ICAI : UNE INTERFACE CONVERSATIONNELLE POUR UNE AIDE INTELLIGENTE

Soutenue le 02 Décembre 2005 devant le jury composé de :

MME. Zahia GUESSOUM (Président)
MM. Jean-Paul A. BARTHES (Directeur de thèse)
Bertrand DAVID (Rapporteur)
René MANDIAU (Rapporteur)
Claude MOULIN

À Leticia

Remerciements

La réalisation d'une thèse de doctorat n'est pas une tâche simple. L'aboutissement de celle-ci a été possible grâce plusieurs personnes.

Je voudrais, tout d'abord, remercier Jean-Paul A. Barthès, pour m'avoir permit de partager ses connaissances. Ses qualités humaines et techniques m'ont beaucoup aidé.

Je remercie la CAPES et la PUCPR pour leur support financier pour mener cette thèse. Je tiens aussi à remercier mes collègues du laboratoire HEUDIASYC pour leur accueil et mes collègues du thème DOC.

Je tiens à remercier Mme. Mouret non seulement pour le travail de correction de français, mais surtout pour son amitié. Je remercie aussi à tous mes amis de la « Ville Impériale ».

Je voudrais également remercier toute ma famille et ma belle-famille, qui m'ont tous beaucoup manqué.

Et finalement, mais de façon très forte, je tiens à remercier mon épouse Leticia. C'est à cause de sa patience et compréhension que je suis arrivé à mener ce travail.

Résumé

Cette thèse concerne la conception d'une interface conversationnelle pour les agents assistants personnels. Ce travail s'inscrit dans un processus de développement d'un modèle générique d'agent assistant personnel. Après avoir examiné les divers types d'interfaces utilisées par des agents assistants, nous avons conclu que les interfaces conversationnelles étaient particulièrement bien adaptées à nos besoins. Cette thèse est donc focalisée sur la définition et la conception d'une interface conversationnelle pour un agent assistant, en tenant compte de plusieurs contraintes et hypothèses liées à l'agent lui-même, à son pilotage et à la nature du rapport utilisateur-agent, mais aussi aux caractéristiques des applications potentielles. Nous estimons qu'une interface conversationnelle spécialement conçue pour l'agent assistant, simplifiera son pilotage, entraînant une amélioration de la qualité de l'aide apportée par rapport aux interfaces classiques purement graphiques. Ceci diminuera la charge cognitive de l'utilisateur, puisque le travail d'interprétation et reconnaissance des intentions de ce dernier devient de responsabilité de la machine. Nous postulons aussi que le comportement intelligent de l'agent peut contribuer à l'amélioration de la performance liée à une telle interface, grâce à une politique d'affichage d'informations.

De notre travail est né le concept d'interface conversationnelle pour une aide intelligente - ICAI. Une interface conversationnelle pour une aide intelligente est le résultat de la conjonction d'un mécanisme conversationnel en langage naturel parlé et de la gestion intelligente de ce mécanisme, permettant le déroulement d'un dialogue coopératif, et capable de gérer le déclenchement de plusieurs tâches à la demande de l'utilisateur, tout en n'exigeant qu'un minimum d'effort de la part de ce dernier.

Notre approche s'appuie sur les ontologies, fil conducteur de nos travaux, qui nous permettent d'interpréter les messages, provenant de l'utilisateur ou d'ailleurs.

Mots-clés : agent assistant, interface conversationnelle, ontologie, système multi-agents.

Abstract

This thesis concerns the design of a conversational interface for personal assistant agents. Precisely, we are interested in developing a conversational interface for a generic personal assistant. We studied the typical user interface used in personal assistants. We conclude that conversational interfaces are the one especially suitable for personal assistants. To design the conversational interface, we took in account several constraints related to the agent itself and the potential applications. As the result of this approach we expect to improve the quality of assistance and to reduce the user's cognitive load.

In this thesis we present a new concept called ICAI: A Conversational Interface for an Intelligent Assistance. An ICAI is the result of the conjunction of a conversational spoken natural language mechanism and an intelligent management of such mechanism, enabling a cooperative dialogue between the user and the assistant agent.

Our approach is based on ontologies that allow us to interpret messages from human and artificial agents.

Keywords: personal assistants, conversational interface, ontologies, multi-agent systems.

Table des matières

Table des figures

Tableaux

Liste d'abréviations

ACA : Agents conversationnels animés.

AS : Agent de service.

AP : Agent assistant personnel.

API : *Application Programming Interface*.

BDI : *Believe*, *Desire* et *Intentions*.

BNF : *Backus-Naur Format*.

CFG : *Context-Free Grammar.*

DARPA : *Defense Advanced Research Projects Agency*.

DLL : *Dynamic Link Library*.

EIAH : Environnements interactifs d'apprentissage humain.

FIPA : *Foundation for Intelligent Physical Agent*.

IA : Intelligence artificielle.

ICAI : Interface conversationnelle pour une aide intelligente.

OMAS : *Open Multi-Agent System.*

OMAS-WA : *Open Multi-Agent Systems with Assistants.*

OWL : *Web Ontology Language.*

KQML : *Knowledge Query and Manipulation Language***.**

LTAG : *Lexicalized Tree Adjoining Grammar***.**

PDA : *Personal Digital Assistant*.

SAPI : *Speech Application Programming Interface*.

SMA : Systèmes multi-agents.

UDP : *User Datagram Protocol*.

VoiceXML : *Voice Extensible Markup Languag.*

CHAPITRE 1

Introduction

1. Présentation

Cette thèse concerne la conception d'une interface conversationnelle pour les agents assistants personnels. Ce travail s'inscrit dans un processus de développement d'un modèle générique d'agent assistant personnel initié par Ramos (2000) et poursuivi par Enembreck (2003). Au départ, l'architecture générale proposée par Ramos était composée d'un agent assistant personnel créé pour aider l'utilisateur dans ses tâches informatiques quotidiennes et d'agents spécialisés appelés agents de service. Par la suite, quelques modifications et améliorations furent introduites par Enembreck dans le but principal de concevoir un modèle d'agent adaptatif pourvu d'une méthode propre d'apprentissage. De plus, l'approche proposée par Enembreck réduisait la complexité de l'agent assistant en distribuant les compétences entre des agents de coordination (dits de staff).

Malgré la cohérence des modèles proposés par Ramos, et à la suite d'une première étude de la personnalisation de l'interaction avec l'utilisateur menée par Enembreck, nous avons estimé que l'interface de l'agent assistant méritait une étude plus approfondie. En conséquence, après avoir examiné les divers types d'interfaces utilisées par des agents, nous avons conclu que les interfaces conversationnelles étaient particulièrement bien adaptées à nos besoins. Nous nous sommes donc focalisés sur la définition et la conception d'une interface conversationnelle pour un agent assistant, en tenant compte de plusieurs contraintes et hypothèses liées à l'agent lui-même, à son pilotage et à la nature du rapport utilisateur-agent, mais aussi aux caractéristiques

des applications potentielles. Nous estimons que ce n'est qu'après avoir réalisé ce travail et après doté l'agent de l'interface conversationnelle correspondante que le modèle générique de l'agent assistant pourra effectivement être validé.

Notre but est donc de concevoir une interface conversationnelle capable de mener des conversations en langage naturel parlé comme celle du Tableau 1 (*AP* = agent assistant et *USR* = utilisateur).

AP (1):	Welcome to the Voice Personal Assistant.
USR (2):	Open the report on agents.
AP (3):	*AP rajoute le message « The report is open. » dans la file de message de basse priorité.*
AP (4):	You have a meeting scheduled for today: participant: Mike, time: 11:00, place: office, description: agent platform.
USR (5):	Send an email to Mike.
AP (6):	What is the subject?
USR (7):	Our meeting today.
AP (8):	What is the content?
USR (9):	We have a meeting today. Please bring your report on agent platforms.
AP (10):	You have a new email: sender: Mary.
USR (11):	OK, could you open my email account?
AP (12):	Yes.
AP (13):	*AP rajout le message « Email account loaded. » dans la file de message de basse priorité.*
USR (14):	List all meetings in my office tomorrow morning.
AP (15):	You have no meetings scheduled for tomorrow morning.
USR (16):	What is the email address of Mary?
AP (17):	The email is: mary@company.com.
USR (18):	Send an email to her and to Mike.
AP (19):	What is the subject?
USR (20):	Setting a meeting for tomorrow.

AP (21):	What is the content?
USR (22):	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?
AP (23):	You have a new email: sender: Paul.

Tableau 1 : Un exemple de dialogue.

Le contexte de cette conversation a été extrait du cas issu d'expérimentations du système de capitalisation et de gestion de connaissances développé par Tacla (2003) à l'UTC. Dans l'approche proposée par Tacla, les utilisateurs sont immergés dans un espace de travail collaboratif (un environnement de gestion des connaissances - EGC), où toutes leurs actions peuvent être prises en compte pour la capitalisation des connaissances, comme l'envoi d'un courriel, la rédaction d'un rapport, la recherche sur Internet. L'agent assistant et son staff d'agents sont capables de mettre à jour des mémoires personnelles au fur et à mesure de ses activités. Il est capable aussi d'identifier deux ou plusieurs personnes intéressées par les mêmes sujets et de les mettre en relation.

Étant donné ce contexte, supposons un groupe de trois personnes (John, Mike et Mary) travaillant dans le cadre d'un même projet. Chacun d'entre eux a son propre agent assistant capable d'accomplir les tâches décrites dans le paragraphe précédent. Le dialogue du Tableau 1 illustre une conversation entre John et son agent assistant. Après le message qui marque le début de la session (1), l'utilisateur demande à son agent assistant l'ouverture d'un rapport sur les agents. Cette tâche est exécutée par un des agents de service. Une fois le rapport ouvert, John y consacre quelques instants. L'agent de service responsable de la gestion de l'agenda, envoie un message d'alerte pour l'informer de la proximité d'une réunion (4). Ensuite, John décide d'envoyer un courriel à Mike pour lui demander d'apporter le rapport qu'il prépare sur les plates-formes d'agents (5-9). En interprétant le contenu du message, l'agent assistant constate que le sujet du message, les plates-formes d'agents, a un intérêt pour un autre participant du groupe de projet : Mary. Sans que Mike en soit averti, son agent assistant notifie l'agent assistant de Mary qui à son tour imprime un message d'information. Mary rédige alors un courriel à John pour lui proposer une présentation de ses activités. Une fois le message reçu, l'agent assistant de John l'informe (10) tout de suite. John pose une question à l'agent assistant (11) qui lui répond affirmativement. On note la stratégie coopérative du dialogue, qui mène l'agent assistant à répondre à la question posée et tout de suite après, à ouvrir la boîte à lettre de John. À travers l'énoncé (14) l'utilisateur cherche à savoir s'il y aura des réunions le lendemain matin. La

réponse arrive et l'agent assistant l'imprime immédiatement. À ce stade du dialogue, John cherche à savoir s'il peut proposer une réunion entre Mary, Mike et lui même. John demande à son agent assistant s'il connaît l'adresse électronique de Mary. En recevant une réponse positive, John écrit un courriel à Mike et à Mary. Au cours de son travail de saisie du courriel (18-22), un message d'alerte arrive (23) annonçant un nouveau courriel. Le message restera dans la file d'attente jusqu'au moment où la pile de tâches sera vide, ce qui va se produire après l'envoi du courriel que John est en train d'éditer.

La conception et la réalisation d'une interface conversationnelle pour l'agent assistant sont des tâches difficiles. Les principaux défis liés au développement d'une telle interface, comprennent les éléments suivants :

- la conception d'un mécanisme d'analyse syntaxique robuste, capable de traiter des énoncés parlés, pas forcément bien reconnus ;
- la conception d'un analyseur sémantique, capable d'interpréter les énoncés, pas toujours liés au domaine de l'application ;
- la conception d'un gestionnaire de dialogue capable de gérer des conversations sur des domaines spécifiques, capable de déclencher plusieurs tâches simultanément et capable de bien gérer l'arrivée de messages provenant de plusieurs sources différentes (d'autres agents) ;
- la mise en place d'une structure de mémoires, capable de fournir des informations à la gestion du contexte de la conversation ;
- la conception d'une politique d'affichage d'informations, gérée par l'agent assistant, pour organiser la façon d'interrompre l'utilisateur : pour lui poser des questions et pour lui présenter des informations diverses, comme des résultats d'exécution des tâches ;
- la capacité d'ancrer le raisonnement de l'agent sur les ontologies dont l'agent dispose.

2. Objectifs et contributions

Nous estimons qu'une interface conversationnelle spécialement conçue pour l'agent assistant, simplifiera son pilotage, entraînant une amélioration de la qualité de l'aide apportée par rapport aux interfaces classiques purement graphiques (connues sur le nom de WIMP – *window,*

icon, menu, pointer). Ceci diminuera la charge cognitive de l'utilisateur, puisque le travail d'interprétation et de reconnaissance des intentions de ce dernier devient la responsabilité de la machine. Nous postulons aussi que le comportement intelligent de l'agent peut contribuer à l'amélioration de la performance liée à l'utilisation d'une telle interface, grâce à une politique effective d'affichage d'informations.

Nos travaux se situent à l'intersection de deux domaines de recherches : les agents intelligents et l'interaction homme-machine.

Le domaine de l'interaction homme-machine a évolué énormément grâces aux avances technologiques. Si dans un premier temps les limitations physiques imposaient l'utilisation des interfaces purement textuelles ou graphiques, aujourd'hui, la recherche s'est concentrée sur comment faciliter les échanges d'information avec l'utilisateur humain, en s'adaptant à ses besoins et à ses capacités. Les interfaces ont la possibilité d'utiliser d'autres modes de communication que le clavier, l'écran et la souris, comme la parole ou les gestes. Les agents intelligents, plus particulièrement les agents assistants, peuvent bénéficier de ces avances.

Dans cette thèse, nous proposons une interface conversationnelle spécialement conçue pour les agents assistants, permettant à l'utilisateur d'utiliser le langage naturel parlé pour piloter son assistant. De notre travail est né le concept d'ICAI - Interface Conversationnelle pour une Aide Intelligente. Une ICAI est le résultat de la conjonction d'un mécanisme conversationnel en langage naturel parlé et de la gestion intelligente de ce mécanisme, permettant le déroulement d'un dialogue coopératif et capable de gérer le déclenchement de plusieurs tâches à la demande de l'utilisateur, avec un minimum d'effort de la part de ce dernier.

Notre approche s'appuie sur les ontologies, fil conducteur de nos travaux, qui nous permettent d'interpréter les messages, provenant de l'utilisateur ou d'ailleurs. Les ontologies sont la seule forme de représentation des connaissances dont nous avons besoin.

Les principales contributions de notre recherche sont :

- la conception de l'interface conversationnelle, basée sur :
    - la définition d'un système de dialogue fondé sur les actes de langage directifs (ordre, question et réponse) ;
    - l'adoption d'une stratégie coopérative pour le système de dialogue ;
    - l'ancrage sémantique à travers les ontologies ;

    - la séparation physique des connaissances de domaine et de tâches.
- la mise en place d'une politique d'affichage d'informations, un premier pas vers une politique de présentation ;
- une liste de principes de base à respecter pour élaborer une ontologie pour la gestion de dialogues avec un agent assistant.

3. Plan de la thèse

Ce mémoire est organisé suivant le schéma de la Figure 1.

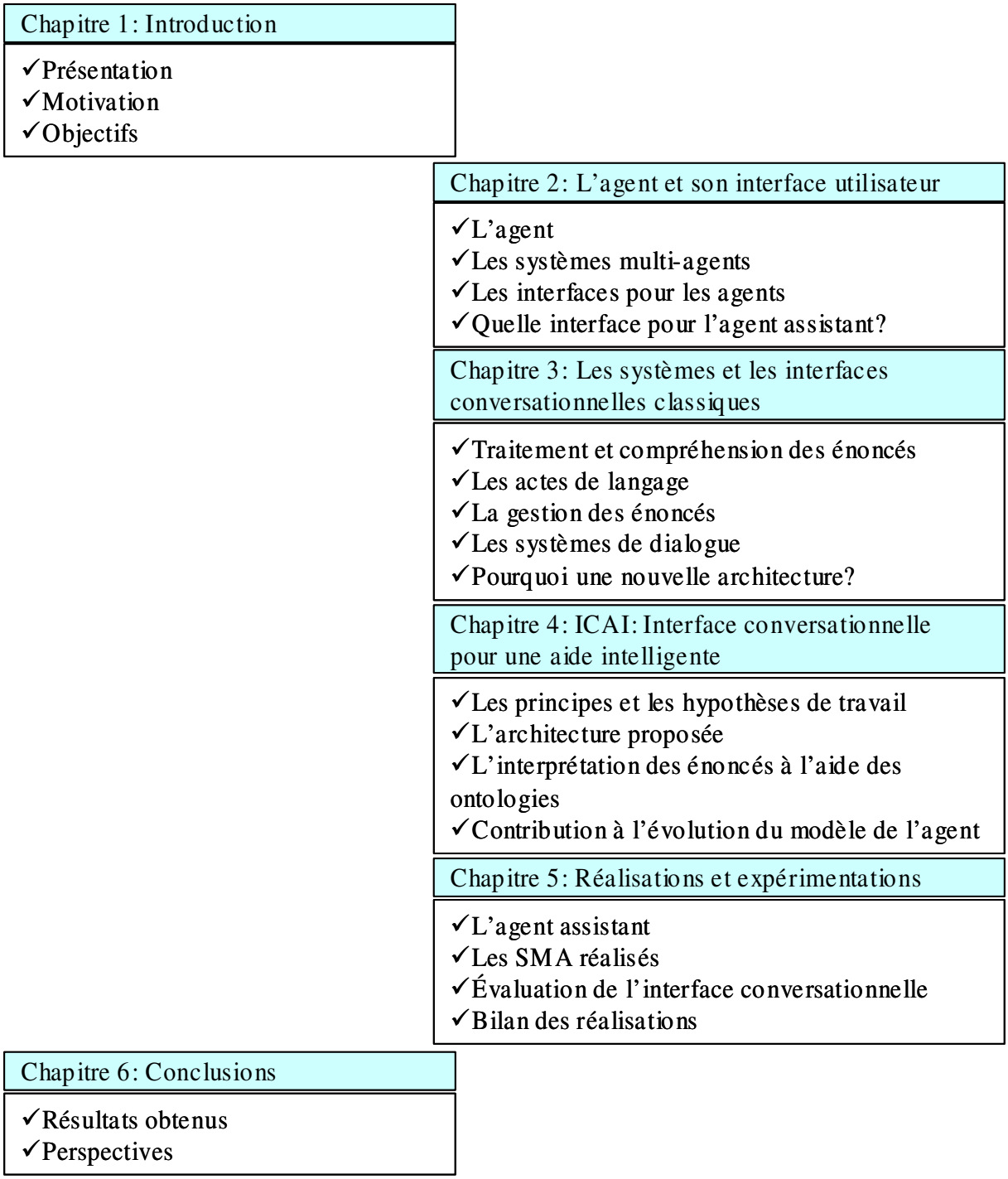


Figure 1 : Structure de la thèse.

Les chapitres sont classés comme illustré sur la Figure 2, selon les aspects scientifiques, technologiques ou applicatifs de la recherche.

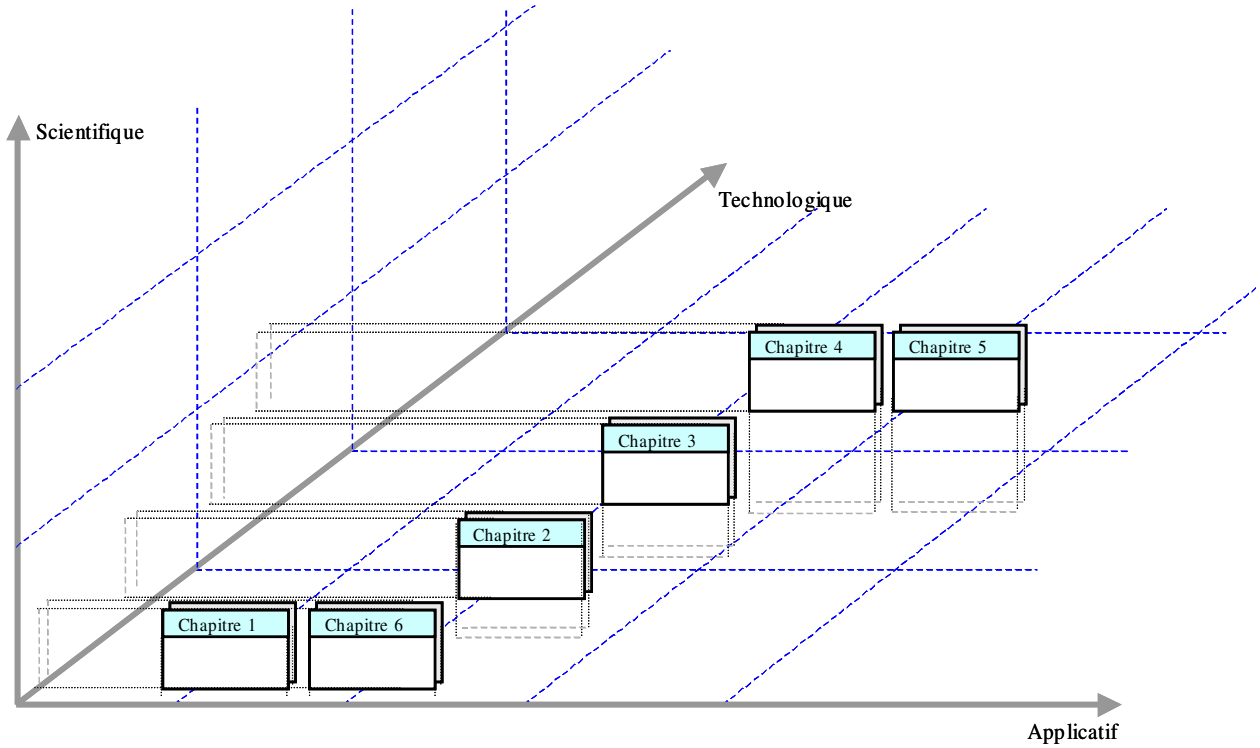


Figure 2 : Les trois axes d'encadrement de la thèse.

