1234, email: barthes@utc.fr
8	USR: Open my email account.
9	PA: Email client loaded.
10	USR: Who is responsible for updating the project database?
11	PA: I cannot do what you want.
12	USR: Give me a list of all participants.
13	PA: Members are: Leader: Paul, Project Database Manager: Mike, Engineer: Mary.
14	USR: Give me all information about Mike.
15	PA: Mike Palmer, Lotus Database manager, branch: 2328, email: mike.palmer@utc.fr
16	USR: I want to send a book to Mike.
17	PA: Sorry, I cannot do that. You may send electronic messages.
18	USR: Please send an email to Mike.
19	PA: What is the subject?

This excerpt of conversation illustrates the way users interact with the system.

6.3. Implementation details

Our platform runs on a Microsoft Windows™ environment, using the default automatic speech recognition and text to speech engines. No other special software or component is required.

Ontologies are XML files. The lexicon files are mainly composed of thousands of words extracted from WordNet.

The window interface runs in a minimized mode, which means that it does not need to get the focus to be active. This was defined intentionally because the user is working with other programs, like word processors, or CAD/CAM applications. The interface (main window snapshot shown in Fig. 8), however, remains "listening" and takes into account whatever the user is saying.

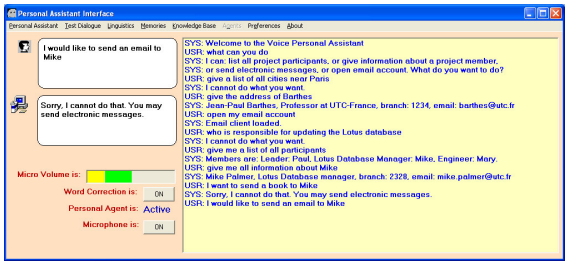


Fig. 8. Snapshot of the main window.

7. Improvements in the quality of assistance

Using a speech interface in a PA improves the quality of assistance in some specific situations, as in KM context. First of all, it makes the system operation faster and easier, since the user does not need to be an expert in KM. The PA handles all the complexity. We think that inexperienced users will easily operate the PA's interface compared to interfaces using traditional approaches, with only GUI elements, menus and sub-menus.

Another expected result is that the user can explore all the system functionalities. Instead of looking for a command or a specific feature among sometimes confusing menus and hidden dialogue boxes, the user just says something and, if the functionality is supported, gets some result. Our wish is that the last action the user will do is to open a help manual, because it is simpler to try an utterance or to order the PA to list its competences in a cooperative fashion.

Since the application is a PA, an essential feature of the user interface was respected: predictability. It was an assumption we made 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 in an ontology.

In some applications, speech interfaces are the only way to interact with the system, especially in applications in which the user has her hands busy or when the user is physical impaired. In such cases an application containing a PA with speech interface is well adapted.

All those conclusions came from first observations and an analysis of further results is under way and will be discussed in a future paper.

8. Related work

Although speech interfaces and dialogue systems are used in several projects, our application to personal assistants and knowledge management is original. However, our contribution may also arise from some design decisions we took.

Knowledge handling is crucial for the effectiveness of this interface. We are using a set of task and domain ontologies, separating out domain and task models for reasoning. As suggested by Allen, this observation is interesting for domains where task reasoning is crucial.

Johnson et al [21], in their animated pedagogical assistant agent, *Adele*, prefer to use a traditional interface, using buttons to invoke the agent assistance. *Adele* helps a student to learn through clinical problems. Despite the complexity of the domain, the user's interface is control-oriented and suitable for conversational speech. They report that users experience difficulty in some situations, due to a lack of coordination. We think we could handle this problem with our speech interface, since the PA coordinates the user's actions during the dialogue.

In [36], Purver et al. 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).

A question-answering system for knowledge management applications was proposed by Cheng and mentioned in [6]. The main objective was to develop a framework for an NPL based system for extracting useful information from semi-structured pieces of text. The system was prototyped but does not support spoken dialogue. The question-answering interface limits the user's interventions to questions, and does not support direct orders.

A growing number of researchers, like Quesada et al or Yates et al, are working with

interfaces to household appliances ([38] and [45]). They are proposing speech interfaces in their projects, but they do not consider a very important issue, namely intelligent support. None of them uses an intelligent and specialized mechanism to support the user, as we did using the assistant agent approach. *A Personal Assistant 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.

9. Conclusions and future work

Although speech interfaces and dialogue systems are used in several projects, our application to PAs and knowledge management is original. In this paper, we presented an architecture for an intelligent speech interface for a PA in specialized domains. We described it and implemented it in a PA that works in a multi-agent system for managing knowledge. From the user's point of view, the system is a *Service Centre* and the PA is its *Service Provider*.

