Architecture d'une interface conversationnelle pour les agents assistants personnels

Emerson PARAISO et Jean-Paul BARTHÈS

Laboratoire HEUDIASYC – Université de Technologie de Compiègne
BP 20529 – 60.205 – Compiègne - France
{eparaiso, barthes}@utc.fr

Résumé – Abstract

Cet article décrit la conception d'une interface conversationnelle et d'un système de dialogue en langage naturel pour les agents assistants personnels. Nous estimons que l'utilisation de la parole facilitera l'interaction entre l'homme et la machine, car l'utilisateur peut parler avec l'agent en utilisant ses propres termes, obtenant par là une aide de meilleure qualité. D'ailleurs, les interfaces purement graphiques rendent l'application des agents assistants difficile dans un certain nombre de situations, par exemple celles basées sur l'utilisation du téléphone. La difficulté principale cependant est la manipulation du langage parlé et la compréhension du contexte. Pour cela, nous limitons les échanges à ceux qui contiennent des actes de langage illocutoires : *inform*, *question* ou *anwser*. Nous présentons l'architecture d'un système multi-agents appliquée à la gestion personnelle de l'information. Les tout premiers résultats font espérer une amélioration de la qualité de l'interface et de l'aide à l'utilisateur. Les futurs développements prévoient une étude approfondie de la qualité de l'interaction.

This paper describes the design of a speech and natural language dialog interface for personal assistants. We believe that the use of speech will facilitate the interaction between human and machine, since the user may speak with the agent using her own terms, increasing the quality of the assistance. Moreover, pure GUI interfaces make the utilization of personal assistants difficult in some situations like those involving the use of telephones. The main difficulty however is to handle spoken natural language, understanding its actual context. To do so, we restrict the exchanges to Directives Speech Act classes, i.e., inform, request, or answer. We present a conversational speech interface architecture in a multi-agent system applied to personal information management. As a clear result of this conversational speech interface, we expect an increase in the quality of the interface and of the assistance to the user, which seems to be the case on preliminary results. However, we plan to develop an in-depth evaluation of quality of the interaction to confirm such results.

Mots Clés – Keywords

Interface conversationnelle, système de dialogue, agent assistant, ontologies, langage naturel.

Speech interfaces, dialog systems, assistant agents, ontology, natural language.

1 Introduction

L'utilisation de la parole dans les interfaces utilisateur est une réalité. Des requêtes simples au sujet du solde bancaire ou des transferts d'appel téléphonique, peuvent déjà être manipulées par des systèmes de reconnaissance de la parole. Cependant, si l'interface parlée peut apparaître comme une amélioration certaine, il est important de se rendre compte que la parole par elle-même ne produit pas automatiquement une interface facile à utiliser. Remplacer sans précautions des choix dans des menus par des expressions parlées prédéterminées peut aggraver le problème, puisque l'utilisateur doit employer une liste limitée de commandes arbitraires et qui lui sont inconnues. D'autre part, les interfaces conversationnelles comme définies par (Kölzer, 1999) laissent les utilisateurs dirent ce qu'ils veulent dans leurs propres termes, comme ils le feraient en s'adressant à une autre personne. Par ailleurs, interfacer des humains aux systèmes informatiques utilisant des agents assistants personnels constitue un bon terrain de test pour une approche conversationnelle. Les agents assistants personnels sont des systèmes qui aident les utilisateurs humains utilisant des ordinateurs de façon coopérative (Maes, 1994). Il est donc important de faciliter l'interaction en fournissant une interface conviviale. Nous sommes persuadés que les composants graphiques traditionnels d'une interface utilisateur (GUI), comme les arbres de menu et les boutons, sont inadéquats et peu motivants pour des systèmes où une attention particulière est nécessaire pour piloter le système. Nous sommes convaincus que l'utilisation d'interfaces conversationnelles améliorera la qualité de l'aide qu'un agent assistant peut offrir dans certaines situations spécifiques.