3.1. Les chapitres de la thèse

La thèse est organisée en deux parties : La première partie concerne les agents et leur interface utilisateur et comprend les chapitres 2 et 3. La deuxième partie englobe les chapitres 4 et 5 et concerne la conception et la réalisation de l'interface conversationnelle et la mise en œuvre des deux prototypes de validation.

Chapitre 2 - L'agent et son interface utilisateur

Dans ce chapitre, nous présentons les concepts principaux concernant les systèmes d'agents. Nous présentons les agents, les systèmes multi-agents et les agents assistants personnels en portant une attention spéciale aux aspects liés à l'interaction avec l'utilisateur. Cela nous permettra d'identifier les points clés de la relation entre l'utilisateur et l'agent et nous guidera au cours de la recherche de solutions présentées dans les chapitres suivants.

Chapitre 3 - Les systèmes et les interfaces conversationnelles classiques

Dans ce chapitre, nous présentons les composantes nécessaires pour la mise en œuvre d'une interface conversationnelle. Le chapitre débute par la présentation générale d'une interface conversationnelle, décrivant ses principales composantes. Nous discutons d'abord les aspects liés à la saisie de la parole pour ensuite progresser vers les mécanismes d'analyse sémantique et la

gestion du dialogue. Ce chapitre met l'accent sur les éléments plus sensibles, comme les grammaires, les systèmes de dialogue et la place des ontologies dans la gestion du dialogue, tout en préparant le lecteur à la compréhension de notre architecture, présentée dans le chapitre suivant.

Chapitre 4 – ICAI : Interface conversationnelle pour une aide intelligente

Nous présentons dans ce chapitre l'architecture de l'interface conversationnelle qui nous permettra de réaliser la nouvelle interface conversationnelle de l'agent assistant. Le chapitre aborde toutes les étapes de conception de l'architecture, en prêtant une attention spéciale à la place des ontologies dans le traitement syntaxique et sémantique. Nous introduisons nos contributions à l'évolution du modèle de l'agent assistant, comme la nouvelle politique d'affichage d'informations et la façon dont l'interface est couplée à l'agent.

Chapitre 5 - Réalisations et expérimentations

Ce chapitre est consacré au côté expérimental de la recherche. Notre but est essentiellement de construire un agent assistant personnel avec l'interface conversationnelle fondée sur les caractéristiques et modèles proposés dans les chapitres précédents. Nous avons conçu et réalisé deux prototypes de SMA pour valider l'agent assistant avec sa nouvelle interface.

Chapitre 6 - Conclusions, contributions et perspectives

À la fin du mémoire, nous présentons nos conclusions, contributions et les perspectives d'évolutions de la recherche.

Après ces chapitres, viennent les références bibliographiques et quelques annexes.

CHAPITRE 2

"*I have always wished that my computer would be as easy to use as my telephone. My wish as come true. I no longer know how to use my telephone*."

Bjarne Stroustrop (originator of C++)

L'agent et son interface utilisateur

Dans ce chapitre, nous présentons les concepts principaux concernant les systèmes d'agents. Nous allons présenter les agents, les systèmes multi-agents et les agents assistants personnels en portant une attention spéciale aux aspects liés à l'interaction avec l'utilisateur. Cela nous permettra d'identifier les points clés de la relation entre l'utilisateur et l'agent et nous guidera au cours de la recherche de solutions présentées dans les chapitres suivants. À la fin de ce chapitre nous présentons les éléments qui nous ont guidés pour la définition du type d'interface d'un agent assistant.

1. L'agent

Les nombreuses définitions d'agent que nous pouvons trouver dans la littérature témoignent de l'intérêt croissant des chercheurs pour les agents. Russel et Norvig (1995) par exemple, proposent la définition suivante :

« Un agent est une entité qui peut percevoir son environnement par des capteurs et agir sur cet environnement par des effecteurs. »

Evidemment cette définition est trop générale. Jennings (1998) à son tour nous propose :

« Un agent est une entité informatique, placée dans un environnement, qui est capable d'exécuter des actions d'une façon flexible et autonome afin d'atteindre les objectifs établis lors de sa conception. »

Sans vouloir nous engager dans des discussions théoriques, nous allons utiliser la définition proposée par Enembreck (2003) comme étant la mieux adaptée à nos besoins :

« Un agent est une entité logicielle qui joue un rôle et qui peut avoir plusieurs caractéristiques (autonomie, adaptabilité, perception, sociabilité) et modèles internes (de soi-même, des autres, de communication ou de l'environnement). Le rôle de l'agent et les ressources informatiques disponibles définissent les caractéristiques et modèles que l'agent doit contenir dans le cadre d'une application spécifique. »

Cette définition évoque des caractéristiques et des modèles internes que nous allons présenter maintenant.

1.1. Propriétés des agents

Pour être capable de préciser les différents types et caractéristiques individuelles de chaque agent, nous devons d'abord dresser une liste de propriétés « désirables » pour n'importe quel agent. Une liste non exhaustive est présentée dans le Tableau 2.

Propriétés	**Description**
Apprentissage	Capacité d'évolution dynamique de modèles internes en fonction d'expériences passées.
Autonomie	L'agent est le seul responsable du contrôle de l'exécution de ses tâches.

Communicativité	Capacité sociale de communication avec d'autres agents artificiels et naturels (humains).
Continuité au cours du temps	A priori l'agent doit rester actif sans interruption (tourner en continu).
Mobilité	Capable de se déplacer dans un réseau de communication.
Personnalité	Capacité d'exprimer ses émotions.
Pro-activité	Prendre l'initiative pour attendre un but.
Réactivité	Capacité de percevoir et de répondre à des changements d'environnement.
Honnêteté	Concerne la volonté ou pas de donner de fausses informations.

Tableau 2: Propriétés des agents.

Ces propriétés peuvent être conjuguées en fonction des objectifs donnés à chaque agent. Dans la plate-forme OMAS développée dans notre laboratoire (décrite en détail dans la section 2.1) tous les agents, indépendamment de leur type, ont les propriétés suivantes : apprentissage, autonomie, communication, continuité, pro-activitié et réactivité.

Il faut rajouter à cette liste de propriétés une dimension qui donne à l'agent sa capacité de raisonnement. Pour cela deux stratégies se démarquent : la cognition et la réactivité.

1.1.1 Cognition

Les agents cognitifs ont un modèle explicite et symbolique du monde et sont capables de raisonner sur le monde. Ils utilisent des formes complexes de représentation de connaissances et sont capables d'établir de plans spécifiques au cours de leur vie pour atteindre leurs buts. Le modèle d'agent cognitif le plus populaire actuellement est le modèle BDI (*Beliefs*, *Desires* et *Intentions*) proposé par Rao et Georgeff (1995). Ce modèle est basé sur trois concepts :

- Croyances (*Beliefs*) : indique que l'agent est capable de représenter ses croyances en fonction de l'environnent dans lequel il est immergé ;
- Buts (*Desires*) : indique les objectifs que l'agent a planifiés ;
- Intentions (*Intentions*) : indique l'état que l'agent a choisi pour atteindre ses buts.

1.1.2 Réactivité

Les agents dits réactifs sont des agents pour lesquels la prise de décision est réalisée par une simple fonction de perception/action. Ces agents réagissent seulement aux stimuli externes et ne communiquent pas directement entre eux. Leur capacité de raisonnement est très limitée.

Dans nos expérimentations, présentées dans le Chapitre 5, nous nous servons d'agents cognitifs ou réactifs selon nos besoins.

Ainsi, une vision générale d'un agent autonome et de son environnent est donnée sur la Figure 3. La structure interne de nos agents sera décrite dans la section 2.1.1.

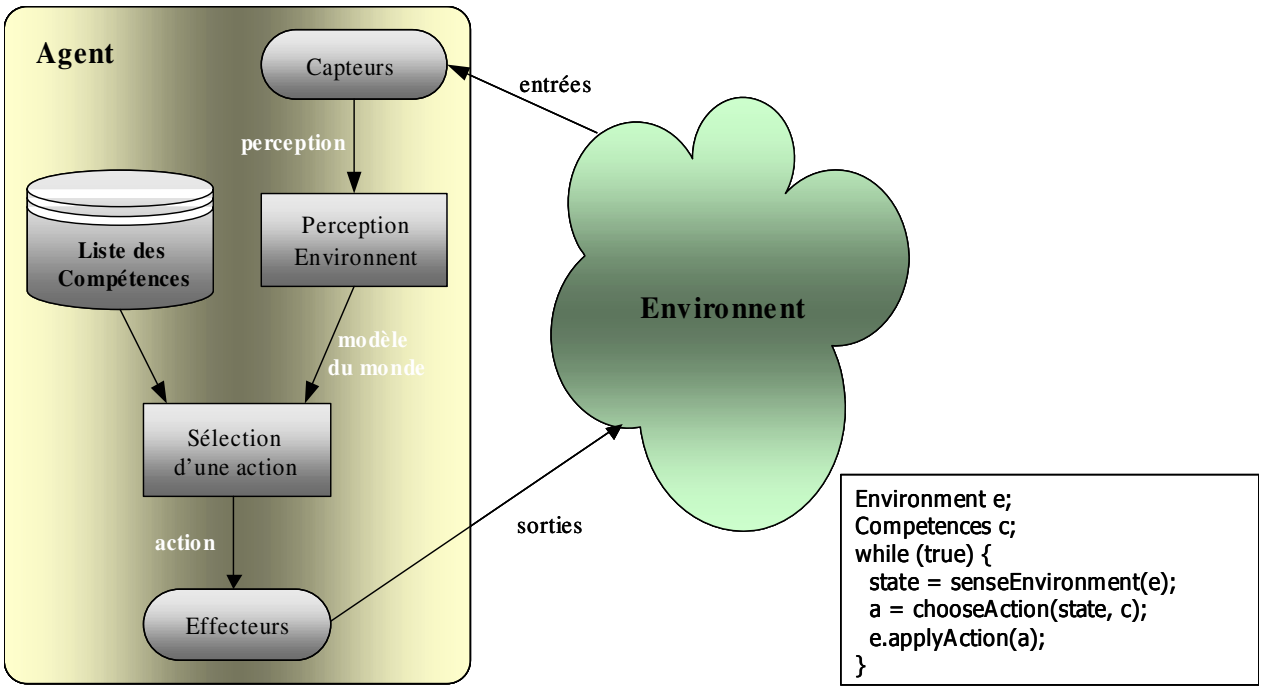


Figure 3 : L'agent et l'environnent (Singh et Huhns, 2005).

1.2. La communication inter-agents

La communication entre les agents est un aspect fondamental dans une application basée sur des agents. Les agents doivent utiliser un langage de communication commun pour permettre l'échange d'informations et de connaissances. Le langage KQML (*Knowledge Query and Manipulation Language*) (Finin et al. 1994) et le standard FIPA ACL (FIPA, 2003) sont actuellement les plus utilisés. La FIPA (*Foundation for Intelligent Physical Agent*) est une association à but non lucratif établie à Genève (Suisse) en 1996. Le but de la FIPA est de fournir des standards pour faciliter l'interopérabilité entre les systèmes à base d'agents.

Un message FIPA ACL contient un ou plusieurs paramètres. Les paramètres nécessaires pour la communication changent selon la situation. Le seul paramètre qui est obligatoire dans tous les messages est le paramètre *performatif*, bien que la plupart des messages contiennent également des paramètres concernant l'expéditeur, le récepteur et le contenu. Un *performatif* est un « Acte de Communication » qui exprime la nature du message. Par exemple, dans FIPA ACL, il y a quatre *performatifs* de base : *inform*, *request*, *confirm* et *disconfirm*.

Le Tableau 3 liste les paramètres d'un message FIPA ACL.

Paramètre	Description
Performative	Type d'acte de communication.
Sender	Expéditeur du message.
Receiver	Récepteur du message.
Reply-to	Envoyer le message à *Reply-to* au lieu de *Receiver*.
Content	Contenu du message.
Language	Indique le langage utilisé pour décrire le *Content*.
Encoding	Codage du contenu.
Ontology	L'ontologie est employée en même temps que le paramètre *Language* pour permettre l'interprétation du contenu.
Protocol	Indique le protocole d'interaction.
Conversation-id	Contrôle de la conversation.
Reply-with	Utilisé pour identifier la réponse à ce message.
In-reply-to	Utilisé pour identifier la réponse à ce message.
Reply-by	Indique le délai d'attente d'une réponse.

Tableau 3 : Paramètres d'un message FIPA ACL.

La Figure 4 donne un exemple d'échange de messages entre un agent ***i*** et un agent ***j***.

L'agent ***i*** demande à ***j*** de livrer une boîte à un certain endroit. ***j*** répond qu'il est d'accord sur la demande mais que la demande a une priorité basse.

```
(request
     :sender (agent-identifier :name i)
     :receiver (set (agent-identifier :name j))
     :content
        "((action (agent-identifier :name j)
          (deliver box017 (loc 12 19))))"
     :protocol fipa-request
     :language fipa-sl
     :reply-with order567)

(agree
     :sender (agent-identifier :name j)
     :receiver (set (agent-identifier :name i))
     :content
        "((action (agent-identifier :name j)
          (deliver box017 (loc 12 19)))
          (priority order567 low))"
     :in-reply-to order567
     :protocol fipa-request
     :language fipa-sl)
```

Figure 4 : Un exemple de messages FIPA (2003a).

Toute interprétation de message passe forcément par l'analyse de son contenu. Chaque agent est libre d'interpréter les messages selon ses propres modèles. Cependant, pour assurer une sémantique commune à la terminologie utilisée pour exprimer les messages, il nous faut une forme de représentation universelle. En général cette représentation est une *ontologie*. L'ontologie est utilisée pour l'interprétation du contenu des messages reçus par l'agent, dans la

représentation de ses compétences, mais aussi pour faciliter l'interaction avec l'utilisateur humain (quand c'est nécessaire). Nous reviendrons sur ces points plus tard dans ce mémoire.

L'agent aura aussi besoin d'un mécanisme de représentation de ses connaissances. Dans la section suivante, nous allons décrire rapidement les systèmes les plus classiques de représentation des connaissances.

1.3. La représentation de connaissances dans les agents

Dans ce mémoire, nous allons nous confronter à une série de définitions tout à fait utiles et cohérentes. Peut-être la définition la plus complexe et controversée est celle du concept d'intelligence. Nous n'allons pas entrer dans cette discussion et n'avons pas la prétention de définir le concept d'intelligence formellement et rigoureusement. En intelligence artificielle (IA) tous les discours sur l'intelligence convergent sur deux points : la capacité de raisonner et de représenter ses connaissances de façon explicite. Représenter ses connaissances permettra à un système (comme nos agents) dit intelligent d'exécuter des actions d'une façon flexible, autonome et, peut être, de produire de nouvelles connaissances.

Les systèmes de représentation de connaissances ont été étudiés dès les débuts de l'informatique et plus particulièrement par les chercheurs en IA à partir des années 60s. Plusieurs grandes familles de systèmes et formalismes de représentation se sont développés (voir Figure 5). Une multitude de systèmes nous permettent des construire des base de connaissances utiles pendant toute la « vie » de l'agent.

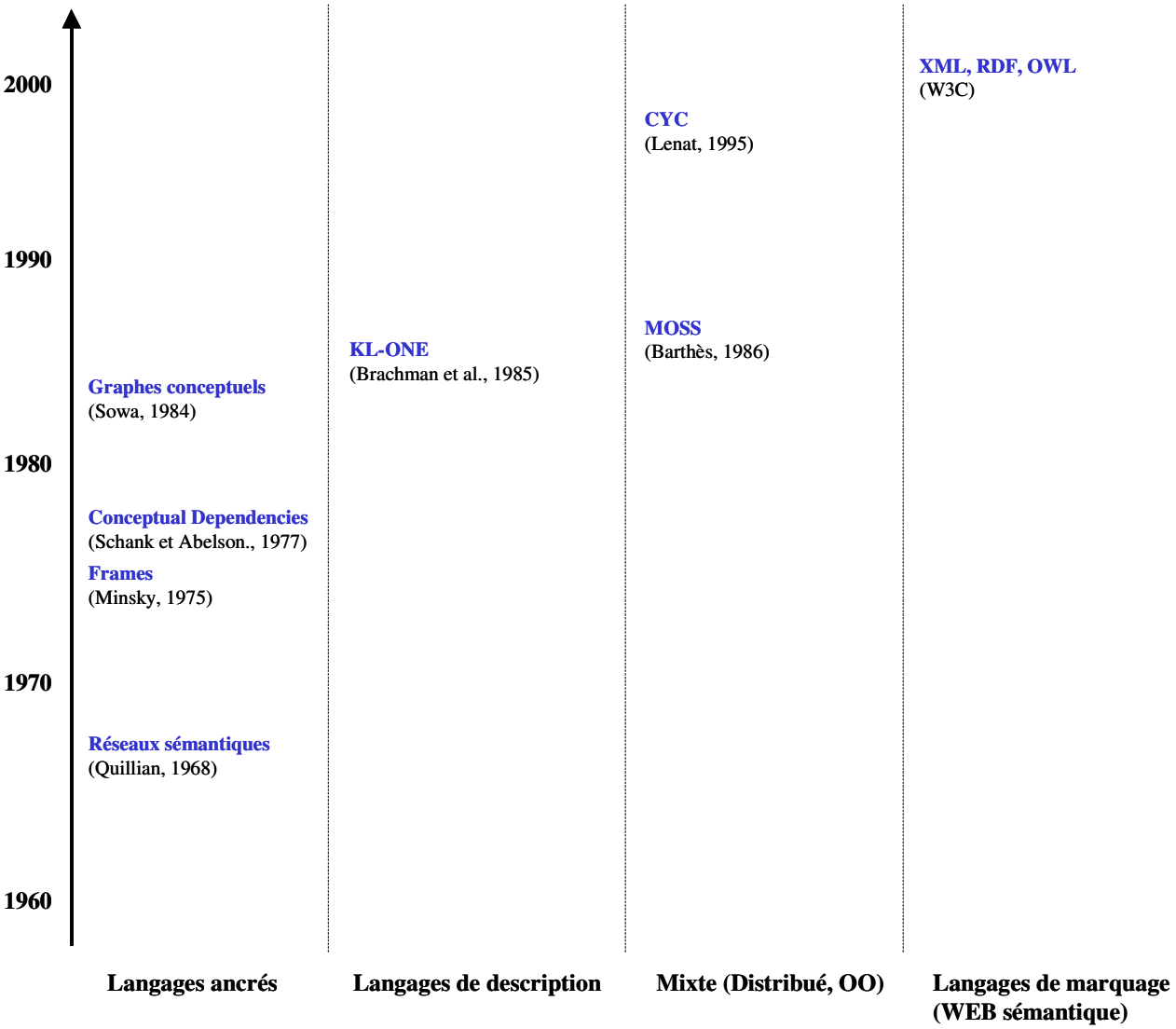


Figure 5 : Formalismes pour la représentation de connaissances.

Bien que d'autres formes soient utilisées, ce sont les ontologies qui ont ouvert un nouveau chapitre dans ce domaine. Dans un sens large, nous pouvons adopter pour la notion d'ontologie la caractérisation suivante (Uschold, 1998) :

« Une ontologie peut prendre différentes formes, mais elle inclura nécessairement un *vocabulaire* de termes et une *spécification* de leur signification. Cette dernière inclut des définitions et une indication de la façon dont les concepts sont reliés entre eux, les liens imposant collectivement une structure sur le domaine et contraignant les interprétations possibles des termes. »

Les buts principaux d'une ontologie sont de permettre le partage et la réutilisation des connaissances. Les travaux de Gruber (1993) et de Guarino (1996) sont les déclencheurs récents[1] de l'utilisation d'ontologies en informatique, vu qu'au départ les ontologies sont nées dans la branche de la philosophie qui traite de la nature et de l'organisation du monde.