Such an architecture is suitable for PAs, particularly in specialized domains like KM. Our main goal is to improve the quality of the assistance. We have listed a number of expected improvements. The system operation becomes easier and faster, leaving more time for main activities. A formal evaluation process will evaluate the results and guide us for future improvements. We also plan to implement a mechanism to improve the analysis of the utterance, enriching it or adapting it.

Another important direction for future work is allowing users to access their assistant by different means, such as telephone or personal digital assistants (PDA). The user's mobility may impose interesting constraints to be tackled by the next versions of the speech interface.

This speech interface may be easily applied to new domains of application like household appliances. Since the domain is very precise and hands free operation is imperative, the speech interface is worth trying.

Finally, we are working on an application prototype of our interface to a module of the TERREGOV project [44]. TERREGOV addresses the issue of interoperability of eGovernment services for local and regional governments. The project integrates the dimensions of technological R&D, pilot applications involvement and socio-economic research in order to offer a European reference for the deployment of interoperable eGovernment services in local governments. In this prototype, users are *civil servants* that give information (advice) to citizens. The aim of the prototype is to accelerate the interaction, increasing its productivity.

Acknowledgments

Emerson Cabrera Paraiso would like to thanks CAPES – Brazil (grant 1306-02-2) and Pontificia Universidade Catolica do Parana - Brazil that supported him in this research.

References

[1] J. L. Austin, How to do things with words, Harvard University Press, Cambridge, MA, 1962.

[2] J. Allen, G. Ferguson and A. Stent, An architecture for more realistic conversational systems, in: *Proceedings of Intelligent User Interfaces (IUI-01)*, Santa Fe, NM, 2001.

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

[4] J.-P. A. Barthès, OMAS v 1.0 Technical Reference, *Technical Report nº 144*, HEUDIASYC UMR 6599, Université de Technologie de Compiègne, 2000.

[5] J. Caelen, Modèles de dialogue pour l'interaction homme-machine, in:

Tutorial of the *16th Conférence Francophone sur l'Interaction Homme-Machine*, Namur, Belgique, 2004.

[6] J. Cheng, B. Kumar and K. Law, A question answering system for project management applications, in: *Advanced Engineering Informatics*, No. 16, Elsevier, 2002, pp. 277-289.

[7] R. A. Cole, J. Mariani, H. Uszkoreit, A. Zaenen and V. Zoe, Survey of the State of the Art in human language technology, Cambridge University Press, 1996.

[8] M. O. Dzikovska, M. D. Swift and J. F. Allen, Integrating linguistic and domain knowledge for spoken dialogue systems in multiple domains, in: *Proceedings of IJCAI-03 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Acapulco, Mexico, August, 2003.

[9] M. d'Inverno and M. Luck, Understanding agent systems. Springer, 2nd edition, 2004.

[10] F. Enembreck and J.-P. A. Barthès, Agents for collaborative filtering, in: *Proceeding of Cooperative Information Agents VII – CIA 2003,* Finland, August 2003, Springer-Verlang, pp. 184-191.

[11] A. Eriksson, Design of ontologies for dialogue interaction and information extraction, in: *Proceedings of International Joint Conference on Artificial Intelligence IJCAI 2003*, Acapulco, Mexico, 2003.

[12] C. Fellbaum, WordNet: An electronic lexical database, MIT Press, Cambridge, MA, 1998.

[13] T. Finin, Y. Labrou and J. Mayfield, KQML as an agent communication language, invited chapter in: Jeff Bradshaw (Ed.), *Software Agents*, MIT Press, Cambridge, 1997.

[14] FIPA Interaction Protocol Library Specification, http://www.fipa.org/specs/fipa00025/PC00025C.html.

[15] J. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubézy, H. Eiksson, N. F. Noy and S. W. Tu, The evolution of Protégé: an environment for knowledge-based systems development, Report available at: http://smi.stanford.edu/pubs/SMI_Abstracts/SMI-2002-0943.html, 2002.

[16] N. Guarino, Formal ontology in information systems, in: *Proceedings of FOIS'98,* IOS Press, Italy, 1998, pp. 3-15.

[17] Y. He and S. Young, Semantic processing using the Hidden Vector State model, in: *Computer Speech and Language*, Vol. 19, No. 1, 2005, pp. 85-106.

[18] J. Hylton, Identifying and merging related bibliographic records, *Master of Engineering thesis*, MIT Department of EECS, June, 1996 (actually completed February, 1996). Also published as LCS Technical Report MIT/LCS/TR-678.

[19] H. Hyötyniemi, J. Hemanus and V. Lantz, Exchanging emotions – SOM approach, in: *Proceedings of* ECAI - *European Conference on Artificial Intelligence*, Valence - Espagne, 2004.