Nous avons développé une architecture pour le dialogue parlé que nous utilisons pour construire une interface pour les agents assistants. Le système de dialogue parlé que nous avons conçu est capable de réaliser des tâches simples, comme localiser un document, mais aussi des tâches plus complexes qui doivent être décomposées en tâches subsidiaires. Dans cette optique, chaque utilisateur a un agent assistant et peut utiliser la voix pour le commander ou lui demander d'accomplir des tâches précises. Dans ce contexte, l'utilisateur et son agent assistant utilisent des dialogues pratiques pour accomplir des buts de façon coopérative comme décrit par (Allen et al, 2001).

Les tâches d'un agent sont décrites comme des fonctions correspondant à ses compétences et dont l'ensemble traduit sa raison d'être, le but pour lequel l'agent a été construit. Comme exemple de compétences, on peut citer la recherche d'un document dans une base de données ou sur Internet, ou encore l'envoi de messages. Cette approche permet à l'utilisateur de rester concentré sur ses activités principales tout en discutant de temps en temps avec son agent. Ainsi, notre approche diffère d'autres approches, comme le système TRIPS (Allen et al, 2001), puisqu'elle, n'utilise qu'un minimum d'informations récoltés à la suite d'échanges avec l'utilisateur dans un domaine bien précis. Ceci peut être réalisé grâce aux ontologies qui décrivent de façon précise le domaine. Le domaine à son tour est bien connu et limité à des applications professionnelles. Les types d'applications que nous avons envisagées nous ont permis de limiter l'espace des dialogues à ceux contenant seulement des actes de langage de type illocutoires et, parmi les classes proposées par (Searle, 1975), à la classe *Directives* : *inform*, *question* ou *anwser*. D'ailleurs, il est difficile d'imaginer actuellement un système qui pourrait manipuler un dialogue véritablement ouvert avec un vocabulaire illimité. Donc, notre approche reste cantonnée à des situations spécifiques qui limitent l'étendue du dialogue et la taille du vocabulaire.

Comme résultat nous espérons que la qualité de l'aide va nettement s'améliorer sans que nous ayons besoin de modifier les compétences internes de l'agent assistant. Notre but est :

- de réduire la surcharge cognitive de l'utilisateur ;
- de faciliter l'exploration de l'ensemble de l'application ;
- de traiter les requêtes hors contexte grâce aux ontologies du domaine et grâce à un traitement linguistique ;
- d'utiliser l'agent assistant même pour les personnes inexpérimentées ou celles qui ont des difficultés, comme les personnes ayant les mains occupées ou les personnes handicapées.

Dans ce qui suit, nous présentons tout d'abord notre interface conversationnelle, puis nous expliquons comment l'interface a été mise en place sur un agent assistant. En conclusion, nous indiquons les résultats attendus, surtout concernant l'augmentation de la qualité de l'aide. Ceci fera l'objet d'une expérimentation ultérieure.

2 Architecture de l'interface conversationnelle

L'architecture globale est composée de trois parties principales (Figure 1) : une fenêtre principale correspondant à saisie de la parole, les modules de traitement linguistiques, et les modules *Agency*, responsables de la gestion du dialogue. Ces modules sont détaillés dans les paragraphes suivants.