En termes pragmatiques nous allons utiliser la définition suivante :

« Une ontologie est une description explicite et formelle des *concepts* dans un domaine du discours, contenant des *propriétés* (des attributs et/ou des relations) de chaque concept. »

À partir de la définition d'une ontologie, nous pouvons créer des instances qui contiennent des informations particulières qui représentent un état spécifique du domaine couvert par l'ontologie. Prenons l'ontologie présentée dans (Gennari, 2002) qui décrit plusieurs types de vins, leurs producteurs et les types différents de cépages qu'ils contiennent. L'ontologie décrit le concept de vin, qui représente tous les vins. Les vins spécifiques sont des instances de ce concept. Le vin de Bordeaux est une instance de la classe des vins.

Une ontologie contenant un ensemble d'instances constituera une base de connaissances. Une telle base de connaissances sera utile pour tous nos agents.

Les ontologies sont de plus en plus utilisées dans de nombreux domaines. Dans le domaine de l'aide à l'utilisateur, elles jouent un rôle essentiel, allant de la simulation de tremblement de terre (Kim et al., 2004) à la représentation du profil de l'utilisateur dans un système de recommandation (Middleton et al., 2001). La Figure 6 illustre une ontologie utilisée dans le projet *BirdQuest* (Eriksson, 2003) comme support d'un système de dialogue en langage naturel et aussi pour améliorer l'extraction d'informations d'une base de documents. Dans cet exemple, les concepteurs de l'ontologie ont choisi de représenter les concepts du domaine du point de vue de l'utilisateur et du point de vue des sources d'information.

[1] Goldstein et Roberts (1977) ont été parmi les premiers à construire et utiliser une ontologie en informatique. Ils ont développé une ontologie qui décrivait les concepts liés à la gestion d'un emploi du temps (dans le système Nudge).

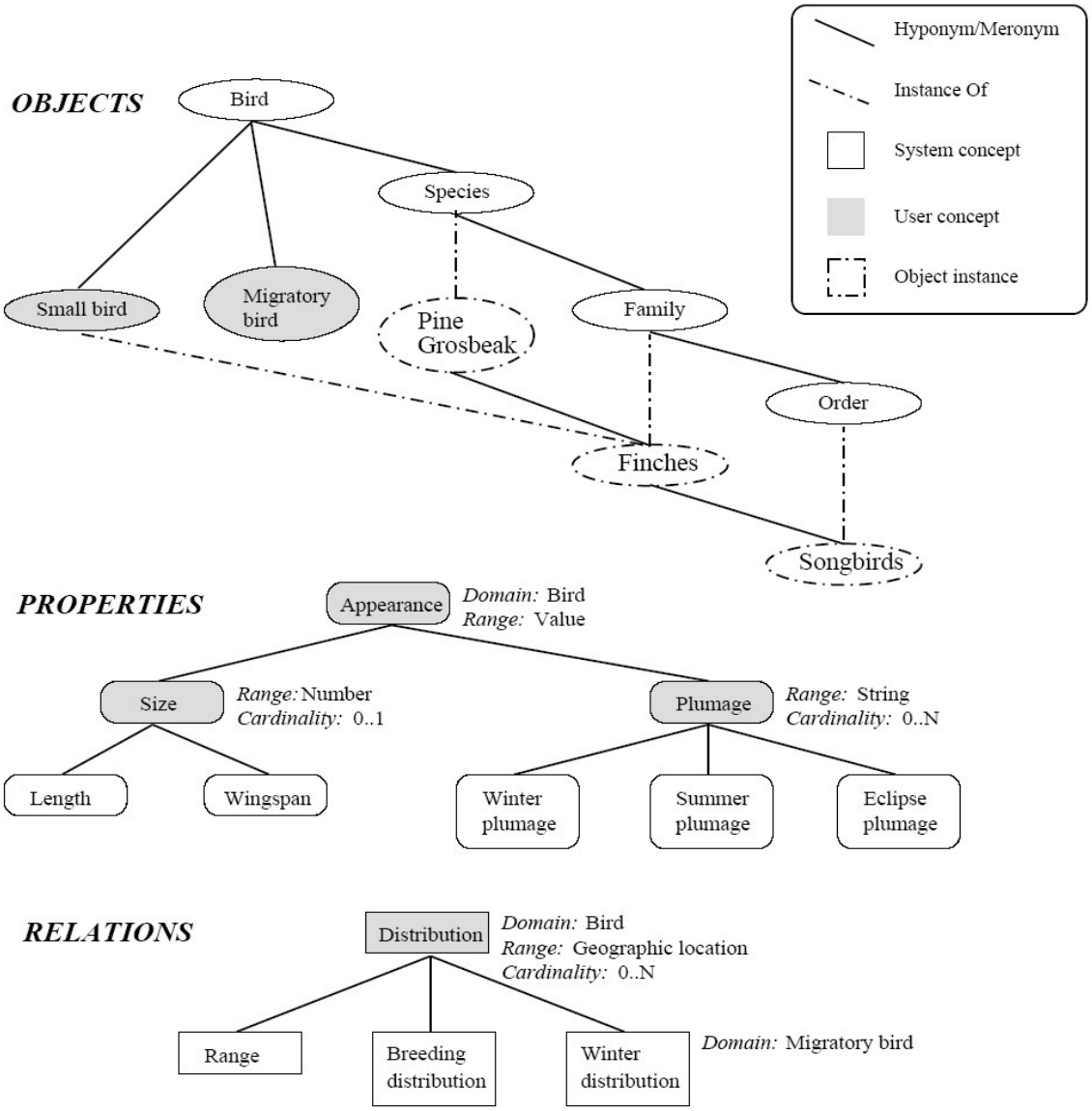


Figure 6 : Exemple d'une ontologie : *BirdQuest* (extrait de (Eriksson, 2003)).

Dans notre travail, nous allons nous appuyer sur des ontologies pour :

- interpréter le contexte des messages envoyés par d'autres agents ou par l'utilisateur ;
- garder une représentation informatique des connaissances utiles au moment de l'inférence.

Les ontologies auront un rôle déterminant pour le bon déroulement de notre système d'interprétation sémantique et de contrôle du dialogue. D'ailleurs, dans le Chapitre 4, nous présentons une étude sur la conception d'ontologies spécialement conçues pour le traitement de dialogues avec l'utilisateur.

Après la notion d'agent, nous pouvons maintenant introduire les systèmes multi-agents (SMA).

2. Les systèmes multi-agents

L'idée générale derrière les SMA est que deux ou plusieurs systèmes peuvent obtenir de meilleurs résultats en travaillant de façon coopérative plutôt qu'en travaillant isolément. Pour Sycara (1998) : « les outils les plus puissants pour manipuler la complexité sont la modularité et l'abstraction ».

Dans ce texte nous allons adopter la définition suivante d'un SMA :

« Un système informatique composé de plusieurs agents qui coopèrent mutuellement pour réaliser une tâche, qui ne pourrait pas, ou pas facilement, être réalisée autrement. »

Nous pouvons lister quatre caractéristiques qui distinguent un SMA (Sycara, 1998) :

1. chaque agent ne peut pas ou n'a pas d'information suffisante pour résoudre le problème en sa totalité ;
2. le contrôle n'est pas centralisé ;
3. les données pour la résolution du problème sont décentralisées ;
4. le processus de résolution est asynchrone.

Dans un SMA, les agents sont indépendants, ils partagent l'environnement, peuvent être en concurrence pour les mêmes ressources (telles que le temps, l'espace et les machines) et peuvent coopérer. Même s'ils n'ont pas été écrits dans le même langage, ils ont la capacité de formater et de transmettre leurs besoins en utilisant une structure de représentation commune.

Dans ce cadre, un agent dans un SMA, a deux buts principaux :

1. exécuter ses tâches, et ;
2. aider d'autres agents autant que possible.

Par conséquent, les agents doivent coordonner leurs activités et coopérer, afin d'éviter la duplication des efforts (d'Inverno et Luck, 2004).

Pour réaliser les systèmes mult-agents plusieurs plates-formes sont proposées (voir (Mandiau et al., 2002) ou (Boissier et Guessoum, 2004)). Nous allons rapidement présenter la plate-forme développée dans notre laboratoire et utilisée pour l'implémentation de notre SMA.

2.1. La plate-forme OMAS

La plate-forme OMAS (*Open Multi-Agent System*) est le fruit du développement de plusieurs travaux dans de domaines complexes, comme par exemple, la conception en mécanique (Shen et Barthès, 1996). OMAS est directement dérivé de l'architecture ouverte d'agents cognitifs indépendants appelée OSACA (Scalabrin et Barthès, 1993).

OMAS en sa dernière version (Barthès, 2002), développée en LISP, permet la création de deux types différents d'agents : les agents de service et les agents assistants personnels. Actuellement les agents sont tous connectés sur le même sous-réseau interne et utilisent le protocole UDP (*User Datagram Protocol*) pour communiquer.

L'agent de service est un agent qui, a priori, n'a pas d'interface direct avec l'utilisateur. Il fournit un type spécifique de service correspondant à des compétences qui lui ont été attribuées. Il fournit ses services à la communauté d'agents qui forment le SMA.

L'agent assistant est responsable de l'interface avec l'utilisateur. Son rôle est de comprendre et d'aider son maître[2]. Comment l'agent assistant interagit avec l'utilisateur et la façon dont on peut améliorer cette relation, est le centre d'intérêt de cette thèse et sera discuté en détail dans les chapitres suivants.

Dans OMAS chaque agent a une même structure présentée dans les paragraphes suivants.

2.1.1 L'agent générique

Les agents OMAS sont dérivés d'un modèle générique proposé par Ramos (2000) et présenté sur la Figure 7.

[2] Pour nous la référence *maître* est équivalente à l'utilisateur de l'agent assistant.

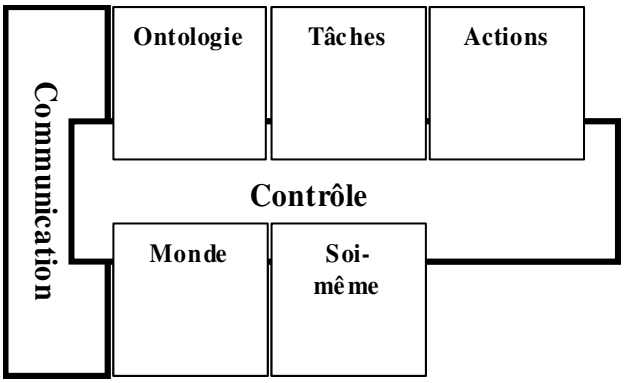


Figure 7 : L'agent générique OMAS.

L'agent est composé de plusieurs modules à « remplir » par le concepteur de l'application du SMA. Ce modèle a été repris par Enembreck (2003) qui a proposé quelques modifications pour la mise en place d'un agent assistant (voir section 2.2).

Monde

Chaque agent est inséré dans un environnement composé d'autres agents, avec lesquels il est « obligé » d'interagir. Pour gérer ces interactions, l'agent a besoin de créer et de maintenir une représentation interne des agents qui partagent avec lui le même environnement. Ramos suggère qu'il est utile de connaître les domaines d'intérêt et les compétences des agents qui sont dans le même groupe.

Soi-même

La représentation de soi contient la mémoire de l'agent, une description des compétences, et une représentation des buts à long terme. Enembreck a rajouté une méthode d'apprentissage automatique qui est capable de construire des généralisations à partir d'un ensemble de messages. Il donne un exemple dans lequel cette méthode peut être utilisée pour découvrir à qui l'agent doit demander un certain service ou encore prédire quel sera le résultat d'un service donné.

Ontologie

Pour assurer une sémantique commune à la terminologie utilisée pour exprimer les messages, le modèle propose l'utilisation d'une ontologie. L'ontologie est utilisée dans l'interprétation du contenu des messages reçus par l'agent et dans la représentation de ses compétences. Pour cela, une des possibilités est l'utilisation du système MOSS (Barthès, 1994). MOSS permet la construction d'un réseau sémantique contenant les concepts et ses propriétés, il

fournit un moteur pour raisonnement. La Figure 9 montre un agent de service chargé de la gestion d'une base de carnet d'adresses réalisé avec MOSS.

Tâches

Les tâches représentent l'état courant de l'agent, ce qu'il est en train d'exécuter. Ces tâches peuvent engendrer des sous-tâches qui sont contrôlées par OMAS.

Actions

Les actions correspondent en fait aux compétences individuelles de chaque agent. Ces compétences sont réalisées sous forme de fonctions. Dans OMAS, une fonction est déclanchée si son nom et ses arguments sont explicitement déclarés dans le contenu d'un message du type *request*, *inform* ou *grant* qu'un agent donné vient de recevoir. Les compétences peuvent être statiques, quand l'agent est capable de les exécuter sans avoir besoin d'aide, ou dynamiques, quand l'agent doit partager l'exécution entre plusieurs agents.

Contrôle et Communication

Ces modules sont exclusivement accessibles par OMAS. Le module de communication permet à l'agent de formater ses messages et de les envoyer en utilisant les mécanismes déjà présents dans OMAS, à savoir : FIPA ACL et un langage propre, comme langage de communication et CONTRACT-NET et un protocole de base, comme protocole de communication[3]. Le CONTRACT-NET est un protocole de communication basé sur l'offre de contrats. Son fonctionnement est illustré sur la Figure 8.

[3] Noter dans la Figure 9, l'indication du contenu d'un message reçu (M-36) et d'un message envoyé (M-68).

- Un agent demandeur annonce un contrat aux autres agents ;

- Les autres agents évaluent le contrat et, s'ils sont concernés, ils soumettent une offre ;

- L'agent demandeur évalue les offres et choisit la plus appropriée(s) ;

- Le contractant et le(s) contracté(s) communiquent directement.

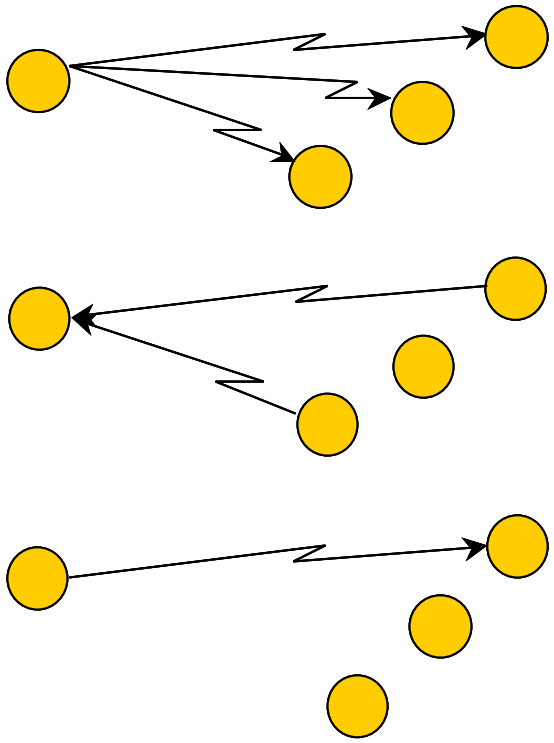


Figure 8 : Le fonctionnement du protocole CONTRACT-NET.

Pour finir, le module de contrôle pilote tous les autres modules.

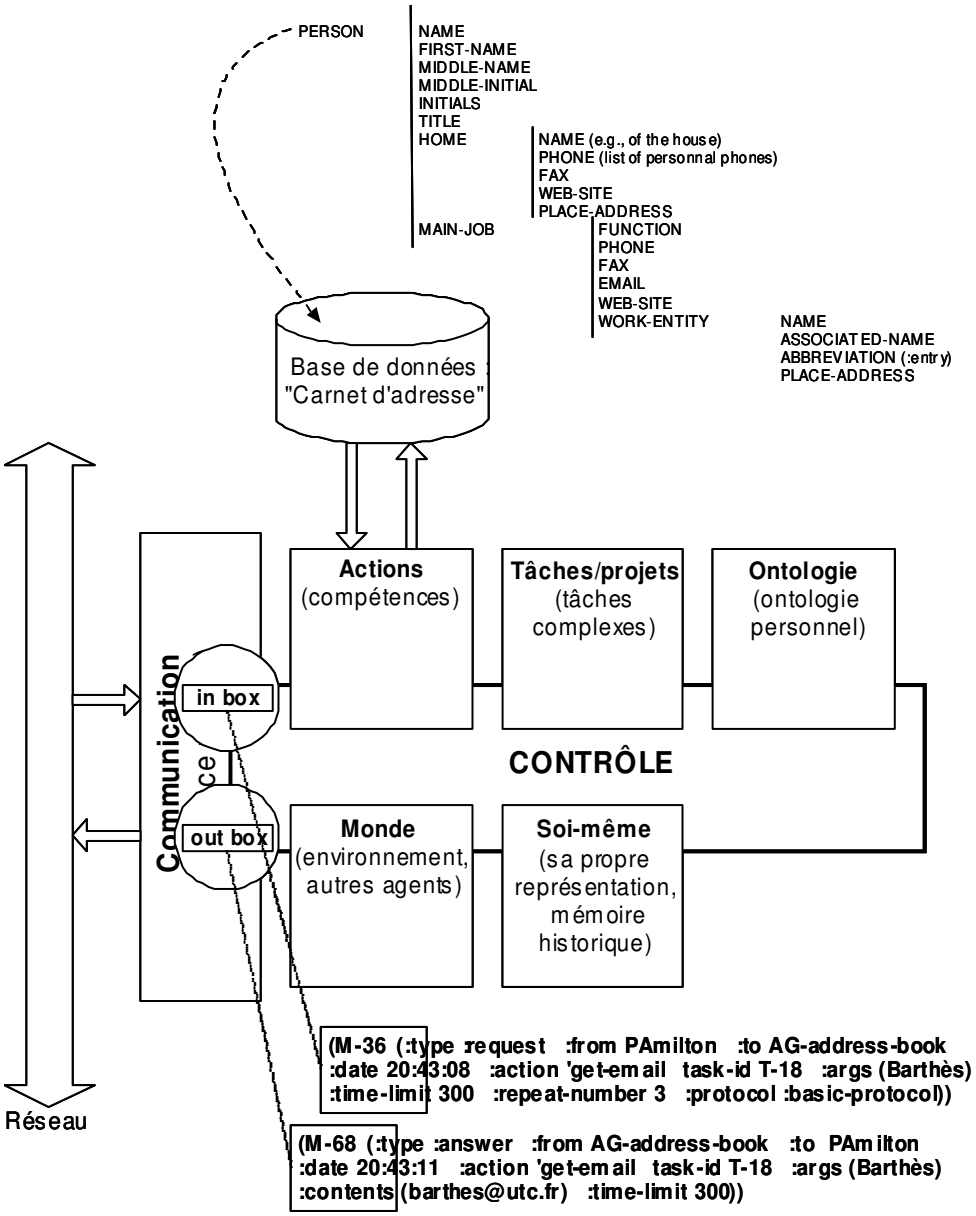


Figure 9 : L'agent de service chargé de la gestion de la base *carnet d'adresses*, extrait de (Ramos, 2000).

Avec cette structure minimale nous pouvons construire des agents comme celui qui est présenté sur la Figure 9. Par contre, ce modèle générique manque encore de certains éléments pour nous permettre de réaliser un agent assistant personnel. Comme l'agent assistant personnel est au cœur de notre travail, nous allons lui porter une attention particulière dans la section suivante.

2.2. L'agent assistant personnel

Avant d'être capable de définir ce qu'est un agent assistant personnel arrêtons-nous sur l'environnent dans lequel se trouve l'utilisateur potentiel de notre agent. Il s'agit d'un

environnement professionnel où les individus effectuent des activités, de plus en plus complexes, dans un délai de plus en plus court. Notre rôle est de mettre l'accent sur ces contraintes en proposant un système capable de rendre les activités de l'utilisateur plus aisées. Ainsi, dans un premier temps, nous excluons toute application de l'agent assistant à but non professionnel ou non liée directement au travail quotidien de l'utilisateur.

Dans ce contexte nous définissions un agent assistant personnel comme étant :

« Un agent qui assiste son maître (utilisateur) afin de diminuer sa charge de travail pendant la réalisation de ses activités professionnelles nécessitant une interaction avec l'ordinateur. »

$$\sum \left(\begin{array}{c} \textit{assister l'utilisateur} \\ + \\ \textit{réduire la charge de travail} \\ + \\ \textit{faciliter l'interaction} \end{array} \right) =$$

Les idées clés de cette définition sont : assister l'utilisateur, réduire la charge de travail et faciliter l'interaction avec l'ordinateur.

Tout au long de cette thèse, nous avons cherché à ce que la somme de ces trois idées soit la conséquence directe de l'application de notre travail.

Nous croyons que l'utilisation des agents assistants pour interfacer des humains aux systèmes informatiques peut améliorer la qualité de l'interaction, particulièrement quand l'utilisateur est impliqué dans plusieurs tâches simultanées (par exemple : rédaction d'un rapport technique, recherche d'informations sur le WEB, écriture d'un courriel, etc.).