[20] N. R. Jennings, K. Sycara and M. Wooldridge, A Roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems 1* (1998) pp. 7-18.

[21] W. Johnson, E. Shaw, A. Marshall and C. Labore, Evolution of user interaction: the case of agent Adele, in *Proceedings of Intelligent User Interfaces (IUI-03),* ACM Press, 2003, pp.93-100.

[22] D. Jurafsky, and J. Martin, Speech and Language processing: an introduction to natural language processing, Computational Linguistics and Speech Recognition. Prentice Hall, 2000.

[23] K. A. Kemble, An introduction to speech recognition. *Voice Systems Middleware Education* - IBM Corporation, 2001.

[24] J. Kim, M. Spraragen and Y. Gil, An Intelligent Assistant for interactive workflow composition, in: *Proceedings of Intelligent User Interfaces 2004 (IUI-04)*, Madera, Portugal, 2004.

[25] A. Kölzer, Universal dialogue specification for conversational systems, in: *Proceedings of IJCAI 99 – Workshop on Knowledge and Reasoning in Practical Dialogue Systems,* 1999.

[26] B. Kotelly, The art and business of speech recognition, Addison-Wesley, 2003.

[27] J. Liu, K.C. K. Wong and K. K. Hui, Adaptive user interface based on personalized learning, in: *IEEE Intelligent Systems*, Vol. 18, No. 2, 2003, pp. 52-57.

[28] P. Maes, Agents that reduce work and information overload, in: *Communications of ACM*, Vol. 37, No.7 (1994), ACM Press, pp. 30-40.

[29] J. L. Meyer, Reasoning about emotional agents, in: *Proceedings of* ECAI - *European Conference on Artificial Intelligence*, Valence - Espagne, 2004.

[30] S. E. Middleton, D. C. De Roure and N. R. Shadbolt, Capturing knowledge of user preferences: ontologies in recommender systems, in *Proceedings of the International Conference on Knowledge Capture (KCAP'01)*, ACM Press, 2001, pp. 100-107.

[31] D. Milward and M. Beveride, Ontology-based dialogue systems, in: *Proceedings of International Joint Conference on Artificial Intelligence IJCAI 2003,* Acapulco, Mexico, 2003.

[32] E. C. Paraiso and J.-P. A. Barthès, Architecture d'une interface conversationnelle pour les Agents Assistants, in: *Actes de la journée Agents et Langue*, Paris, France, 2004, pp. 83-90.

[33] E. C. Paraiso, J.-P. A. Barthès and C. A. Tacla, A speech architecture for personal assistant in a knowledge management context, in: *Proceedings of* ECAI - *European Conference on Artificial Intelligence*, Valence - Spain, 2004, pp. 971-972.

[34] E. C. Paraiso and J.-P. A. Barthès, Une interface conversationnelle pour les agents assistants appliqués à des activités professionnelles, In: *Proceedings of 16th Conférence Francophone sur l'Interaction Homme-Machine*, Namur, Belgium, 2004, pp. 243-246.

[35] A. M. Popescu, O. Etzioni and H. Kautz, Towards a theory of natural language interfaces to databases, in: *Proceedings of Intelligent User Interfaces (IUI-03),* ACM Press, 2003, pp. 149-157.

[36] M. Purver, J. Niekrasz and S. Peters, Ontology-based multi-party meeting understanding, abstract accepted for the *CHI 2005 workshop The Virtuality Continuum Revisited*, April 2005.

[37] M. P. Ramos, Structuration et évolution conceptuelles d'un agent assistant personnel dans les domaines techniques, *PhD dissertation* (in French), presented at UTC in December 2000.

[38] J. F. Quesada, F. Garcia, E. Sena, J. Bernal and G. Amores, Dialogue managements in a home machine environment: linguistic components over an agent architecture, SEPLN, 2001, pp. 89-98.

[39] R. C. Schank and C. J. Rieger, Inference and the computer understanding of natural language, in: *Artificial Intelligence*, Vol. 5, n. 4, North-Holland Publishing, 1974, pp. 373-412.

[40] J. R. Searle, A taxonomy of illocutionary acts, in: *Proceedings of Language, Mind and Knowledge*, Vol. 7, University of Minnesota Press, 1975, pp. 344-369.

[41] C. Sharma and J. Kunins, Voice XML, Wiley, 2001.

[42] D. Sleator and D. Temperley, Parsing English with a link grammar, in: *Third International Workshop on Parsing Technologies*, 1993.

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

[44] TERREGOV Project Web Site: http://www.terregov.eupm.net/.

[45] A. Yates, O. Etzioni, O. and D. Weld, A reliable natural language interface to household appliances, in: *Proceedings of Intelligent User Interfaces (IUI-03),* ACM Press, 2003, pp. 189-196.