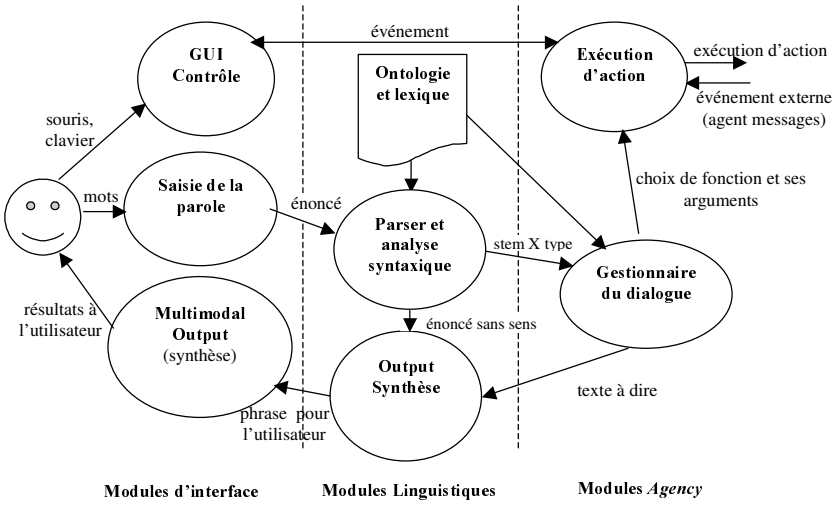

Figure 1 : Diagramme de l'interface

2.1 Les modules d'interface

Si l'interface est prioritairement orientée vers la voix, ce n'en est pas moins une interface multi-modale. Par conséquent, les événements produits par des clics de souris ou à partir du clavier doivent être pris en compte. La fenêtre principale est composée de deux régions distinctes (Figure 2) : une première région utilisant des composants graphiques (la plus petite) et une deuxième région destinée aux échanges en langage naturel. La première région peut contenir des boutons ou des éléments graphiques affichant des informations générales comme des arbres de documents. Les événements correspondants sont directement exécutés.

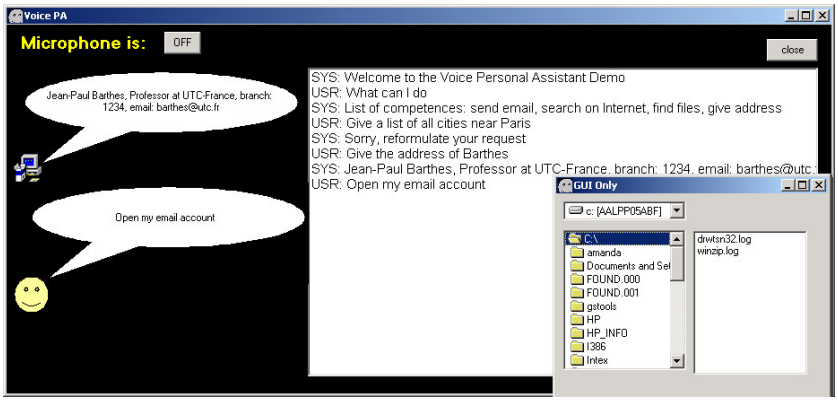

Figure 2 : Copie de la fenêtre principale

Cette région est utilisée de façon intermittente en fonction des besoins comme la sélection d'un fichier dans une arborescence. D'autre part, toutes les entrées vocales sont envoyées aux modules linguistiques appropriés. Une zone d'affichage montre en permanence les énoncés échangés entre l'utilisateur et son agent personnel.

La parole est saisie et traitée par un produit commercial, mot par mot. L'enchaînement est ensuite réalisé avant la transmission à l'analyseur. Pour segmenter les phrases on a employé un algorithme simple basé sur un temporisateur qui utilise une période de silence pour déterminer la fin d'une entrée vocale.

2.2 Le traitement linguistique

Comme dans la plupart des systèmes de dialogue, nous traitons chaque expression parlée séquentiellement. Le Tableau 1 présente quelques expressions possibles et les actions correspondantes qui peuvent être déclenchées.

Énoncés	**Action déclenchée par l'agent**
" Search Agents.doc in the database."	Envoie la requête à l'agent de service *Document Manager*
"Please, open my email account."	Ouvre le courir électronique
"Book two flight tickets to New York."	Hors contexte: non exécuté

Tableau 1 : Exemples d'énoncés

Un énoncé peut être un mot simple ou peut contenir plusieurs mots (une expression ou une phrase). Chaque expression est analysée par un processus indépendant.