Selon Maes (1994) un agent assistant peut assister l'utilisateur de différentes manières :

- en masquant la complexité des tâches difficiles ;
- en accomplissant des tâches au nom de l'utilisateur ;

- en formant l'utilisateur ;
- en aidant différents utilisateurs à collaborer ;
- en surveillant des événements et des procédures.

L'agent assistant peut être autonome pour « remplacer » son maître dans la prise de certaines décisions et réactif lorsqu'il reçoit des stimuli à partir de l'environnement. Il va aussi agir en fonction des demandes de l'utilisateur en s'appuyant, dans tous les cas, sur des agents de services dans son groupe ou ailleurs dans le réseau.

Dans le chapitre sur l'expérimentation nous reviendrons sur la question de l'organisation d'un SMA avec OMAS.

2.2.1 Evolution de l'agent assistant dans OMAS

Nous pouvons maintenant porter notre attention sur la réalisation d'un agent assistant à partir du modèle générique présenté dans la section 2.1.1. Cette implémentation fut l'objet d'une étude théorique réalisée par Ramos. Pour designer l'agent assistant, Ramos a étudié plusieurs architectures différentes, en tenant compte surtout des aspects liés à l'interaction. Il a proposé une architecture censée être appropriée à l'implémentation d'agents assistants personnels. À partir du modèle générique, il a rajouté deux nouveaux modules (Figure 10) responsables des relations avec l'utilisateur : le module *maître* et l'*interface utilisateur*.

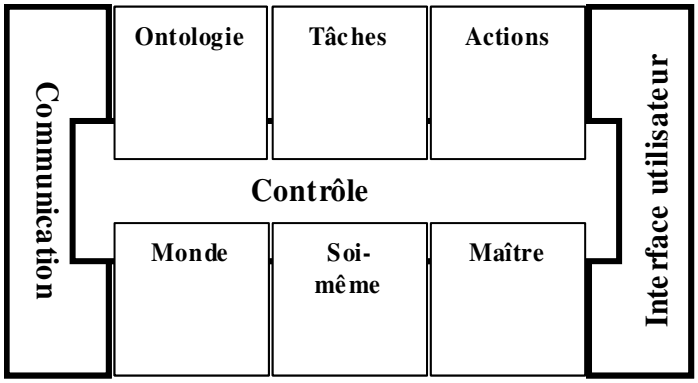


Figure 10 : L'architecture générique d'un agent assistant personnel selon Ramos.

Maître

Ce module doit décrire l'utilisateur en fonction de ses préférences ; de ses habitudes. Il peut contenir des informations personnelles telles que le nom ou l'adresse électronique de

l'utilisateur, ou même ses centres d'intérêts. Ceci sera utile pour une recherche d'informations sur le WEB par exemple (Enembreck et Barthès, 2003).

Interface utilisateur

C'est, sans aucun doute, le module le plus complexe du modèle. Son objectif est de coordonner l'interaction entre l'utilisateur et l'agent à travers un système de dialogue. Ce système de dialogue ne comporte pas un modèle classique capable de gérer des échanges en langage naturel (tel que celui qui sera présenté dans le Chapitre 4) mais plutôt une solution intermédiaire, contenant une bibliothèque d'actions possibles, définies au préalable et déclenchées à partir de mots clés trouvés dans des « messages » tapés par l'utilisateur, comme montré ci-dessous (extrait de (Ramos, 2000)):

Agent: Hello! What can I do for you?
Maître: Please give me the number of "Kassel".
Agent: I found de following information:
Kassel, Gilles : 0344 234466
Do you want to keep that?
Maître: No.
Agent: What else can I do for you?

En s'appuyant sur l'ontologie, le système trouve un agent de service capable de répondre à une requête de l'utilisateur. Dans l'entrée « Please give me the number of Kassel », *number* est synonyme de *téléphone* et *Kassel* le point d'entrée utilisé pour la recherche d'information dans la base de données.

L'interprétation des entrées de l'utilisateur est faite sans l'utilisation des structures linguistiques tel que les grammaires par exemple. Cela peut poser des problèmes quand les expressions données sont plus complexes, par exemple : comment traiter de multiples demandes dans une seule requête (« Find and open the report on agents ») ou comment traiter les modificateurs applicables aux objets trouvés dans l'expression (« List the *latest* report on agents »). Le système de dialogue proposé est aussi limité. Il ne peut pas traiter plusieurs demandes simultanées et sa capacité d'aide à l'utilisateur est restreinte. Comme mécanisme de dialogue il utilise les graphes d'états, directement codés dans l'interface de l'agent. Malheureusement, cette technique limite l'évolution du système, lors de l'ajout de nouvelles tâches. Nous allons décrire ce système de dialogue en plus de détails dans le Chapitre 4 dédié à l'architecture de notre interface.

Sans mettre en cause le modèle original, Enembreck a repris le travail de Ramos et a proposé quelques modifications et améliorations. Ces modifications ont été obtenues au fur et à mesure qu'une série d'applications ont été déployées. Une architecture résultant de l'agent assistant a été donc présentée et est illustrée sur la Figure 11. Le but principal été de concevoir un modèle d'agent adaptatif, dont une méthode propre d'apprentissage, et un nouveau système de dialogue. L'approche proposée par Enembreck réduit aussi la complexité de l'agent en distribuant ses compétences entre des agents de coordination (dits de staff).

Les modules d'ontologie, tâches et maître sont gérés par le système de dialogue. Les modèles du monde et de soi-même contiennent l'identification d'autres agents ainsi que leurs descriptions de services.

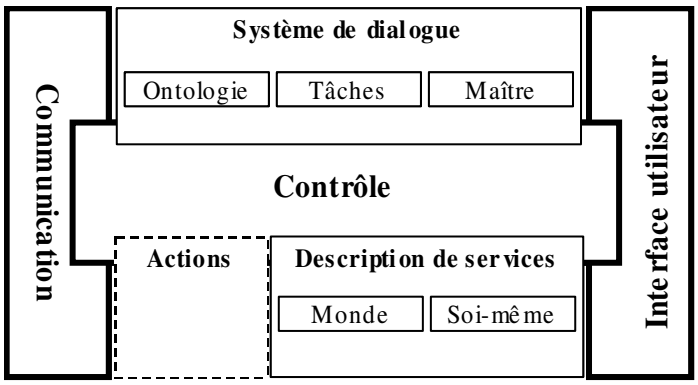


Figure 11 : Architecture modifié proposé par Enembreck.

Le nouveau modèle est capable de traiter des expressions en langage naturel plus complexes et comporte un système de dialogue du type orienté tâches. Le moteur de dialogue a été placé dans le module de contrôle, mélangeant contrôle de l'agent et contrôle du dialogue. À notre avis, cette décision compromet la stabilité de l'agent car, ce module est un « déclencheur » de fonctions et doit rester libre le plus possible. Nous reviendrons sur ce point dans le Chapitre 4 qui présente l'interface conversationnelle que nous avons conçue.

Puisque nous parlons des aspects liés à l'interaction avec l'utilisateur, il est temps d'étudier les types d'interfaces utilisées au cours du développement d'un agent assistant.

3. Les interfaces utilisées dans les agent assistant

Le domaine de l'interaction homme-machine a évolué énormément grâces aux avances technologiques. Si dans un premier temps les limitations physiques imposaient l'utilisation des

interfaces purement textuelles ou graphiques, aujourd'hui, la recherche s'est concentrée sur comment faciliter les échanges d'information avec l'utilisateur humain, en s'adaptant à ses besoins et à ses capacités. Les interfaces ont la possibilité d'utiliser d'autres modes de communication que le clavier, écran et souris, comme par exemple, la parole ou les gestes.

Dans le domaine des systèmes d'assistance intelligente, nous trouvons d'abord plusieurs applications utilisant les interfaces graphiques classiques, avec des composants comme les boutons et les menus. C'est le cas du système MAIS (*Multi-agent Based Internet Search*) (Enembreck, 2003) montré sur la Figure 12.

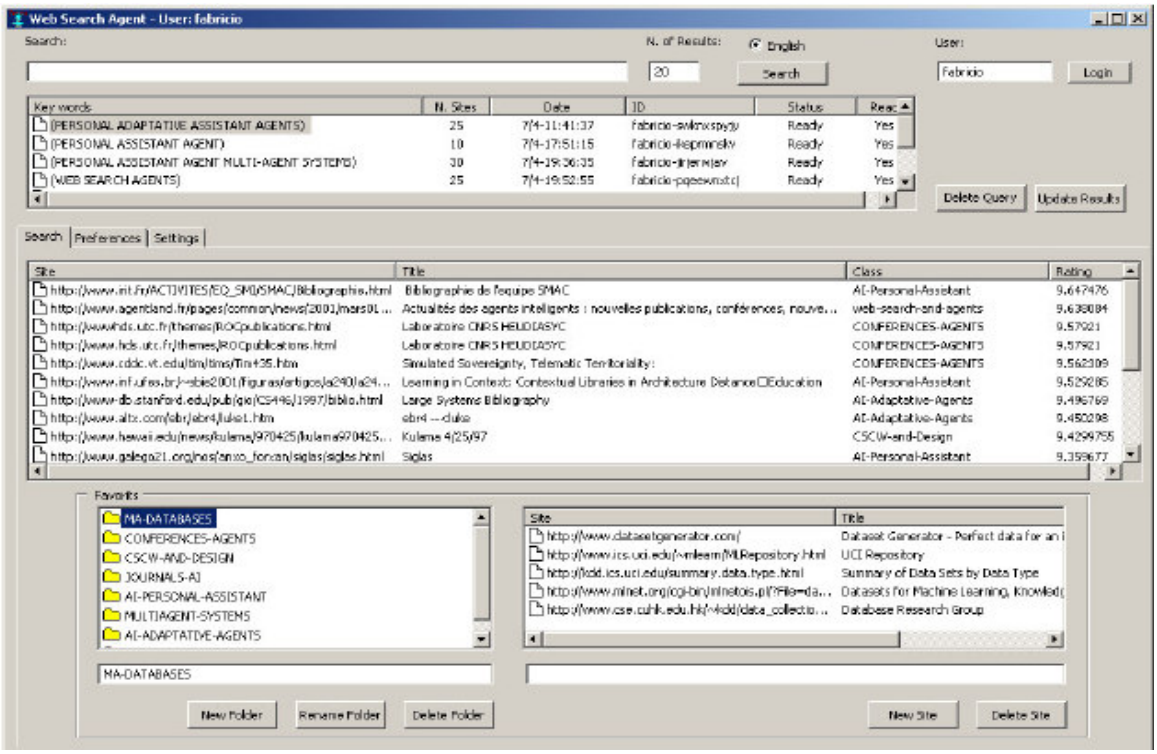


Figure 12 : Fenêtre de l'agent assistant du système MAIS.

Il s'agit d'un système de recherche d'information sur le WEB capable de modéliser les préférences de l'utilisateur et de les prendre en compte au cours de la recherche. Bien évidemment, c'est un exemple d'application où l'interface graphique est appropriée, car l'intervention de l'utilisateur est minimale, limitée à quelques mots clés nécessaires pour déclancher une requête.

Un autre exemple d'interface purement graphique est illustré par la Figure 13. Il provient du système CAT (Kim et al, 2004). Le système CAT (*Composition Analysis Tool*) est capable d'aider l'utilisateur au cours de la composition d'un flux de travaux dans le domaine de la simulation d'un tremblement de terre. Cette interface est basée sur le WEB, entraînant

l'utilisation d'un navigateur (*browser*). L'avantage principal réside dans sa portabilité. En revanche, le comportement de l'interface est plus instable à cause de la disparité technique entre les navigateurs disponibles actuellement.

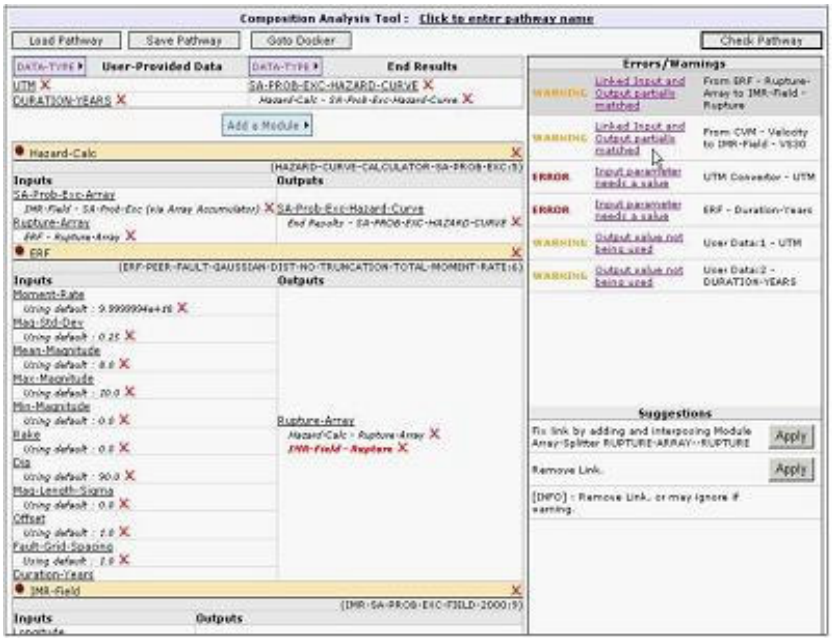


Figure 13 : L'interface du système CAT.

L'agent assistant personnel conçu par Tacla (2003) permet la construction coopérative de mémoires de projet à des fins de capitalisation des connaissances issues des échanges de messages et des activités quotidiennes entre les participants d'un groupe de travail. L'interface de l'agent montrée sur la Figure 14, comporte une partie (supérieure gauche) où l'utilisateur doit entrer de commandes textuelles relativement simples comme *SAVE file* ou *SEND text to*. Le concepteur a voulu faciliter l'interaction de l'utilisateur en créant un ensemble de commandes textuelles interprétées par un analyseur syntaxique. Cette application peut être un excellent terrain de test de notre architecture. Le pilotage de l'interface peut être remplacé par un système conversationnel comme celui que nous proposons, où plusieurs tâches décrites sous forme d'une ontologie de tâches remplacent les commands disponibles.

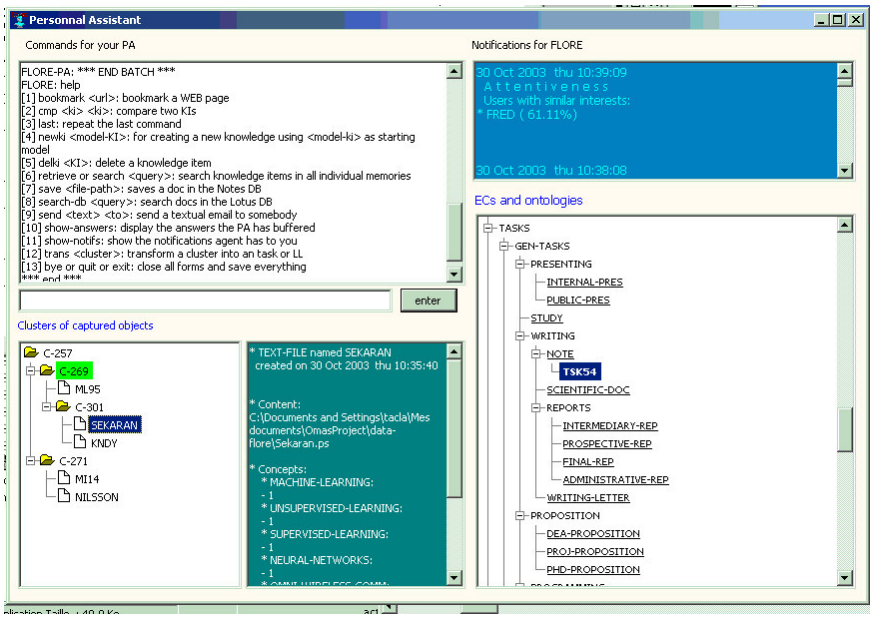


Figure 14 : L'interface de l'agent assistant avec le composant d'entrée des expressions textuelles.

Les interfaces évoluent aussi en fonction de la mobilité de l'utilisateur. Les nouveaux dispositifs portables, comme par exemple les ordinateurs de poche (PDA - *Personal Digital Assistant*) imposent des contraints supplémentaires aux concepteurs d'interfaces. Nichols et Myers (2005) proposent le système PUC (*Personal Universal Controller*) capable de générer de façon automatique des interfaces pour ce genre d'appareil. Le système produit l'interface à partir d'une description abstraite et d'un modèle du dispositif. L'application montrée sur la Figure 15 est un outil pour écouter de la musique.

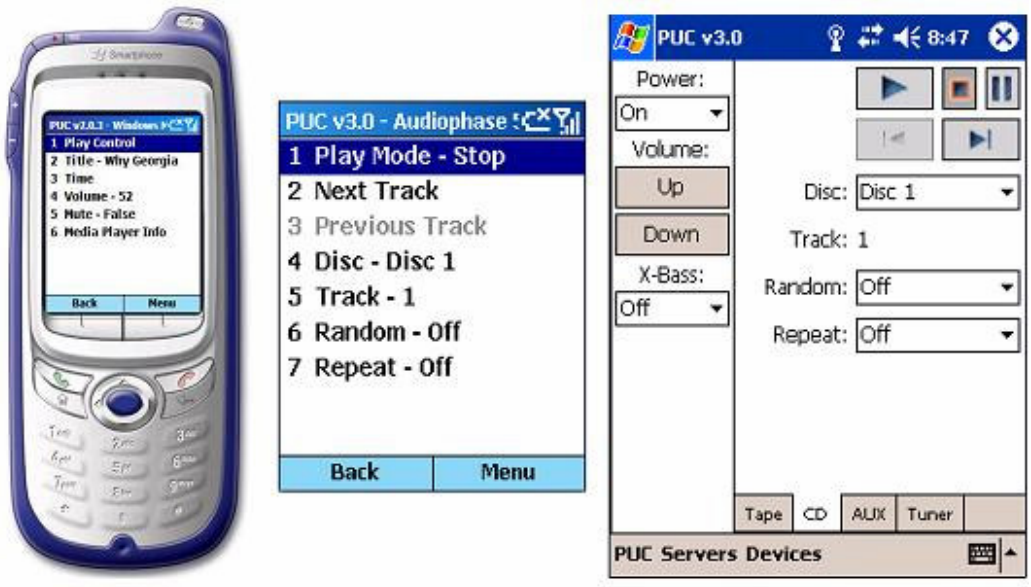


Figure 15 : Exemples d'interfaces générées par le système PUC.

Dans cette même direction, Quesada et al. (2001) et Yates et al (2003) proposent des interfaces multimodales pour les appareils électroménagers.

Une autre direction de recherche porte sur les agents conversationnels animés (*Embodied Conversationnal Agents*). Il s'agit ici de compléter l'interface graphique usuelle avec des personnages virtuels utilisant des signes non-verbaux qui ont un rôle dans la communication humaine (regard, expressions faciales, gestes de la main, postures). Le domaine applicatif des agents conversationnels animés (ACA) est celui des personnages virtuels interactifs, placés dans des environnements médiatisés, qui peuvent jouer trois rôles principaux (Sansonnet, 2004) :

- assistants : pour accueillir les utilisateurs et les aider à comprendre et à utiliser la structure et le fonctionnement d'applications et de services informatiques;
- partenaires des acteurs dans des environnements virtuels : partenaire ou adversaire de jeu, participant dans les systèmes de conception participative, membre d'une communauté mixte;
- tuteurs des apprenants : dans les environnements interactifs d'apprentissage humain (EIAH).

ADELE (Johnson et al. 2003) est un agent pédagogique animé qui a pour but de favoriser l'interaction entre les étudiants et les environnements d'apprentissage basés sur ordinateur. La Figure 16 illustre l'application de l'agent dans le cadre d'une formation médicale. L'étudiant interagit avec l'agent en cliquant sur des boutons ou par l'entrée d'expressions en langage naturel typés.

Figure 16 : L'agent animé ADELE.

Dans le cadre du projet européen IST-NICE (Martin et al. 2004), Martin et son équipe ont proposé des agents conversationnels bi-directionnellement multimodaux, où l'utilisateur peut combiner parole et gestes 2D via une tablette tactile pour interagir avec des personnages 2D. Le projet s'appuie sur des agents du type LEA (*Limsi Embodied Agents*) (Martin et al., 2002). Les agents LEA (illustrés sur la Figure 17) sont représentés sous formes de figurines 2D, animées par composition de fichiers graphiques dans le format GIF (110 images combinables pour produire des comportements variés). Sur le plan de la multimodalité, LEA peut exprimer une information de plusieurs manières (synthèse vocale, posture du corps, phrase écrite).

Figure 17 : L'agent LEA.