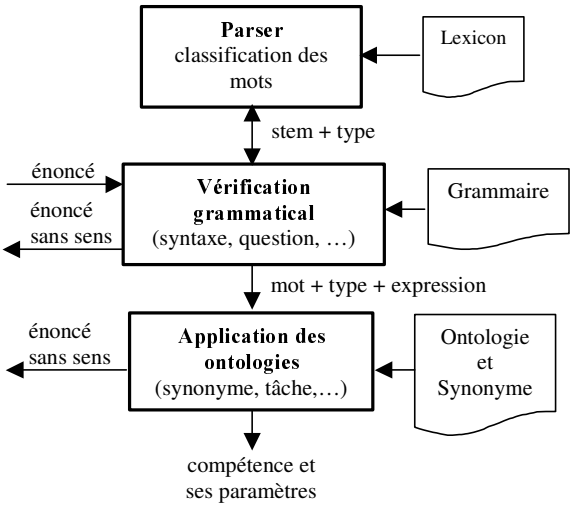

Figure 3 : Le traitement linguistique

Le traitement d'une expression est fait parallèlement à la saisie de la parole. Après saisie, l'énoncé est traité par le module de Vérification Grammaticale (Figure 3).

En raison de la présence de bruit ou à cause d'une mauvaise prononciation, l'expression peut être mal formée et contenir des mots différents des mots réellement prononcés, ou encore contenir des mots vides. Dans la version courante nous enlevons les mots vides. Toutefois, nous avons l'intention d'installer par la suite un mécanisme pour enrichir et/ou modifier l'expression avec l'aide d'un lexique lié au domaine.

L'algorithme d'analyse fonctionne de façon ascendante en remplaçant chaque mot de l'expression par sa catégorie syntaxique (verbe, nom, adverbe, etc.) avec l'aide d'un lexique et de WordNet. L'analyseur est guidé par le module de Vérification Grammaticale qui teste chaque règle grammaticale sur l'expression à traiter.

Le rapport maître-esclave entre l'utilisateur et son agent, nous a permis de limiter l'espace des dialogues à ceux ne contenant que des actes de langage de type illocutoires et parmi les classes proposées par (Searle, 1975), à la classe *Directives* : *inform*, *question* ou *anwser*. Cette décision permet une classification plus précise des énoncés, réduits seulement à trois types.

La nature de l'application nous guide pour un grande nombre d'expressions qui ne se composent que d'un nom et d'une expression verbale. Il faut toutefois signaler que les règles grammaticales sont indépendantes du domaine. Les règles grammaticales ont été divisées, afin de classer une expression dans une des trois catégories : ordre, interrogation ou réponse. Après classification, il est possible de commencer le traitement concernant le domaine, à l'aide d'une ontologie du domaine et d'une liste de synonymes. L'occurrence d'expressions absurdes est rare, puisque notre système essaie d'agir à partir d'une information minimale. Elle peut néanmoins se produire. Par exemple, si une phrase n'est pas bien formée, c'est-à-dire ne respecte pas la structure grammaticale ou si elle est hors domaine, elle est classée comme expression absurde. Dans ce cas, l'utilisateur est invité à reformuler sa phrase.

2.3 Les ontologies et l'interprétation sémantique

Les ontologies ont été utilisées dans ce projet pour bien interpréter les énoncés mais aussi comme aide au raisonnement. Le but principal d'une ontologie est de permettre le partage et la réutilisation d'une connaissance qui a été explicitée de façon formelle. Les composantes clés d'une ontologie sont un vocabulaire des termes de base et des spécifications précises de ce que signifie chaque terme. Selon la taxonomie proposée par (Guarino, 1998) nous avons utilisé une ontologie du domaine et une ontologie de tâches. Les ontologies dans notre système servent: (i) à interpréter le contexte des messages envoyés par d'autres agents ou par l'utilisateur; et (ii) à garder une représentation informatique de la connaissance utile au moment de l'inférence. Les ontologies sont de plus en plus utilisées dans plusieurs domaines.

De récents travaux, (Eriksson, 2003) et (Milward, Beveride, 2003), abordent les questions issues de la conception des ontologies pour l'interaction dans le dialogue. Erikson en particulier énumère les spécificités d'une ontologie conçue spécialement pour des systèmes de dialogue. Cette approche est mieux adaptée et susceptible d'améliorations. Le projet ARTIMIS par exemple (Sadek, 1999), utilise une modélisation à base de prédicats pour le raisonnement. Ce choix conduit à une complexité plus importante.

Nous avons également utilisé WordNet (Fellbaum, 1998) qui est un système lexicologique de référence dont la conception est inspirée de théories psycholinguistiques. Avec WordNet il est possible de deux verbes et de déterminer s'ils font partie de la liste de synonymes. Ainsi, quand on trouve un verbe dans un énoncé, on peut utiliser la liste de synonymes pour savoir s'il fait partie des ontologies qui décrivent les compétences de l'agent assistant. WordNet peut être utile dans plusieurs situations. Par exemple, dans le projet QALC (Ferret et al, 2001) WordNet est utilisé pour trouver les variantes sémantiques d'un mot.

Après l'interprétation syntaxique, nous faisons une interprétation sémantique à l'aide des ontologies et de WordNet. Celle-ci est basée sur l'idée que la signification des énoncés peut être obtenue à partir de mots-clés. Plus précisément, le module d'Application de l'ontologie recherche la liste des verbes qui caractérisent la tâche qui devra être exécutée. Les mots-clés correspondent aux concepts de l'ontologie directement liés à une liste d'actions.

Pour illustrer le fonctionnement de cette approche, considérons l'expression liée à une application de gestion de projets:

USR: *Could you list all project participants?*

Comme l'énoncé est une question et est de plus lié au domaine d'application, le module de Vérification Grammaticale crée une matrice contenant la liste d'éléments (tokens) et leur classification syntactique. En recherchant les éléments dans l'ontologie des tâches, on constate que l'élément « list » est une action. Notez que l'on utilise une liste de synonymes fournie par WordNet (par exemple « list », « enumerate » ou « name » sont des synonymes dans ce cas). On trouve également que « project » est un objet et que « participant » est une de ses propriétés. Nous avons donc une liste qui contient une compétence avec ses paramètres.

Une compétence dans l'ontologie de tâches est décrite comme montre le Tableau 2 :

Compétence *i*
Compétence: et: <liste de mots clés> ou: <liste de mots clés>
Paramètres obligatoires: p*i*: <liste de mots clés> q*i*: <question pour p*i*> t*i*: <type de champ> s*i*: statut <T/F>
Paramètres optionnels: p*i*: <liste de mots clés> q*i*: <question pour p*i*> t*i*: <type de champ> s*i*: statut <T/F>
Action: <script> Autorisation = Yes/No
Statut de l'action: <*executed, pending, running, read, suspended*>

Tableau 2 : Structure d'une compétence

Le champ *compétence* est une liste de mots-clés qui peuvent apparaître dans un énoncé et permettent d'identifier l'action. Les connectives permettent de lister un ou plusieurs mots pour identifier l'action. La compétence peut avoir des paramètres obligatoires ou optionnels. Chaque paramètre est composé : d'une liste de mots clés qui l'identifient; d'une question à poser éventuellement à l'utilisateur ; du type de données prévu et de l'état du paramètre : saisi ou pas. Le champ *action* est un *script* qui sera exécuté quand la compétence sera prête. Le champ autorisation définit si l'utilisateur sera consulté avant l'exécution de la tâche. Enfin, le *statut de l'action* est une variable de contrôle utilisée par le gestionnaire de dialogue pour savoir dans quelle situation se trouve la compétence en mémoire.