Dans le projet *LookOut* (Horvitz, 1999), Horvitz a utilisé les avatars Microrsoft™ pour animer une application capable d'aider l'utilisateur à gérer son emploi du temp. La Figure 18 montre l'avatar « Genie » indiquant l'affichage d'une fenêtre d'informations. Nous ne sommes pas convaincus de son efficacité, surtout comme montre la Figure 18, où l'avatar affiche un message peu informatif et pas vraiment utile. D'ailleurs, ce genre de composants animés a été diffusé par Microsoft™ (le compagnon d'assistance « Microsoft Paperclip ») dans le logiciel Office et rapidement jugé inutile à cause de sa façon intrusive de se présenter (Graesser et al. 2003).

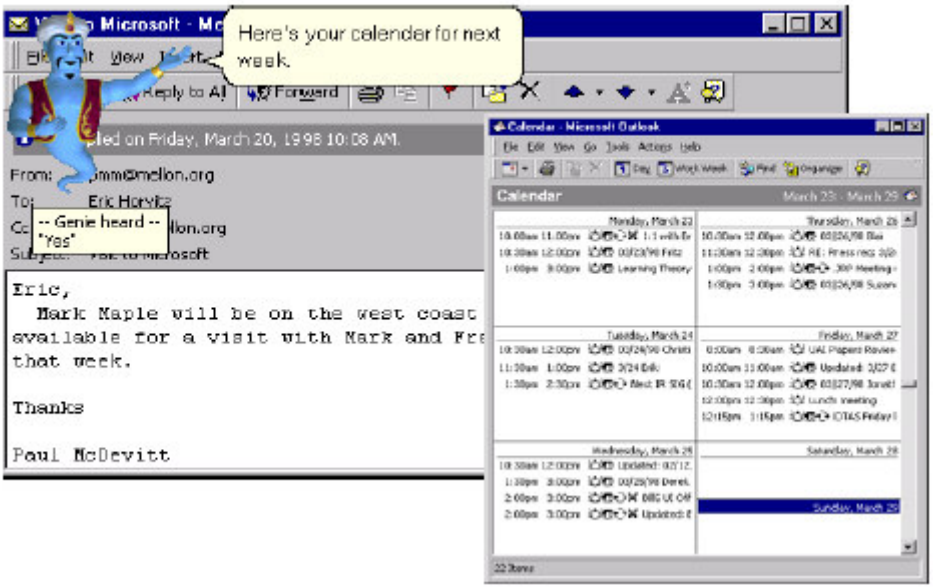


Figure 18 : L'avatar « Genie » dans *LookOut*.

Toutefois, les agents animés peuvent être une bonne alternative pour des applications comme les environnements interactifs d'apprentissage ou les jeux, où l'utilisateur doit interagir avec l'application sur une longue période de temps. Beun et al. (2003) ont mené une étude qui a montré que la présence d'agents animés influence de façon positive la mémorisation d'information. Même si Graesser et collègues ont mené une étude similaire et ont trouvé un résultat différent, tous sont d'accord sur un point : il faut plus de recherche pour arriver à des résultats convaincants.

Pour finir, nous estimons que pour des applications professionnelles, où l'utilisateur passe moins de temps devant l'application, une attention particulière est nécessaire pour que l'aspect « ludique » du personnage virtuel ne discrédite pas l'ensemble du système. Krämer et al. (2003) sont d'accord et ajoutent que les agents sont jugés amusants mais perturbent la performance globale du système.

Dans la section suivante, nous allons apporter les éléments qui nous ont guidés au cours du choix de l'interface idéale pour notre agent assistant.

4. Quelle interface pour l'agent assistant personnel

La question que nous nous posons maintenant est : quelle interface choisir pour un agent assistant personnel ? En évoquant les principales caractéristiques de notre agent assistant personnel, nous pouvons recenser un certain nombre d'éléments intéressants pour répondre à cette question.

Comme nous allons présenter dans le Chapitre 4, notre approche est basée sur le modèle d'un assistant qui seconde un travailleur intellectuel, c'est-à-dire quelqu'un qui utilise de façon intense ses capacités intellectuelles dans son travail, comme un ingénieur, un chercheur, un directeur d'entreprise, un avocat, etc. Cette assistance se caractérise par une aide en continu sur quelques tâches quotidiennes. Ramos a défini la relation entre utilisateur et son assistant en fonction de ces concepts d'assistance :

- la proximité : un couplage fort entre utilisateur et assistant est fondamental pour une bonne adaptation de l'assistant à son utilisateur. Une assistance effective est toujours basée sur l'identité entre les deux parties et sur un certain degré de connaissance partagée (habitudes, préférences, jargon commun, etc.) ;
- une forme de communication efficace : la communication est à la base d'un processus de collaboration surtout dans la relation utilisateur/assistant ;
- la distribution de l'exécution des tâches : la distribution de l'exécution des tâches spécifiques à d'autres agents, va spécialiser l'assistant uniquement dans la tâche d'assistance à son utilisateur, c'est-à-dire quand et comment l'aider.

Étant donné que notre agent doit respecter les engagements décrits ci-dessus, nous avons adopté l'interaction multimodale en langage naturel comme les composants déterminants de notre choix. Lemeunier (2000) soutient que : « … *au-delà de ce simple constat, l'usage des langues naturelles nous semble plus intéressant pour ce qui concerne l'adaptation de la machine au fonctionnement cognitif des utilisateurs* ». Sur les interfaces purement graphiques, il ajoute: « *... l'utilisateur doit se livrer à un travail d'auto-interprétation avant et pendant l'interaction, pour savoir ce qu'il doit faire* ».

Nous croyons qu'en laissant à la machine le travail d'interprétation des énoncés de l'utilisateur et la reconnaissance de ses intentions, la charge cognitive de l'utilisateur diminue (Paraiso et al., 2004). Des exemples d'applications qui vont dans cette direction ne manquent pas : comme Allen et collègues (1996) et Sadek (1996) qui ont proposé des systèmes de vente de titres de transport ou comme Rickel et collègues (2001) qui ont travaillé dans les systèmes tuteurs. Dans l'architecture de l'agent assistant proposée par la FIPA (FIPA, 2001) le mode d'interaction choisi est aussi une interface multimodale, où l'utilisateur peut dialoguer avec son assistant.

Nous pouvons facilement trouver plusieurs exemples pour conforter notre choix. Un exemple classique concerne la recherche de vol dans un système automatique de vente de tickets aériens. Une entrée typique est montrée ci-dessous :

USER: Show me all flights from Boston to Denver for tomorrow.

La complexité de cet énoncé est relativement restreinte et un résultat similaire pourrait être obtenu par une interface graphique « équivalente » comme celle montré sur la Figure 19, proche du remplissage d'un formulaire.

Figure 19 : L'interface graphique « équivalente ».

Le système *Mercury Flight Reservation System* (Seneff et Polifroni, 2000) est capable de traiter ce genre de problème. Le système *Mercury* permet l'accès, par téléphone, à une base de données de vols ainsi que la planification d'itinéraires entre plusieurs aéroports dans le monde. Les énoncés sont traités et interprétés par un système de dialogue en langage naturel.

Bien évidemment, le système *Mercury* est capable de traiter des énoncés beaucoup plus compliqués, comme par exemple :

USER: Show me all flights non-stop from Boston to Denver for tomorrow before noon.

Dans ce cas précis nous pouvons nous demander : combien de champs d'un formulaire classique l'utilisateur doit remplir pour obtenir un résultat équivalent ?

Comme conséquence, notre hypothèse initiale est que :

« L'agent assistant est l'interface privilégiée avec l'utilisateur, car il peut connaître ses préférences et il a la capacité de communiquer avec lui en langage naturel parlé. »

Pour mettre en place notre hypothèse, nous allons utiliser les interfaces conversationnelles telles que décrites par Kölzer (1999). Une interface conversationnelle laisse les utilisateurs dire ce qu'ils veulent, dans leurs propres termes, exactement comme ils le feraient en s'adressant à

une autre personne. Ainsi, l'utilisateur pourra rester concentré sur ses activités principales et, de temps en temps, parler à son agent assistant pour lui demander un service ou faire une clarification.

Par rapport à une interface graphique traditionnelle, généralement figée (puisque tout est prévu à l'avance), une interface conversationnelle n'est pas figée, de plus, elle se « construit » au cours du dialogue.

Si l'interface conversationnelle apporte à l'agent assistant une flexibilité majeure, ce dernier peut aussi aider à améliorer la performance d'une telle interface. Zoe et Glass (2000) ont mené une étude qui a montré que le nombre d'énoncés[4] échangés entre l'utilisateur et une application dotée d'une interface conversationnelle, augmente en fonction de la complexité du domaine de l'application. La Figure 20 montre que, pour une application d'aide à la recherche d'un restaurant, le nombre d'échanges (*turn-taking*[5] en anglais) est inférieur au nombre d'échanges d'une application dans le domaine de la recherche d'un travail.

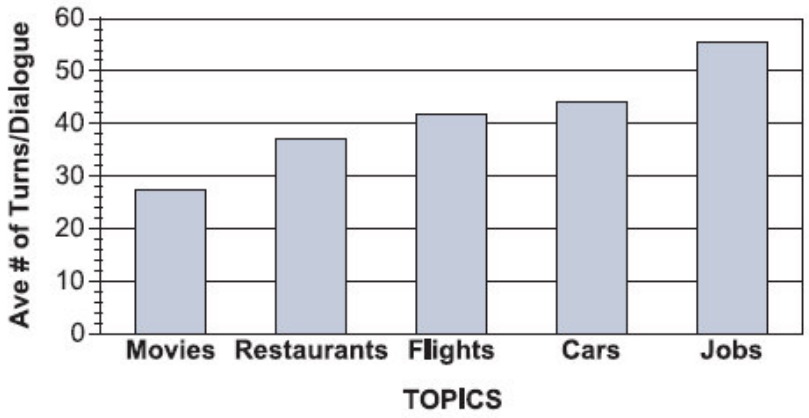


Figure 20 : Nombre d'échanges dans plusieurs domaines (extrait de (Zoe et Glass, 2000)).

Dans ce cas, comment notre agent assistant pourra-t-il aider à réduire le nombre d'énoncés de l'utilisateur, tout en obtenant les résultats attendus par ce dernier ? D'abord, rappelons que l'agent assistant possède une ontologie qui contient les informations du domaine et qu'il pourra l'utiliser pour agir au nom de l'utilisateur tout en respectant ses préférences. Ensuite, l'agent assistant enrichit ses connaissances au fur et à mesure des échanges avec l'utilisateur, grâce au mécanisme d'apprentissage automatique. Ainsi, il pourra « prédire » les demandes de son

[4] Un énoncé peut être un mot simple ou peut contenir plusieurs mots (une expression ou une phrase).

[5] *Turn-taking* peut être défini comme le changement de « main » entre deux ou plusieurs interlocuteurs au cours d'un dialogue.

maître, évitant les questions qui ont déjà reçu une réponse par exemple. Enfin, comme nous allons montrer avec plus de détail dans le Chapitre 4, en limitant l'espace des dialogues à ceux ne contenant que des actes de langage de type illocutoires, nous allons réduire les échanges nécessaires pour accomplir les tâches. Par exemple, nous allons restreindre les échanges du type *Acknowledge* (des expressions tel que « Merci » ou « Bon voyage »). Flammia (1998) a montré qu'en fonction du domaine ces échanges peuvent représenter jusqu'à 50% du total. Dans le domaine de la recherche de films, ils peuvent atteindre 30% des énoncés donnés par l'agent fournisseur d'informations.

Bien évidemment, notre approche est plus appropriée à une certaine classe d'applications. Ce sont des applications où :

- le domaine est limité et bien connu, comme par exemple la gestion de connaissances ou les environnements d'apprentissage basés sur l'ordinateur ;
- les actions d'utilisateur sont complexes et exigent une connaissance préalable du domaine ;
- l'utilisateur doit être guidé pour accomplir une tâche ;
- une interface graphique traditionnelle peut ennuyer l'utilisateur, à cause d'un grand nombre de composants tels que menus, sous-menus, boutons, etc. ;
- l'utilisateur doit « mémoriser » les actions passées pour piloter l'application (selon Larson (2003), la mémoire humaine à court terme peut retenir environ 7 actions passées pour la même tâche) ;
- un écran n'est pas disponible ;
- l'utilisateur a des limitations physiques, comme ses mains occupées ou un handicap moteur.

Dans le Chapitre 5, nous essaierons de montrer que la qualité de l'aide fournie par l'agent assistant est directement liée à la capacité d'interaction de l'agent. C'est à ce titre que nous allons utiliser les interfaces conversationnelles, car nous sommes convaincus qu'elles amélioreront la qualité de l'aide qu'un agent assistant peut offrir dans certaines situations spécifiques. Nous proposons donc une interface conversationnelle spécialement conçue pour les agents assistants permettant à l'utilisateur d'utiliser des expressions en langage naturel parlé.

La conception et la réalisation d'une telle interface implique l'intégration de plusieurs dispositifs, tels que : les systèmes de dialogue, les parseurs, les grammaires, les moteurs de reconnaissance et de synthèse de la parole, les ontologies entre autres. Le chapitre suivant présente une étude théorique approfondie sur ces composants.

5. Synthèse et conclusions de ce chapitre

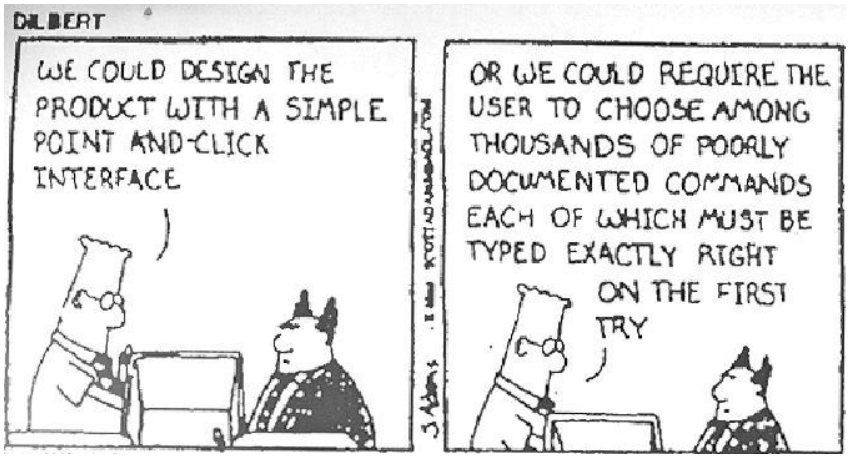
Dans ce chapitre, nous avons introduit les concepts de base sur les agents pour ensuite présenter les interfaces utilisateur généralement utilisées dans des agents assistants.

Nous avons présenté les caractéristiques fondamentales des agents, comme par exemple la communication inter-agents et les systèmes de représentation de connaissances. Nous avons défini les systèmes multi-agents et ensuite, nous avons montré le modèle d'agent dans la plate-forme OMAS et l'évolution du modèle vers l'agent assistant personnel.

Nous avons vu que l'interface utilisateur joue un rôle important dans le modèle de l'agent et que plusieurs types différents sont proposés. Nous avons défini l'interface idéale pour notre agent comme étant une interface conversationnelle, capable d'interagir avec l'utilisateur en utilisant le langage naturel parlé. Nous nous sommes appuyés sur l'idée que l'assistant seconde un travailleur intellectuel dans son travail quotidien professionnel.

Comme nous allons présenter dans les chapitres suivants, notre groupe de recherche a récemment expérimenté l'utilisation du langage naturel *tapé* comme forme d'entrée des informations. Nous voulons dans cette thèse aller plus avant, proposant la conception d'une nouvelle interface, où les entrées se font en langage *parlé*.

CHAPITRE 3



Les systèmes et les interfaces conversationnelles classiques

Dans ce chapitre, nous présentons les composants nécessaires pour la mise en œuvre d'une interface conversationnelle. Après l'étude effectuée dans le chapitre précèdent, nous avons défini les interfaces conversationnelles comme étant notre choix pour l'interface de l'agent assistant. La complexité des mécanismes d'interaction et de gestion des informations rend ce genre d'interface extrêmement compliquée. Comme nous avons pu le constater dans la section 3 du chapitre précédent, la littérature ne compte pas beaucoup d'agents assistants personnels équipés de ce type de facilité. D'ailleurs, la proposition d'une interface conversationnelle dédiée aux agents assistants reste inédite.

Ce chapitre débute par la présentation générale d'une interface conversationnelle, décrivant ses principales composantes et adoptant la stratégie de présentation de ces composants en fonction de leur place dans le flux de traitement des informations. Ainsi, nous allons présenter d'abord les aspects liés à la saisie de la parole pour ensuite progresser vers les mécanismes d'analyse sémantique et la gestion du dialogue. Cette étude va mettre l'accent sur les éléments plus sensibles, comme les grammaires, les systèmes de dialogue et la place des ontologies dans la gestion du dialogue, tout en préparant le lecteur à la compréhension de notre architecture, présentée dans le chapitre suivant.

1. Vision générale

Le but de ce chapitre est de présenter et d'examiner les composants fondamentaux d'un système conversationnel. Au fur et à mesure que ces composants seront présentés, nous allons constater que les systèmes conversationnels utilisés jusqu'ici sont, en réalité, inappropriés à un agent assistant. À ce jour et à notre connaissance, il n'existe pas d'architecture complète dédiée aux agents assistants telle que nous la proposons. Comme résultat, à la fin de ce chapitre, nous allons étendre le concept d'interface conversationnelle pour proposer un nouveau concept : « Interface Conversationnelle pour une Aide Intelligente – ICAI ». Donc, nous allons auparavant étudier les composants d'une interface conversationnelle classique.

Nous limitons nos investigations aux composants nécessaires pour mettre en place des systèmes pour des applications pratiques. Cela veut dire que, par exemple, dans le cadre des systèmes de dialogue, nous allons nous concentrer sur les systèmes du type orientés tâches et plans. Cela veut dire aussi que la plupart de nos exemples sont toujours liés à des domaines « pratiques et professionnels » (dont sont exclus les jeux par exemple). Pour finir, les composants linguistiques nécessaires pour mettre en œuvre l'interface conversationnelle de l'agent assistant ne sont pas nécessairement les mêmes, que pour un système qui fait le compte rendu d'une réunion enregistrée en audio (Niekrasz, 2005). Par conséquent et, en fonction de notre intérêt, nous allons plutôt développer les composants qui seront effectivement présents dans notre interface.

Avant de démarrer, une précision : nous allons distinguer une interface conversationnelle d'un système conversationnel par le fait que ce dernier est formé de plusieurs composants qui ne forment pas une application unique, c'est-à-dire, il est le résultat de l'exécution de plusieurs processus indépendants. L'interface conversationnelle elle, est formée de plusieurs modules, tous localisés (liés) dans une seule application, sous un seul contrôle.

Le principe de base d'une interface conversationnelle est qu'elle laisse les utilisateurs dire ce qu'ils veulent, dans leurs propres termes, exactement comme ils le feraient en s'adressant à une autre personne. Dans le chapitre précédent (section 4) nous avons présenté les principaux motifs qui nous ont permis de choisir les interfaces conversationnelles comme la meilleur solution pour l'interface de notre agent assistant. D'ailleurs, nous avons vu que l'agent assistant aussi ajoute de la valeur à ce type d'interface. Ce « mariage » sera mieux décrit dans le chapitre

suivant. Ce qui nous intéresse ici, est de recenser les composants qui forment une interface conversationnelle.

La Figure 21 présente une architecture générique typique d'une interface conversationnelle. Elle a trois niveaux : un première niveau de saisie et de synthèse de la parole, un deuxième niveau de traitement de la langue et de compréhension et un troisième niveau de gestion du dialogue. Éventuellement, la parole peut ne pas suffire. Dans ce cas, d'autres modalités d'interaction peuvent être ajoutées, comme des composants graphiques (pour, par exemple, afficher des plans cartographiques ou formater un grand tableau de données), ou même des capteurs de mouvements pour des applications dans le domaine de la réalité virtuelle ((Aubry et al., 2005) et (Chalon et David, 2004)).