2.4 Les modules d'*Agency*

La gestion du dialogue est située dans les modules d'*Agency*. Le gestionnaire du dialogue est capable de choisir un modèle de dialogue adéquat pour une session qui commence. Dans notre cas, un modèle de dialogue contient une liste d'interactions successives possibles, correspondant à une action donnée. Chaque session de dialogue est conduite comme une tâche contenant éventuellement des tâches secondaires. Lorsque l'utilisateur demande une action, le gestionnaire de dialogue essaie de l'exécuter en créant une tâche en mémoire. Cependant, si l'expression initiale ne contient pas toute l'information nécessaire, par exemple s'il manque un paramètre d'action, il déclenche des tâches secondaires pour obtenir la liste de paramètres d'action, en demandant les informations manquantes à l'utilisateur. Une fois la liste complétée, il lance l'exécution en faisant appel à d'autres agents si nécessaire.

Les modules d'*Agency* et l'interface conversationnelle forment l'agent, conçu pour travailler dans une communauté d'agents. Parmi les nombreux types des modèles et de systèmes d'agent qui ont été proposés, des agents cognitifs ont été choisis. L'avantage principal des agents cognitifs est la possibilité de concevoir des comportements intelligents en indiquant un ensemble de compétences. Nous ne proposons pas un langage spécifique de conception d'agents comme dans (Sansonnet, Sabouret, Pitel, 2002), mais les agents sont clonés à partir d'un agent générique (Ramos, 2000). Nous avons utilisé cette approche dans plusieurs projets, comme dans (Barthes, 2003) et (Enembreck, Barthès, 2003).

3 Un système multi-agents gestionnaire personnel

Nous proposons l'application de cette architecture conversationnelle pour construire les agents assistants dans un système multi-agents de gestion personnelle. Nous nous appuyons sur le projet BUCKS (Bottom Up Cooperative Knowledge System) qui modélise un environnement de travail au sein d'un groupe de recherche. Cet environnement offre des services personnalisés mais permet aussi de coopérer. Les services sont réalisés par des agents, la coordination est faite par un agent assistant. Les services (agents) peuvent donc être considérés comme les composants d'un système plus complexe. Pour simplifier, la version en cours de développement offre les services suivants :

- gestion financière : gestion de ressources financières d'un projet ;
- suivi de congrès : gestion des événements scientifiques liés au projet ;
- agenda de réunions : contrôle de l'emploi du temps du responsable du projet ;
- carnet d'adresses : liste d'adresses électroniques ;
- courrier électronique : client e-mail.

L'agent assistant est la seule interface entre l'utilisateur et le système et entre l'utilisateur et les services fournis par les autres agents. Nous prenons l'exemple d'une situation où l'utilisateur veut envoyer un courrier à quelqu'un pour lui rappeler la date limite d'une conférence :

USR: Could you list all conferences with deadline in February?
PA: *Agents et Langue: deadline 7, feb, 2004;*
International Conference on AI Application and Innovations: deadline 16, feb, 2004;
European Conference on Artificial Intelligence – 2004: deadline 18, feb, 2004;
USR: I need the email of Jean-Paul Barthès.
PA: *Jean-Paul Barthès, Professor at UTC-France, branch: 1234, email: barthes@utc.fr*
USR: I would like to send an email to Jean-Paul Barthes.
PA: *What is the subject?, …*

Deux services différents ont été utilisés : le suivi de gestion de congrès et le courrier électronique. Les questions posées par l'agent assistant sont décrites dans la description de chaque compétence et ont été définies au préalable pour chaque service disponible.

4 Amélioration de la qualité de l'aide

L'utilisation d'une interface vocale pour les agents assistants peut améliorer la qualité de l'aide dans certaines situations spécifiques, comme dans le contexte de gestion de connaissances ou la gestion des tâches personnelles. Tout d'abord, elle rend l'exploitation du système plus rapide et plus facile, puisque l'utilisateur n'a pas besoin d'être un expert du domaine, toute la complexité étant prise en charge par l'agent assistant. Nous pensons que les utilisateurs inexpérimentés trouveront l'interface plus facile à utiliser que celles qui emploient des approches plus traditionnelles.

Puisque l'application est un agent assistant, une caractéristique essentielle d'une interface utilisateur a été respectée : la prévisibilité. Nous avons fixé comme condition initiale de fournir des réponses correctes et d'agir selon les ordres de l'utilisateur. Les demandes impossibles, du type hors contexte, sont facilement traitées puisque le système emploie une liste de compétences décrites par une ontologie.

On pourrait aussi penser à des usages moins ordinaires comme les interfaces pour les appareils électroménagers. Un nombre de plus en plus important de chercheurs sont en train de développer des applications s'appuyant sur des interfaces conversationnelles pour ce type d'appareil (Quesada et al., 2001), (Yates et al. 2003). L'association d'un agent assistant et d'une interface vocale peut être un apport significatif à cet groupe d'applications.

Ces conclusions résultent des toutes premières observations. Nous projetons de tester cette approche en comparant la convivialité et la qualité de l'aide de l'application de gestion personnelle, et l'approche traditionnelle purement graphique. Nous développons actuellement un agent assistant parlant et un autre agent doté d'une interface graphique. Un groupe d'utilisateurs les utilisera afin que nous puissions mesurer la qualité de l'aide. En fonction de l'accessibilité aux services disponibles, nous pourrons évaluer s'il y a eu ou non une amélioration. Une analyse plus approfondie des résultats fera l'objet d'un prochain article.

Références

Allen J. et al (2001), An architecture for more realistic conversational systems, Actes de *Intelligent User Interfaces 2001 (IUI-01)*, Santa Fe, NM, USA.

Barthès J. (2003), MASH Environments for corporate KM, Actes de *The Knowledge Management Workshop of IJCAI 2003*, Acapulco, Mexico.

Enembreck F., Barthès J. (2003), Agents for collaborative filtering, Actes de *Cooperative Information Agents VII – CIA 2003*, Springer, pp. 184-191.

Eriksson A. (2003), Design of ontologies for dialogue interaction and information extraction, Actes de *International Joint Conference on Artificial Intelligence IJCAI – 03,* Acapulco Mexico.

Fellbaum C. (1998), *WordNet: An electronic lexical database*. MIT Press, Cambridge, MA, USA.

Ferret O. et al (2001), Utilisation des entités nommées et des variantes terminologiques dans un système de question-réponse, Actes de *TALN 2001*, Tours, France.

Guarino N. (1998), Formal ontology in information systems, Actes de *FOIS'98*, IOS Press, pp. 3-15.

Kölzer A. (1999), Universal dialogue specification for conversational systems, Actes de *IJCAI 99 – Workshop on Knowledge and Reasoning in Practical Dialogue Systems*.

Maes P. (1994), Agents that reduce work and information overload. *Communication ACM*, Vol. 37, No.7 , ACM Press, 31-40, 146.

Milward D., Beveride M (2003), Ontology-based dialogue systems, Actes de *International Joint Conference on Artificial Intelligence IJCAI – 03,* Acapulco, Mexico.

Quesada J. et al (2001), Dialogue managements in a home machine environment: linguistic components over an agent architecture, Actes de *SEPLN, 2001*. pp. 89-98.

Ramos M. (2000), Structuration et évolution conceptuelles d'um agent assistant personnel dans les domaines techniques, *thèse présenté à l'UTC*, décembre 2000.

Sansonnet J., Sabouret N., Pitel G. (2002), An Agent Design and Query Language dedicated to Natural Language Interaction, Poster présenté à l'*AAMAS 02*.

Sadek D. (1999), Design Considerations on Dialogue Systems: From Theory to Technology - The Case of Artimis, Actes de *ESCA Workshop Interactive Dialogue in Multi-modal Systems*, pp. 173-187.

Searle J. R. (1975), A taxonomy of illocutionary acts, Actes de *Language, Mind and Knowledge*, Vol. 7, University of Minnesota Press, pp. 344-369.

Yates A. et al. (2003), A reliable natural language interface to household appliances, Actes of *Intelligent User Interfaces (IUI-03),* ACM Press, pp. 189-196.