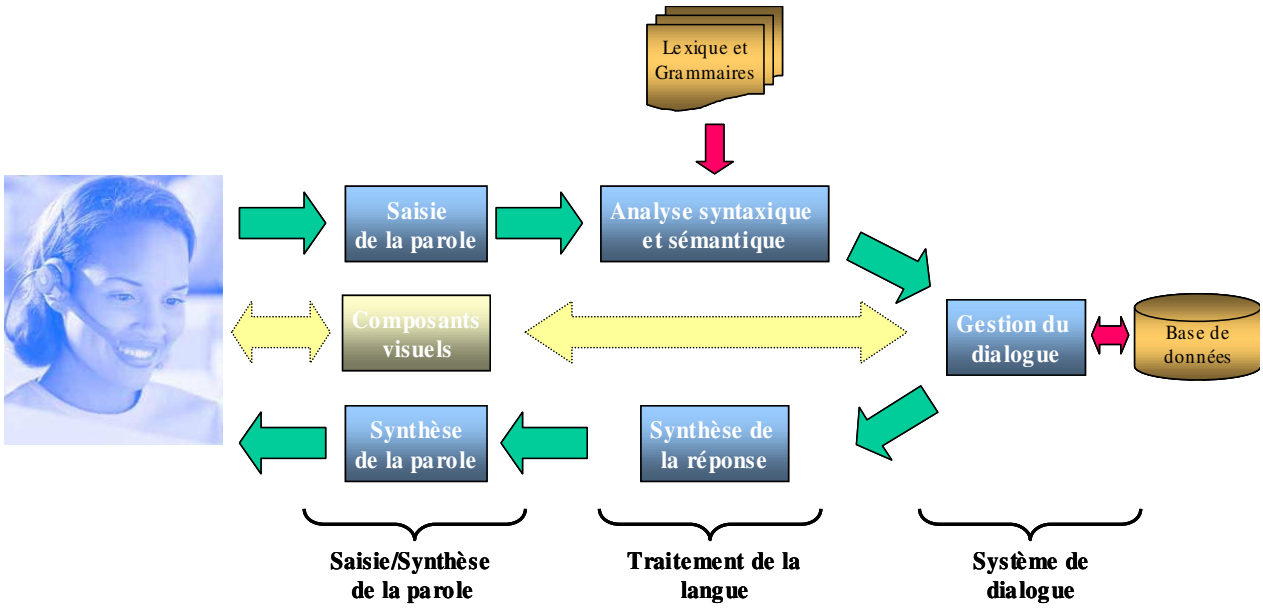


Figure 21 : Diagramme générique d'une interface conversationnelle typique.

La Figure 22 montre le flux de traitement d'un énoncé, qui détermine clairement la disposition des modules de la Figure 21. Pour des raisons d'efficacité, ces modules doivent être gérés par un seul contrôleur.

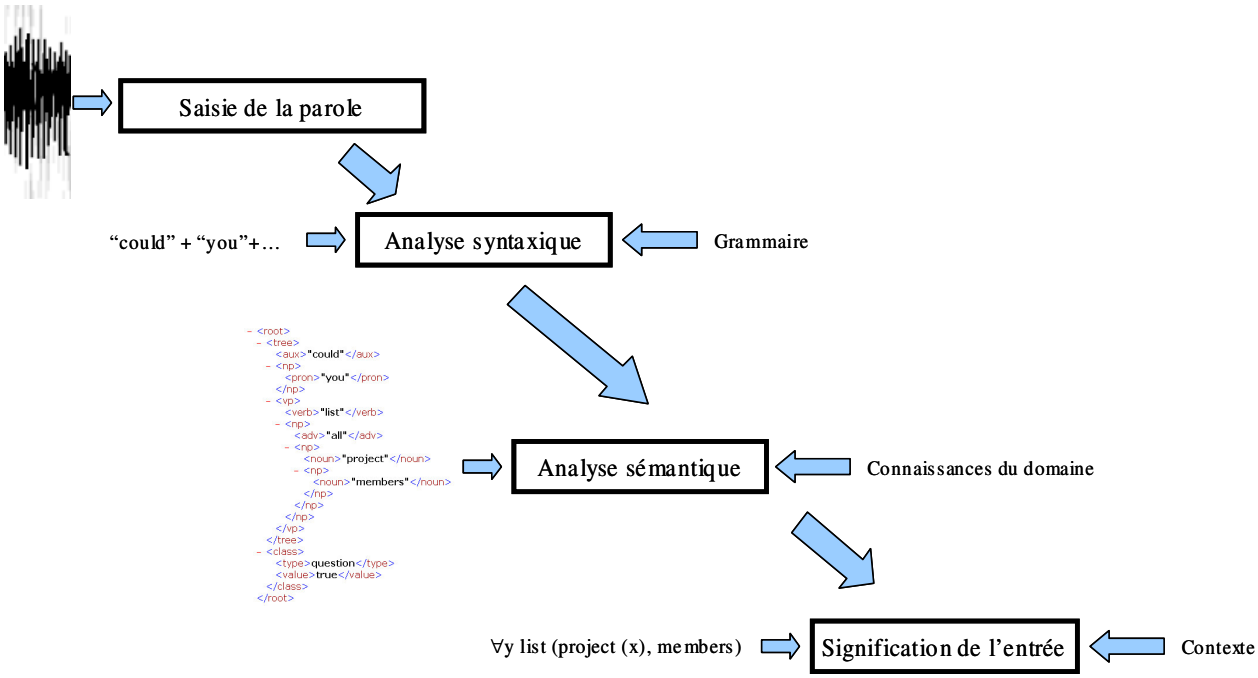


Figure 22 : Flux de traitement d'un énoncé.

Le processus de traitement consiste d'abord à saisir la parole et à la convertir en une chaîne de mots. Cette chaîne de mots (un énoncé) est, ensuite, confrontée à un ensemble de règles grammaticales qui classeront chaque mot selon sa classe grammaticale (verbe, nom, pronom, etc.). La troisième étape démarre le processus de compréhension de l'énoncé, lui donnant une représentation formelle (connue par le système de raisonnement). Finalement, à cette représentation sont rajoutés les éléments du contexte nécessaires à la compréhension par le système de dialogue. En cas de non-conformité, cette chaîne peut être rejetée à la sortie de chaque étape. Dans les sections suivantes, nous allons présenter ce processus en détail.

2. Saisie et synthèse de la parole

Le fonctionnement d'un système de saisie de la parole utilise des techniques mathématiques (le traitement du signal) et linguistiques (la linguistique informatique) pour fournir comme résultat la « numérisation » de ce qui a été dit par l'utilisateur. En sens inverse, les systèmes de synthèse de la parole permettent la prononciation d'un texte écrit numérisé, grâce un générateur de phonèmes. Ces deux domaines combinent les connaissances de plusieurs sciences : l'anatomie et les fonctions de l'appareil phonatoire et de l'oreille, les signaux émis par la parole, la phonétique, le traitement du signal, la linguistique, l'informatique, l'intelligence artificielle, les statistiques, etc. Dutoit et collègues (2002) ont publié une étude qui met l'accent sur la façon dont ces domaines ont évolué ces dernières années.

L'utilisation de la parole dans les interfaces utilisateur est une réalité. Des requêtes simples concernant un solde bancaire ou des transferts d'appel téléphonique, peuvent déjà être traitées par des systèmes de reconnaissance de la parole. Cependant, si l'interface parlée peut apparaître comme une amélioration certaine, il est important de réaliser 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 même augmenter les difficultés, puisque l'utilisateur doit employer une liste limitée de commandes arbitraires qui lui sont en général inconnues.

Aujourd'hui, plusieurs applications commerciales sont disponibles (voir quelques unes en (Tsai, 2005)), utilisant essentiellement le standard VoiceXML (*Voice Extensible Markup Language*). Le VoiceXML est un langage du style XML qui permet le développement d'applications basées sur le WEB accessibles à partir d'un téléphone ou d'autres formes de dispositifs (Scharma et Kunins, 2001). Néanmoins, le VoiceXML ne permet pas la mise en place d'une conversation ouverte. Les applications résultantes, limitent les échanges à des mots ou à des séquences de mots équivalentes à des choix spécifiques.

Pour la réalisation de notre interface, nous nous sommes intéressés plus particulièrement aux moteurs de reconnaissance de la parole et à leurs limitations. La reconnaissance de la parole est un processus complexe qui aboutit, très fréquemment, à des résultats erronés. Ainsi, deux contraintes majeures doivent être prises en compte :

- les résultats provenant du moteur de reconnaissance de la parole peuvent être différents de ce que l'utilisateur a effectivement dit (en termes lexicaux et/ou syntaxiques), et;
- selon (Jurafski et Martin, 2000), les utilisateurs s'adressent aux ordinateurs différemment de la façon dont ils s'adressent à d'autres humains, essayant de s'adapter à ce qu'ils perçoivent comme des limitations de la machine.

Concernant le dernier point, nous croyons qu'en rendant l'interaction plus réaliste, nous pourrons réduire la charge cognitive. Ce rôle est parfaitement joué par l'agent assistant qui assume une stratégie coopérative, en essayent de comprendre les volontés du maître, en déclanchant des actions à son égard et en réduisant au maximum les échanges d'énoncés inutiles.

En ce qui concerne les erreurs de reconnaissance, nous avons recensé quelques points « sensibles », sources de problèmes et d'erreurs de traduction, que nous listons ci-dessous :

- la dépendance au locuteur : les systèmes sont peu adaptables, nécessitant un processus d'apprentissage préalable. Il est demandé à tout nouvel utilisateur de prononcer un ensemble de phrases comportant l'ensemble des phonèmes d'une langue, afin d'adapter les références à sa voix ;
- la taille du vocabulaire reconnu : le moteurs essayent de traiter le plus grand nombre de mots mais, dans ce cas, il y a plus de risques d'erreurs et le temps de traitement est plus long. Un grand vocabulaire est constitué de quelques milliers de mots, un vocabulaire moyen de quelques centaines et un vocabulaire petit de quelques dizaines de mots;
- les ambiguïtés sur les homonymes : les mots de prononciations proches peuvent être mal reconnus, comme par exemple : *book* X *look* ou *list* X *least* ;
- le bruit environnant : son filtrage est essentiel mais difficile à faire. D'ailleurs, il est recommandé d'utiliser un micro de bonne qualité pour essayer de minimiser ce problème.

Le moteur de reconnaissance fait la conversion mot à mot, ce qui signifie que l'application doit enchaîner ces mots pour former un énoncé :

« Un énoncé peut être un mot simple ou peut contenir plusieurs mots (une expression ou une phrase). »

L'énoncé est la matière première de l'étape suivante : le traitement et la compréhension. Bien entendu, l'étape suivante, est plus intéressante pour nous, puisque c'est au cours du traitement syntaxique et sémantique que nous allons rencontrer quelques unes de nos principales contributions.

Le processus de synthèse de la parole ne sera pas développé dans ce mémoire, car nous n'avons pas travaillé sur ce point, qui reste d'ailleurs un travail à faire. Même si l'agent assistant est capable de synthétiser ses énoncés, ses réponses restent limitées à des phrases préalablement définies, comme des messages d'alerte, des résultats d'exécution d'une tâche et des messages provenant d'agents de service. La génération du langage naturel est l'objet de plusieurs travaux, dont quelques uns sont indiqués ici : (Callaway et Lester, 2001), (Baptist, 2000), (Reiter et Dale, 2000) ou (Panaget, 1998).

3. Traitement et compréhension des énoncés

Comme schématisé sur la Figure 22, l'étape suivante consiste à traiter chaque énoncé pour préparer sa compréhension. Le Tableau 4 contient quelques énoncés typiques des domaines d'applications dans lesquels nous avons travaillé.

Enoncé
I'd like to open my email account.
I need a list of all project participants.
Could you list all articles about Agent?
List all meetings at my office for the next three days.
Send an email to Mike.
Give me the address of Paul Taylor.
I need a list of all conditions to receive an RMI.
When is the meeting with Paul?

Tableau 4 : Quelques énoncés typiques.

Traiter l'énoncé veut dire faire une analyse lexico-syntaxique à l'aide d'une série d'algorithmes linguistiques. La Figure 23 illustre les phases impliquées.

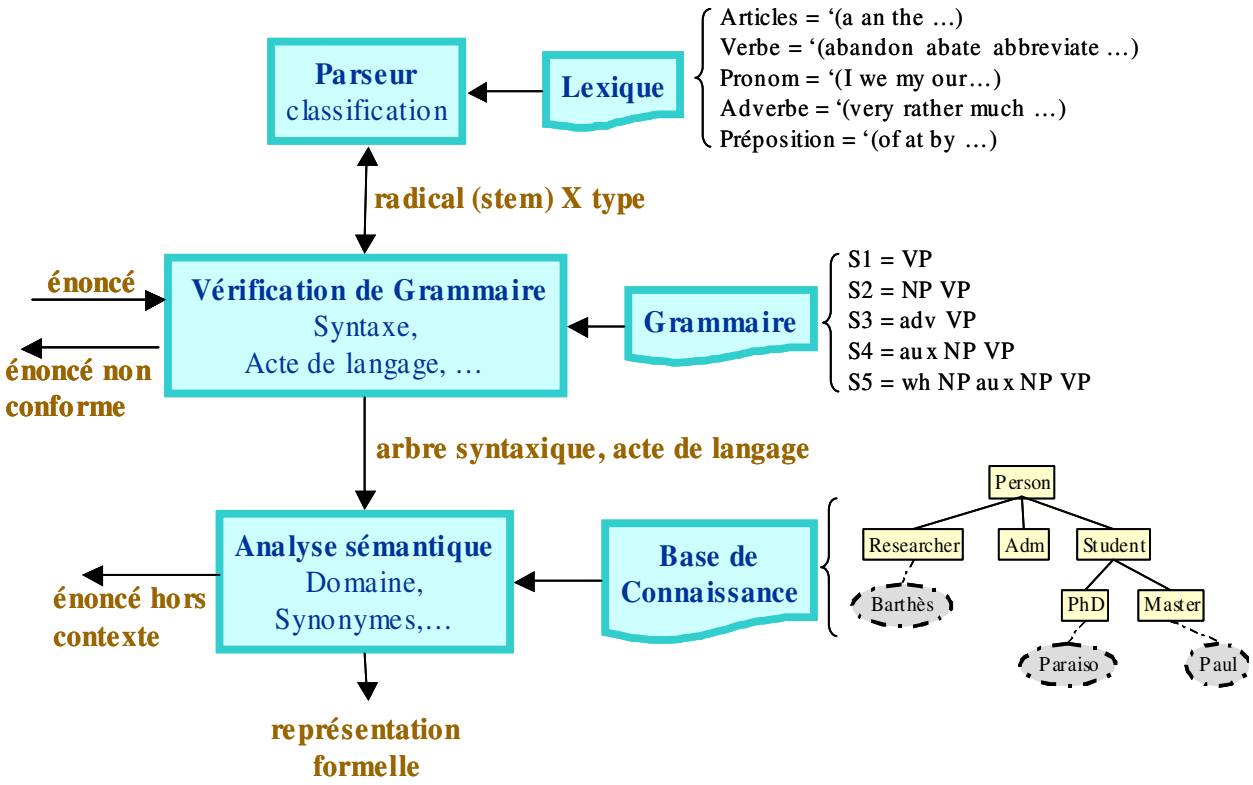


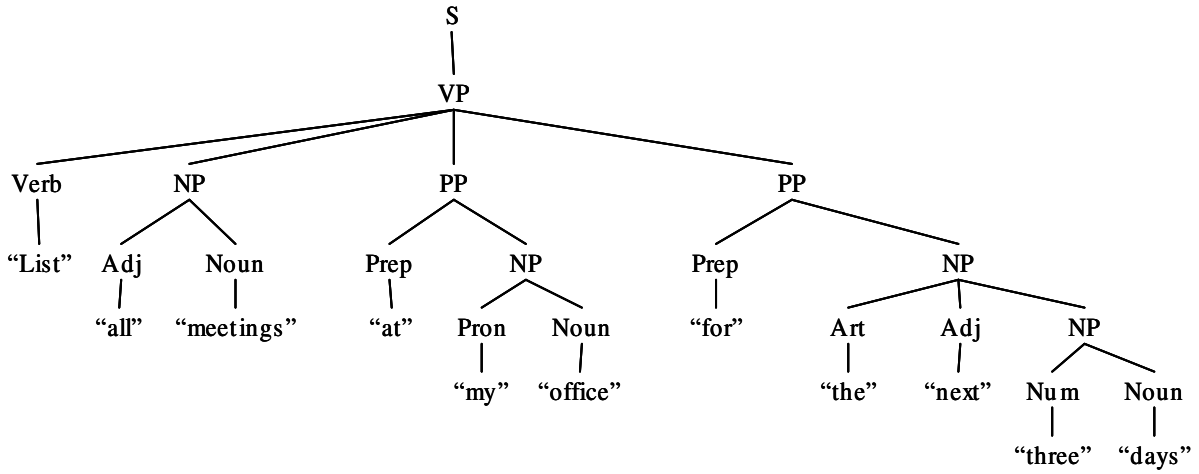
Figure 23 : Vision générale du traitement linguistique d'un énoncé.

D'abord, l'énoncé est soumis au analyseur syntaxique qui lui attribuera une description structurelle syntaxique, selon les règles grammaticales de la langue en question (l'anglais est la langue qui nous avons choisie), et avec l'aide d'un parseur capable d'identifier chaque mot (soit

un verbe, un nom, un article, etc.). Ensuite, l'analyse sémantique vérifie si l'énoncé est lié au domaine de l'application avec l'appui de la base de connaissance (formalisée sous forme d'une ontologie ou d'un graphe conceptuel par exemple). Finalement, après l'analyse sémantique, l'énoncé aura une représentation formelle compatible avec le moteur de raisonnement (peut être des expressions logiques par exemple). Au cours de ce processus l'énoncé peut être refusé, si par exemple il est mal formé (du point de vue du parseur ou de la grammaire de la langue) ou si l'énoncé est considéré hors du champ de l'application.

3.1. L'analyse syntaxique

La syntaxe est la partie du traitement qui s'intéresse à l'application des règles grammaticales qui servent d'abord, à gérer l'ordre des mots dans la phrase et, d'autre part, les relations qui existent entre les éléments qui la composent. Nous avons besoin essentiellement de deux composants : une grammaire et un parseur. Le résultat, comme illustré sur la Figure 24, est en général, un arbre syntaxique, classant les mots selon leur type et les groupant selon leur position et interrelation (correspondant à une organisation hiérarchique des constituants). Cette structure nous intéresse car elle contient des informations importantes pour l'étape suivante : si nous savons que l'énoncé est un ordre, que le mot *List* est un verbe et qu'à ce verbe correspondent un certain nombre d'éléments, nous pouvons plus facilement démarrer la compréhension de l'énoncé.




```
(S (VP List
     (NP all meetings)
     (PP at
        (NP my office))
     (PP for
        (NP the
           (ADJP next)
           three days))))
```

Figure 24 : Arbre syntaxique (en haut) et la même représentation (en bas) sous forme de liste (selon Link Parser (Grinberg et al., 1995)).

3.1.1 Les grammaires

Une des toutes premières approches connue pour exprimer une grammaire a été proposée par Noam Chomsky (1980). Sa stratégie est connue comme « orienté phrase » où la grammaire est un ensemble de « règles de production » que peuvent générer tous les énoncés bien formés d'une langue. Il s'agit des grammaires du type CFG (*Context-Free Grammar*). À cet ensemble de règles, on rajoute un lexique qui contient l'ensemble de mots et symboles de la langue. Le résultat d'une telle stratégie est la construction d'un arbre syntaxique généré par le parseur (Figure 24).

La Figure 25 donne quelques règles simples et un lexique pour la langue anglaise. Le concepteur des règles doit écrire toutes les règles nécessaires à l'analyse de tous les énoncés attendus (de questions, de réponses, de propositions, etc.). Chaque règle, ici représentée par S1, S2, etc., peut contenir deux classes différentes de symboles : les symboles *Terminaux*, qui correspondent à des mots et qui sont les feuilles de l'arbre syntaxique, et les symboles *Non-Terminaux*, qui contiennent des sous-groupes de symboles *Terminaux* ou *Non-Terminaux*, et qui sont les nœuds de l'arbre résultant. Les symboles *Non-Terminaux*, qui sont ici représentés

comme VP (*verb phrase*) ou NP (*noun phrase*), sont récursivement remplacés soit par d'autres symboles *Non-Terminaux*, soit par de symboles *Terminaux*. Ainsi, nous pouvons « lire » la règle S1 comme étant le fait qu'une phrase soit une phrase *Nominale* (NP) et une phrase *Verbale* (VP).

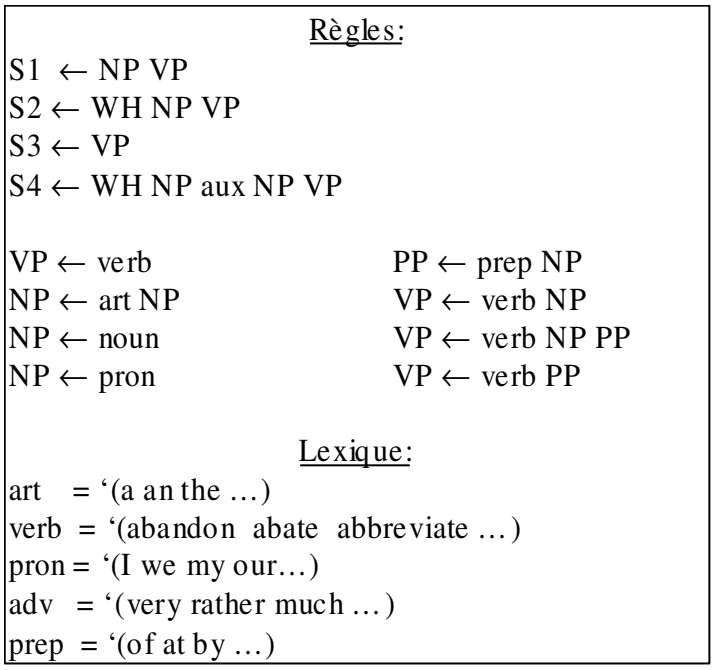


Figure 25 : Exemple de règles de production et d'un lexique d'une grammaire.

C'est au mécanisme d'analyse de construire cette représentation (comme indiqué sur la Figure 24), en respectant les règles grammaticales.

Il est clair que l'interprétation des énoncés reconnus doit être adaptée aux spécificités de l'oral et aux éventuelles erreurs de reconnaissance. Nous avons déjà remarqué que les énoncés « parlés » sont différents de ceux « tapés ». D'ailleurs, Jurafski propose une distinction terminologique entre un énoncé (*utterance* en anglais) pour les entrées parlées et une phrase (*sentence* en anglais) pour les entrées tapées. Nous avons récupéré cette définition et utilisons au cours de ce texte la définition d'énoncé comme étant une entrée *parlée*.

Un exemple de distinction entre les entrées tapées et parlées est la présence dans cette dernière d'une classe de mots connue comme disfluences. Les disfluences sont des mots dits « vides » (*uh*, *um*, entre autres) et qui peuvent être supprimés par l'analyseur (pour la plupart des applications ces mots sont inutiles).

En conséquence, il existe un certain nombre de grammaires spécialement conçues pour le traitement des entrées parlées, comme les grammaires lexicalisées d'arbres adjoints ou encore

des systèmes hybrides comme celui proposé par (Langley et al., 2002), qui mélange les grammaires textuelles et les techniques d'apprentissage automatique.

Gaiffe et collègues (2004), dans le cadre du projet européen OZONE (*Offering an Open and Optimal Roadmap Towards Consumer Oriented Ambient Intelligence*) (Ozone, 2004) ont utilisé les grammaires lexicalisées d'arbres adjoints (LTAG - *Lexicalized Tree Adjoining Grammar*). Les grammaires d'arbres adjoints se situent dans le paradigme des grammaires d'unification (ou grammaires basées sur les contraintes), avec comme particularité d'être intégralement lexicalisées (chaque item lexical correspond à une structure syntaxique). Dans ce cadre, la totalité de l'information grammaticale est exprimée dans le lexique.

Dzikovska (2003) a préféré intégrer des informations sémantiques spécifiques sur le domaine dans le lexique et dans la grammaire. Son but était d'accélérer le processus d'analyse et d'éviter les ambiguïtés au cours de l'analyse sémantique.

(LF LF_Filling*load)
(SUBJ (NP (SEM (phys-obj (intentional +) **(kr-type Person) (origin human) (form solid-object)**))))
    (SUBJ-MAP AGENT)
(DOBJ (NP (SEM (phys-obj (mobility movable) (kr-type commodity)))))
    (DOBJ-MAP THEME)
(COMP3 (PP (ptype into) (SEM (phys-obj (container +) **(kr-type Vehicle) (origin artifact) (form object)**))))
    (COMP3-MAP GOAL)

Figure 26 : L'entrée du verbe *Load* dans le domaine de l'application *Pacifica* (extrait de (Dzikovska et al, 2003)).

La Figure 26 montre une entrée construite à partir de l'énoncé « *load the oranges into the truck* » avec l'aide d'une ontologie de domaine. En principe cette solution ne nous intéresse pas, car nous nous engageons à construire un traitement syntaxique indépendant du domaine. Dans notre approche, la grammaire reste indépendante du domaine, même si le lexique peut être enrichi avec des entrées spécifiques, comme les jargons du domaine de l'application.

En ce qui concerne les grammaires, notre choix s'est porté sur les grammaires du style *Link Grammar* (Grinberg et al., 1995), capables de fournir des informations précieuses comme des relations de quantité, qualité, genre, etc., entre les mots. La grammaire ne contient pas de règles dépendant du domaine. Pour cela, il a fallu développer un mécanisme propre d'analyse. Dans le chapitre suivant nous allons décrire en détail cette approche.

3.1.2 Les analyseurs

Le but est de construire une représentation de l'énoncé, basée sur la grammaire de la langue. Au début, de l'arbre résultant, nous savons que la racine (représentée comme *S* sur la

Figure 24) et les feuilles (les mots de l'énoncé). La recherche de la structure de l'arbre obéit généralement à l'une parmi deux stratégies (Hellwig, 1999) : descendante ou ascendante.

L'analyse descendante

L'analyse descendant démarre à partir de la racine, essayant de trouver les règles grammaticales qui combinées, permettent d'arriver aux feuilles.

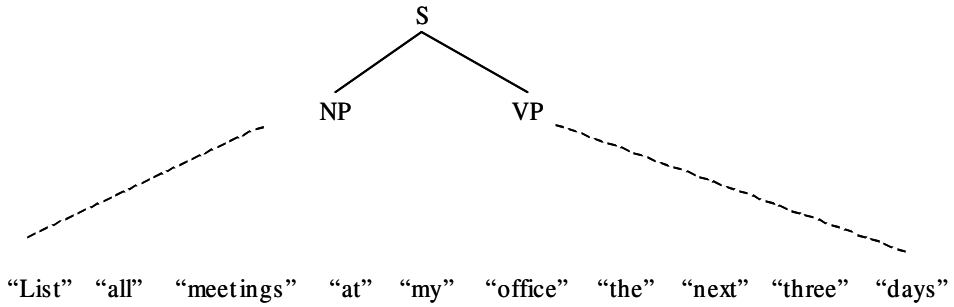


Figure 27 : l'arbre après la première itération descendante.

Par exemple, après l'application de la règle S1 ← NP VP, nous avons la situation montrée sur la Figure 27.

L'analyse ascendante

Dans ce cas, l'analyse progresse en direction de la racine, à partir des nœuds *Terminaux*, comme l'illustre la Figure 28.

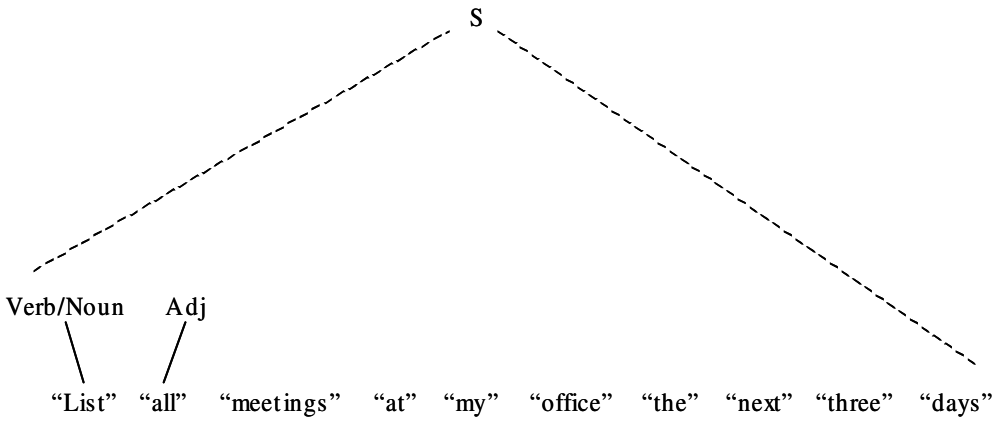


Figure 28 : l'arbre au cours de montage ascendante.

Lors du processus d'analyse, qui est récursif, un mot peut être classé plusieurs fois de façons différentes. Ainsi, un mot comme *book*, peut être classé comme verbe par une règle et comme nom par une autre. Bien évidemment, à la fin du processus un seul arbre syntaxique est fourni comme résultat.

Comme nous l'avons indiqué dans la section précédente, nous avons développé un processus d'analyse, du type descendant basé sur le système *Link Parser* (Grinberg et al., 1995), décrit en détail dans le chapitre suivant.

Avant d'engager le processus de compréhension, arrêtons nous sur une étape intermédiaire : l'étude de la signification des mots. Cette étape, connue comme la sémantique lexicale, nous permettra de comprendre que le lexique est plus qu'une liste de symboles de la langue.

3.2. La sémantique lexicale

La sémantique lexicale concerne les relations entre les mots (ou lexèmes). Ces relations permettront de combiner les mots entre eux formant un réseau sémantique très utile au cours de l'analyse sémantique. Prenons, par exemple, le mot *paper* (papier en anglais). Selon la version 1.7.1 de WordNet (décrit dans la section suivante), le mot *paper* peut être classé soit comme un verbe (où il peut avoir deux sens distincts) soit comme un nom (où il peut avoir sept sens distincts). Toujours grâce a WordNet, nous pouvons lister quelques relations sémantiques existantes entre le mot *paper* et d'autres mots. *Paper* et *newspaper* peuvent être acceptés comme synonymes, quand le sens décrit une publication quotidienne (un journal). Les mots *card*, *confetti* et *sheet* ont une relation d'hyponymie avec *paper*. La Figure 29 illustre graphiquement d'autres relations entre plusieurs mots.

Si nous savons exploiter ces relations nous pouvons augmenter le lexique de notre système. Ainsi, le lexique deviendra la conjonction de plusieurs composants : les lexèmes (racine de mots), leurs sens et une liste de relations sémantiques entre eux. Dans ce contexte nous définissons le lexème et le sens comme étant :

« Le lexème est une entrée individuelle du lexique ; il est l'élément de base d'un mot. Chaque lexème a son propre sens, c'est-à-dire, sa signification. »

Certaines catégories sémantiques nous intéressent plus particulièrement. Nous les décrivons brièvement maintenant.

Synonymie

La synonymie indique que différents lexèmes ont la même signification. En pratique, si nous trouvons dans un énoncé un mot et, si nous le remplaçons par un de ses mots synonymes,

nous sommes certains que le sens de l'énoncé reste le même. Un détail important : les mots de catégories syntaxiques différentes ne peuvent pas être synonymes (ni faire partie du même ensemble de synonymes).

Hyponymie/Hyperonymie

L'hyponymie et l'hyperonymie sont des relations sémantiques liant le sens des mots. Elles sont aussi connues sur le nom de relation *IS-A* ou *kind of*. Nous pouvons définir que *Maple* est un hyponyme d'arbre et que *Oregon Maple* (une variété de *Maple* de l'ouest des Etats Unis) est un hyperonyme de *Maple*.

Méronymie/Holonymie

C'est la relation sémantique qui exprime la relation partitive hiérarchisée : un méronyme *A* d'un mot *B* est un mot dont le signifiant désigne une sous partie du signifiant de *B*. La relation inverse est l'holonymie. Cette relation est aussi connue comme *HAS-A* ou *part-whole*. Par exemple, pédale est un méronyme de bicyclette.

Obtention de la forme infinitive d'un verbe (***Entailment*** en anglais)

Entailment est la relation qui existe entre deux mots provenant du même verbe et consiste à fournir la version infinitive d'un verbe. Ainsi, l'application de cette opération au mot *written* nous donne *write*, ou encore au mot *gone* nous donne *go*.

Étant donné ces définitions, un groupe de chercheurs les ont réunies en un seul système : WordNet. Voyons comment elles sont organisées dans WordNet et principalement à quoi cela peut bien servir.

3.3. L'ontologie Lexicale WordNet

WordNet est un lexique de référence (exclusivement pour la langue anglaise) dont la conception est inspirée de théories psycholinguistiques (Felbaum, 1998). WordNet peut être considéré comme un réseau sémantique où chaque nœud représente un concept (qui peut être une entité, un artéfact, un objet, etc.). Chaque nœud est composé d'un ensemble de synonymes (le *synset*) qui représentent le même concept. Les *synsets* sont reliés par des arcs qui décrivent les relations sémantiques entre les différents concepts. Ils sont divisés en 4 catégories (noms, verbes, adjectifs et adverbes). La Figure 29 illustre une petite partie de l'ontologie, où le concept *Car* est

composé de plusieurs éléments (*accelerator, air bag*) et une spécialisation des concepts plus « généraux » comme un *Motor Vehicle* or un *Artifact*.

WordNet peut être utile dans plusieurs situations : au cours de l'analyse linguistique et/ou pendant le traitement sémantique. Par exemple, dans le projet QALC (Ferret et al., 2001) WordNet est utilisé pour trouver les variantes sémantiques d'un même mot. Dans le système PRECISE (Popescu, 2003) qui est capable de convertir des phrases en anglais en requêtes SQL, WordNet est utilisé à chaque fois qu'une consultation du lexique est nécessaire, surtout pour construire des groupes de synonymes des mots clés sélectionnés à partir de l'entrée. (Aussenac-Giles et Mothe, 2004) se sont appuyés sur WordNet et sur une ontologie de domaine pour développer un système d'exploration d'une base de documents. Le système est capable de reformuler une requête d'utilisateur en y ajoutant des informations liées à la requête en question, trouvés dans le thésaurus de WordNet.

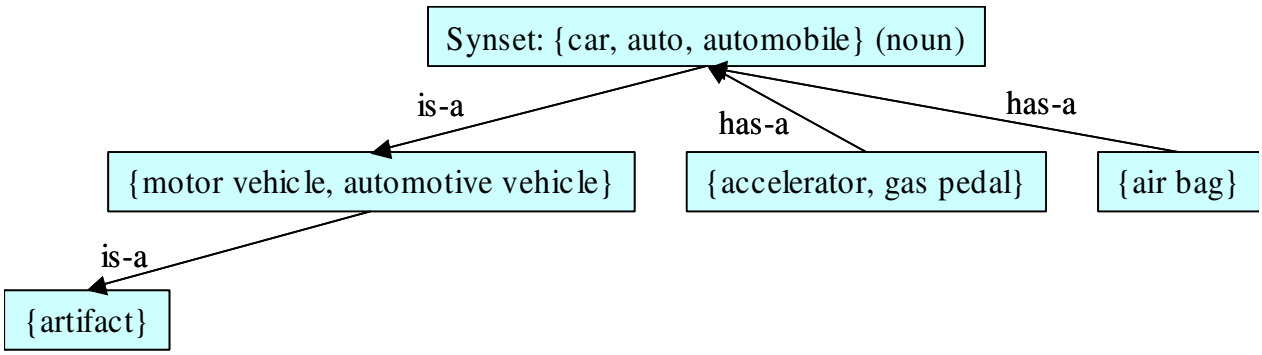


Figure 29 : Quelques concepts WordNet.

WordNet est aussi critiqué par plusieurs chercheurs. Les critiques formulées par (Harabagiu et al., 1999) ou par (Ferret, 2004) portent à la fois sur la nature des sens de chaque mot et leur caractérisation. Ces sens sont jugés à la fois trop fins et incomplets. Par ailleurs, leur caractérisation, réalisée pour l'essentiel à travers des relations de synonymie, d'hyperonymie et d'hyponymie, manque d'éléments définissant leur contexte d'usage (Ferret, 2004). Ferret propose que la définition du sens des mots soit obtenue à partir de leur usage. Plus précisément, il propose de différencier les sens d'un mot à partir d'un réseau de cooccurrences lexicales construit sur la base d'un large corpus.

Conscients de ces limitations, nous avons utilisé WordNet comme source pour notre lexique (environ 126.000 mots), à partir du code source et des fichiers du thésaurus de WordNet qui sont librement disponibles. Dans le chapitre des expérimentations, nous donnons plus de détails sur le rôle de WordNet dans notre approche (section 2.3).

Enfin, il faut signaler qu'une version européenne de Wordnet, l'EuroWordNet (Vossen, 1998) a été produite par un groupe de chercheurs européens à la fin des années 90. Des versions existent pour plusieurs langues, dont le français, l'espagnol ou l'allemand. Nous ne l'avons toutefois pas utilisée au cours de ce travail.

3.4. La compréhension des énoncés

Bien que ce soit un processus complexe, l'interprétation de l'énoncé ne suffit pas. La compréhension passe par la collecte des éléments qui permettront, en fin de compte, de produire une action cohérente, adéquate et pertinente. Prenons l'exemple suivant :

USER : List all meetings at my office for the next month with my advisor.

Il s'agit d'un acte directif (un ordre) au contenu explicite, mais qui exige des connaissances supplémentaires (qui est le « *advisor* » ?). Cet exemple démontre que les énoncés, ne fournissant pas en eux-mêmes, toutes les informations nécessaires pour les comprendre, il faut y adjoindre le contexte (énoncés précédents et les connaissances du domaine). Cela veut dire qu'il faut à l'interface, des connaissances supplémentaires, ou encore qu'il faut, posséder des représentations explicites de la tâche pour appréhender totalement la situation correspondant à l'énonciation (Caelen, 2004). Comprendre c'est donc plus que reconnaître, c'est être potentiellement capable de fournir une réponse à un énoncé.

Notre but est d'arriver à une représentation formelle de ce que le système « a compris » à partir de l'analyse de l'énoncé. La Figure 30 schématise le chemin à parcourir. Elle contient un exemple classique lié à la recherche de vols entre deux villes (*Air Travel Information Service - ATIS*[6]).

[6] La DARPA, dans les années 1990, a promu une série d'expérimentations et de recherches liées à la tâche de trouver des informations dans le domaine des vols (le système connu comme ATIS). Le but était de développer des systèmes de requête dans une base de données relationnelle qui contenaient les informations sur les vols.

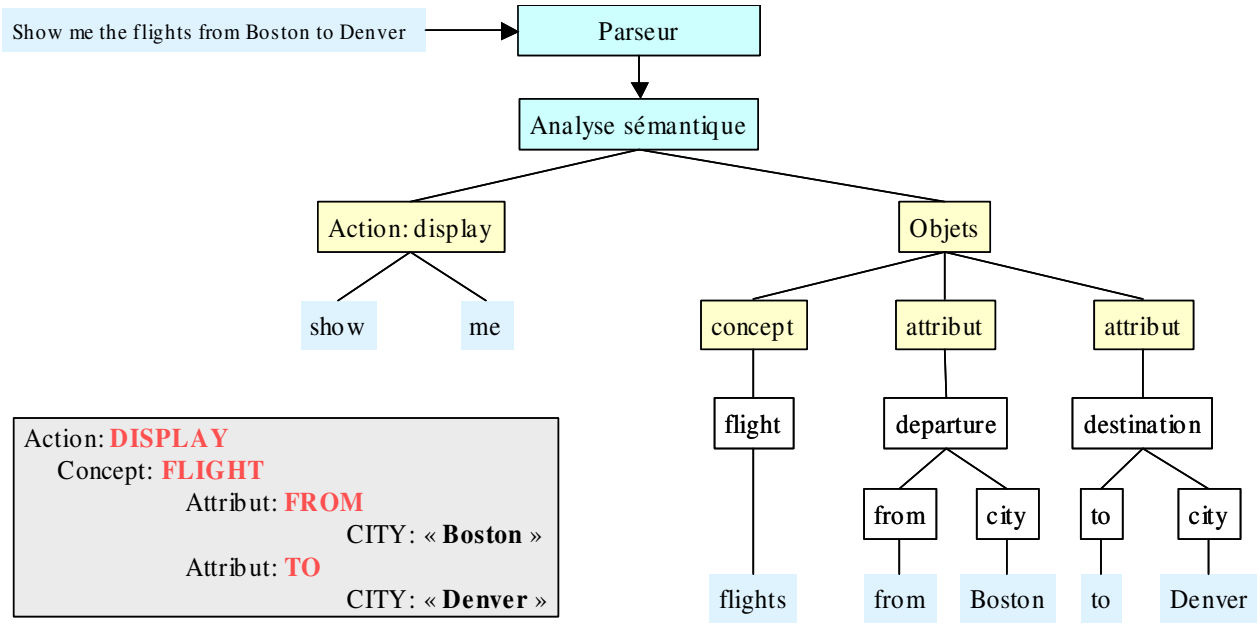


Figure 30 : De la requête à l'action.

Comme entrée du schéma, nous avons l'énoncé « *Show me the flights from Boston to Denver* » qui passe d'abord par l'étape de *parsing*. Ensuite, à partir de l'arbre obtenu, une structure plus formelle est construite (un *frame*) liant les informations utiles récoltées à partir de l'énoncé. La façon d'obtenir cette forme plus formelle varie : il y a ceux qui utilisent les relations entre les objets du domaine (base de connaissances) et les tâches préalablement décrites (Enembreck, 2003), et il y a ceux qui construisent une requête sans avoir besoin d'un modèle des tâches (Barbuceanu, 2003). Dans notre exemple, une hiérarchie d'objets et d'attributs couvrent le domaine : le concept *Fligth* et les attributs *Departure* et *Destination* y sont présents. Ainsi, l'analyseur va chercher les informations manquantes : le nom de la ville de départ et d'arrivée.

Dans le système proposé par Enembreck pour l'agent assistant, cette construction est liée à la structure de la phrase et dépend d'un patron de tâches (Figure 31). La structure de la requête formelle sera déterminée par la tâche sélectionnée comme la plus appropriée à l'énoncé. Nous allons montrer dans le chapitre d'expérimentations que cela est incompatible avec notre approche, vu que les tâches sont distribuées entre les agents de services. Par conséquent, l'agent assistant ne les connaît même pas.

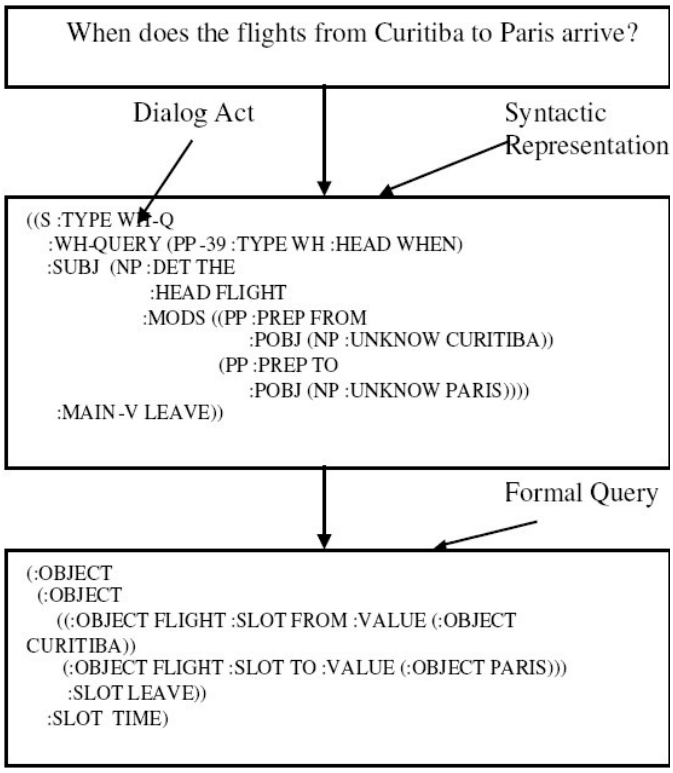


Figure 31 : Les structures internes proposées par Enembreck (2003).

En effet, une forme indépendante des tâches nous sera plus utile. Ceci est possible si nous avons une représentation explicite du domaine, par exemple sous forme d'une ontologie.

« Il faut aussi souligner que la réussite de l'analyse sémantique est fortement liée au bon déroulement de l'analyse syntaxique. »

Nous ne croyons pas aux systèmes qui essayent de répondre aux requêtes de l'utilisateur seulement en cherchant des mots clés dans l'énoncé. Ramos a réalisé en tel système qui a quelques difficultés quand on pose des requêtes un peu plus complexes, comme par exemple, celles que nous avons indiquées dans le chapitre précédent : comment traiter de multiples demandes dans une seule requête (« Give the address of Kassel and open my email account ») ou comment traiter les modificateurs applicables aux objets trouvés dans l'expression (« List the *latest* report on agents »).

3.5. L'interprétation sémantique des énoncés à l'aide des ontologies

Nous observons un intérêt croissant pour les ontologies dans les systèmes de traitement de la langue (Bateman, 1997) et les systèmes de dialogue. Leur rôle varie allant de l'analyse lexicale jusqu'à la description des connaissances et des tâches dans les systèmes de dialogue.

Dzikovska et collègues (2003) ont présenté une méthode d'adaptation d'un mécanisme d'analyse lexicale indépendant du domaine, par l'emploi de deux ontologies distinctes : une ontologie générale pour la représentation de la langue et une autre dédiée au domaine.

Dans un récent papier dans le domaine des systèmes de dialogue et d'extraction des connaissances, Eriksson (2003) explore les exigences pour les ontologies dans les systèmes de dialogue, mettant en avant la place qu'elles peuvent jouer dans la définition d'un vocabulaire commun.

Pour Milward et Beveride (2003), les systèmes de dialogue sont en train d'évoluer vers une nouvelle génération de systèmes pratiques, basés sur une forte description de connaissances et de tâches du domaine sous forme d'ontologies, ce qui leur permet de garder une plus grande indépendance vis-à-vis du domaine. Ils ont appliqué ces principes à des applications dans des domaines assez différents comme un système de diagnostic médical ou au contrôle d'appareils ménagers interconnectés. D'ailleurs, Dahlbäck et Jönsson (1997) estiment que les systèmes de dialogue basés sur la théorie des actes de langage pourraient être utilisés plus extensivement s'ils étaient combinés à de bases des connaissances du domaine.

Un autre exemple d'application des ontologies dans le domaine du dialogue est présenté par Purver et collègues (2005), qui proposent un mécanisme pour traiter des enregistrements de conversations au cours de réunions de projets.

Dans le projet TRIPS (Allen et al., 2001a), les ontologies sont utilisées pour représenter les connaissances et les tâches, ce qui selon les auteurs, réduit la complexité des modules de traitement linguistiques.

D'autres modèles ont été présentés, notamment des modèles statistiques. Les modèles statistiques, comme celui proposé par He et Young (2005), conduisent l'analyse au niveau de l'analyseur lexical, laissant des informations utiles sur le domaine hors du processus d'interprétation.

Concernant le processus de conception d'une ontologie, sa réalisation est très complexe comme l'indiquent quelques travaux dans ce domaine, comme ceux publiés par Noy et Hafner (1997). Les auteurs ont développé une méthode de comparaison entre les ontologies dans plusieurs projets différents dans le domaine de traitement de la langue.

4. Les actes de langage

Une fois l'énoncé syntaxiquement traité, la suite est la recherche du « quoi faire » avec les informations interprétées. Comme nous avons vu dans la section précédente, la transformation de l'arbre syntaxique en une représentation formelle permettra au système de raisonner et donner suite à l'entrée d'utilisateur. Nous allons voir dans cette section que les croyances et les intentions de l'individu qui a fourni l'énoncé peuvent aussi affecter la compréhension de celui-ci. Austin (1962) a proposé dans son travail intitulé *How to do things with words*, une des théories la plus utilisé dans ce domaine : les Actes de langage.

« L'idée centrale tourne autour du fait que chaque énoncé devrait être compris comme une action voulue par son émetteur. »

La théorie des actes de langage est plutôt orientée vers la pragmatique du langage. Austin soutient que les énoncés dans une conversation peuvent être classés selon trois types d'actes:

- les *actes locutoires*: on énonce quelque chose ;
- les *actes illocutoires*: on demande une action en disant quelque chose, avoir l'intention d'obtenir l'action en le disant ;
- les *actes perlocutoires*: énoncés émotionnels où on essaie de convaincre son interlocuteur.

Searle (1975) a approfondi l'étude d'Austin, en identifiant cinq actes illocutoires fondamentaux. Ces cinq actes peuvent être qualifiés de la façon suivante :

- Assertif: qui engage l'émetteur sur la vérité de quelque chose (affirmation, conclusion, déduction, insistance...) ;
- Directif: pour que le sujet récepteur fasse quelque chose (requête, suggestion, invitation, avertissement) ;
- Promissif: qui engage l'émetteur pour une action future (promesse, offre, refus, permission...) ;
- Déclaratif: qui établit une correspondance entre le contenu de l'acte illocutoire et la réalité (constat, déclaration de guerre, nomination, abandon...) ;
- Expressif: qui exprime un état psychologique à propos d'une réalité donnée (remerciement, excuses...).

Dans notre travail nous nous intéressons plus particulièrement aux actes illocutoires, car sont eux qui plus explicitent facilement les intentions d'utilisateur. Nous allons restreindre l'analyse des énoncés à trois catégories d'actes de langages : question, réponse et ordre (Paraiso et Barthès, 2004). D'autres auteurs ont aussi suivi cette stratégie comme (Kumar et Romary, 2002), qui ont spécifié un langage de représentation (MML – *Multimodal Interface Language*) utilisé dans le projet OZONE (Gaiffe et al., 2004).

5. La gestion des énoncés : la référence

Cette section présente la problématique liée à la gestion des énoncés, concernant le traitement des phénomènes de discours et de la référence. Le traitement de la référence est complexe restant encore très actif en linguistique (Beust, 2000) et (Tutin, 2000).

La référence dans les systèmes de dialogue est l'étude de comment les objets du monde réel sont présents dans les énoncés, comment ils doivent être stockés et comment ils sont utiles pour maintenir le contexte d'une conversation. La forme la plus simple de la référence apparaît quand nous avons dans une phrase un nouvel objet dans le contexte ou une référence à un objet déjà connu dans ce contexte. Pour donner un exemple, considérons les énoncés suivants, produits à partir d'une conversation entre l'utilisateur et son agent assistant :

USR (1): Give me the list of articles written by Mike Palmer.
AP (2): *« imprime la liste d'articles »*
USR (3): Is he a participant of the project?
AP (4): Yes.
USR (5): Do you know his telephone?
AP (6): Yes. The telephone of Mike Palmer is: 03 44 23 44 55.
USR (7): and his address?

Dans le premier énoncé, l'utilisateur cite le nom de *Mike Palmer* pour la première fois. Si le système veut garder le contexte de la conversation, cette information doit être retenue. Ensuite, dans l'intervention suivante (l'énoncé 3), il le cite indirectement en utilisant le pronom *he*. Ce phénomène est connu comme une anaphore, c'est-à-dire, le fait de référencer une entité déjà introduite dans le discours.

Un autre phénomène commun dans les dialogues est l'occurrence d'ellipses. La détection d'une ellipse passe par l'analyse d'une phrase qui, dans un premier temps, paraît « incomplète », mais en réalité, les « parties manquantes » ont été présentées dans les énoncés précédents. Dans

l'énoncé (5) l'utilisateur demande le numéro de téléphone de Mike. En (7), il demande son adresse. L'ellipse est évidente dans ce cas, puisque l'énoncé (7) peut être traité comme un complément de l'énoncé (5).

Anaphore et ellipses sont relativement difficiles à détecter, car un énoncé peut contenir des références à des objets non encore connus ou présentés dans un énoncé posté plusieurs tours auparavant. Dans le chapitre d'expérimentation nous allons décrire un algorithme de résolution de la référence.

Ces phénomènes nous serviront à mieux comprendre les systèmes de dialogue que nous présentons par la suite.

6. Les systèmes de dialogue

Le terme « dialogue » est employé dans différentes communautés de différentes manières. Les systèmes issus de ces conceptions de dialogue différents aussi. Il y a ceux qui se contentent de développer des systèmes où l'interaction est guidée par l'application. Nous pensons notamment aux applications du type commerce électronique (voir quelques exemples en (Tsai, 2005)) qui proposent un menu d'options à l'utilisateur. Dans ce cas, la définition de dialogue est figée, car l'initiative du dialogue part toujours de l'application. Comme argumentent (Allen et al., 2001) : d'une part, en commandant l'interaction, les énoncés d'utilisateur sont beaucoup plus prévisibles, conduisant à améliorer l'identification et le traitement linguistique ; d'autre part, les systèmes limitent l'interaction, forçant l'utilisateur à fournir des informations inutiles pour le problème en question.

« Nous nous sommes intéressés à une catégorie de systèmes de dialogue, où l'interaction est plus qu'une série de commandes : l'interaction est collaborative et vise à atteindre l'exécution des tâches et la satisfaction des besoins de l'utilisateur du système. »

Les systèmes de dialogue deviennent un domaine d'intérêt croissant, dans la recherche et même dans des applications pratiques. Ils diffèrent dans leur complexité, due au domaine et à l'approche adoptée dans leur conception. Il y a des systèmes qui dépendent fortement des connaissances du domaine (contenant par fois plusieurs sources d'information agissantes les une sur les autres, comme dans le projet TerreGov (Moulin et al., 2005)). D'autre part, il y a des systèmes fondés sur des modèles et des procédures beaucoup plus simples.

Allen et collègues (2001b) proposent une classification des systèmes de dialogue en fonction de la complexité des tâches à accomplir (voir Tableau 5). La complexité du système de dialogue croît, logiquement, en fonction de la nature de l'interaction avec l'utilisateur. Plus l'utilisateur va être prié de fournir des détails et des informations, plus la complexité sera importante. Dans un service d'information météorologique, exemple de système moins complexe, les informations provenant de l'utilisateur sont assez simples et limitées : grosso modo nous avons besoin de l'endroit géographiquement situé et d'une date pour fournir un bulletin météo.

Complexité de la tâche	**Technique utilisée**	**Type de dialogue manipulé**	**Exemple de tâche**
Moins complexe	Script à l'état fini	L'utilisateur répond à des questions	Service météo
	Basé sur des *frames*	L'utilisateur peut poser des questions. Des clarifications simples par le système	Départ et arrivée des trains
	Ensemble de contextes	Changement entre sujets prédéfini	Agent de réservation de voyage
	Modèle basé sur des plans	Construction interactive d'un plan avec l'utilisateur (sous-dialogue)	Assistant au design d'une cuisine
Plus complexe	Modèle basé sur des agents	Différentes modalités	Gestion d'ambulances

Tableau 5 : Dialogue et la complexité de la tâche.

En fonction de nos domaines d'applications cibles (la gestion et capitalisation de connaissances ou l'aide à la gestion de projets, par exemple), les modèles les plus complexes nous intéressent plus particulièrement. Dans ce contexte, nous avons besoin de définir quelques éléments supplémentaires : la notion de *But*, la notion d'*Incidence* et la notion de *Stratégie de dialogue*.

Le *But* symbolise un certain état final visé du monde. Il sera atteint après quelques échanges (suite de tours de parole) entre les participants du dialogue (dans notre cas, l'utilisateur et l'agent assistant). Même si le but final n'est pas toujours prévisible au départ (Caelen, 2004), au cours de nos exemples, nous allons accepter comme but, le fait que l'utilisateur a montré un intérêt pour une tâche particulière (par exemple la recherche d'un report sur un projet spécifique).

Une *Incidence* est le fait qui provoque le changement de l'état d'un but : le questionner ou le mettre en attente. Elle peut être produite par tous les participants du dialogue.

La *Stratégie de dialogue*, selon Caelen (2004), est la manière de gérer les tours de parole entre les interlocuteurs pour conduire à un échange ou à une incidence. La stratégie vise à choisir la meilleure manière de satisfaire les buts à un moment donné.

6.1. Les stratégies de dialogue

Nous pouvons définir cinq types de stratégies de dialogue.

Stratégie réactive

Elle consiste à déléguer l'initiative à l'autre, soit en lui faisant endosser son but (cas de demande d'aide ou d'assistance), soit en adoptant son but (cas du serviteur). Un participant est passif et l'autre est actif. Cela a pour conséquence d'ouvrir tout type de stratégie au participant actif.

Stratégie directive

Elle consiste à garder l'initiative pour conduire le dialogue : en maintenant le but de l'échange, en imposant son but et en ignorant éventuellement celui de l'autre. Cela a pour conséquence d'imposer une réponse réactive ou négociée à un des participants, et de limiter ainsi la variété de ses stratégies. L'un est actif et l'autre devient passif.

Stratégie constructive

Elle consiste à déplacer le but courant momentanément afin de provoquer un détour (supposé constructif), par exemple pour faire remarquer un oubli, une erreur, faire une citation, rappeler un fait ancien, etc. : le but courant est mis en attente et un nouveau but est posé. Contrairement à une incidence, un détour ne ramène pas nécessairement à l'échange initial, il peut laisser la conversation en suspens ou conduire à un autre détour.

Stratégie de coopération

Elle consiste à tenir compte du but de l'autre en lui proposant une ou plusieurs solutions qui les amènent tous les deux à atteindre leur but, si ces derniers ne sont pas incompatibles. Cela conduit au déroulement d'un processus complexe — évaluer la situation, présenter une explication, éventuellement des exemples, des aides ou des arguments pertinents, en maximisant l'espace de concession.

Stratégie de négociation

La négociation peut se produire dans une situation où les buts sont incompatibles et où les interlocuteurs veulent minimiser les concessions. La négociation avance sur un schéma assez classique, par des séquences argumentatives (proposition, acceptation et réfutation) jusqu'à convergence ou constat d'échec.

Il faut préciser qu'au cours d'une conversation, des stratégies différentes peuvent être utilisées.

6.2. Les composants d'un mécanisme de gestion du dialogue

Comme nous le montre la Figure 21, la gestion d'une conversation est le résultat du travail de plusieurs modules. Il est presque impossible de définir une application pratique sans tous ces modules. La gestion du dialogue est donc, une partie de ce processus, et dépendante d'autres composants. En plus, c'est le gestionnaire du dialogue qui doit décider du « quoi dire » et du « quoi faire ».

Notre conception de gestionnaire de dialogue impose que celui-ci ait un certain nombre de capacités minimales :

1. la mise en place d'une stratégie de dialogue appropriée au but d'utilisateur ;
2. la maintenance d'une mémoire de référence et de contexte, et ;
3. la planification et contrôle d'exécution de tâches ;

Le gestionnaire du dialogue doit être capable de choisir, parmi les cinq stratégies décrites dans la section 6.1, lesquelles employer selon les besoins de la conversation engagée avec l'utilisateur.

Le contexte est primordial pour la cohérence du dialogue. Il nous permettra de fournir des réponses à des questions venues, soit d'utilisateurs, soit d'autres agents, mais aussi de résoudre les problèmes d'anaphore, d'ellipses, entre autres. La maintenance d'une mémoire d'objets mentionnés par un de participants du dialogue, permettra leur réutilisation future (exemple : si l'utilisateur a dit une fois qu'il habite à Paris, le gestionnaire doit savoir stocker cette information et l'utiliser quand il faut, sans la demander à nouveau).

Enfin, le gestionnaire est également responsable de planifier et de contrôler une pile de tâches du système.

Pour donner un exemple, nous avons reproduit sur la Figure 32, l'architecture du système TRIPS (*The Rochester Interactive Planner System*) (Allen et al., 2001a), très connu dans le domaine.

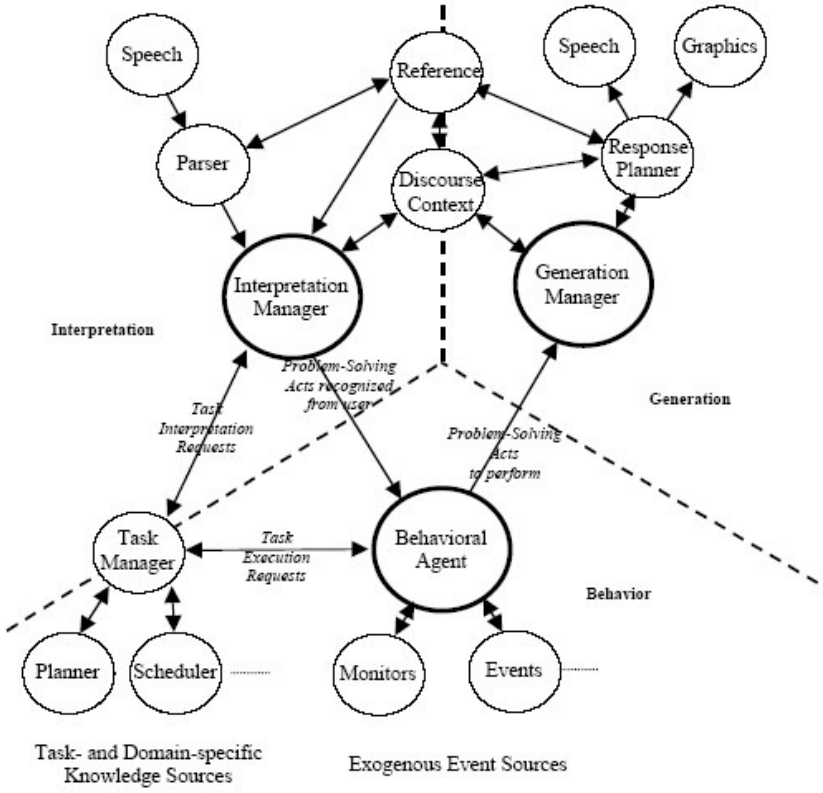


Figure 32 : Architecture du système TRIPS (extrait de (Allen et al., 2001a)).

La Figure 32 montre tous les modules impliqués dans la gestion du dialogue avec l'utilisateur. Elle est composée de trois modules de contrôle : le *Interpretation Manager*, le *Generation Manager* et le *Behavioral Agent*. Même si le système est plus approprié à des applications plutôt planification collaborative, son architecture nous montre clairement tous les composants fondamentaux d'un système de gestion du dialogue : l'interprétation des énoncés (*Interpretation Manager*), la gestion du dialogue (par le *Generation Manager* et les modules *Reference* et *Discourse Context*) et le pilotage de l'agent et ses actions (*Behavioral Agent*).

Le projet Galaxy Communicator (Ward et Pellom, 1999), développé par plusieurs universités américaines et une évolution du système MIT Galaxy (Seneff et al, 1998), propose une architecture (Figure 33) pour la construction et l'optimisation de systèmes de dialogue parlé. La structure résultante est un groupe de composants appelés *serveurs* et qui communiquent les uns avec les autres par l'envoi de messages. Comme nous pouvons le constater sur la Figure 33, définir un centralisateur (le *hub*) implique de figer les protocoles d'échange entre modules,

compliquant la mise en place d'une telle organisation dans une structure dynamique comme un SMA.

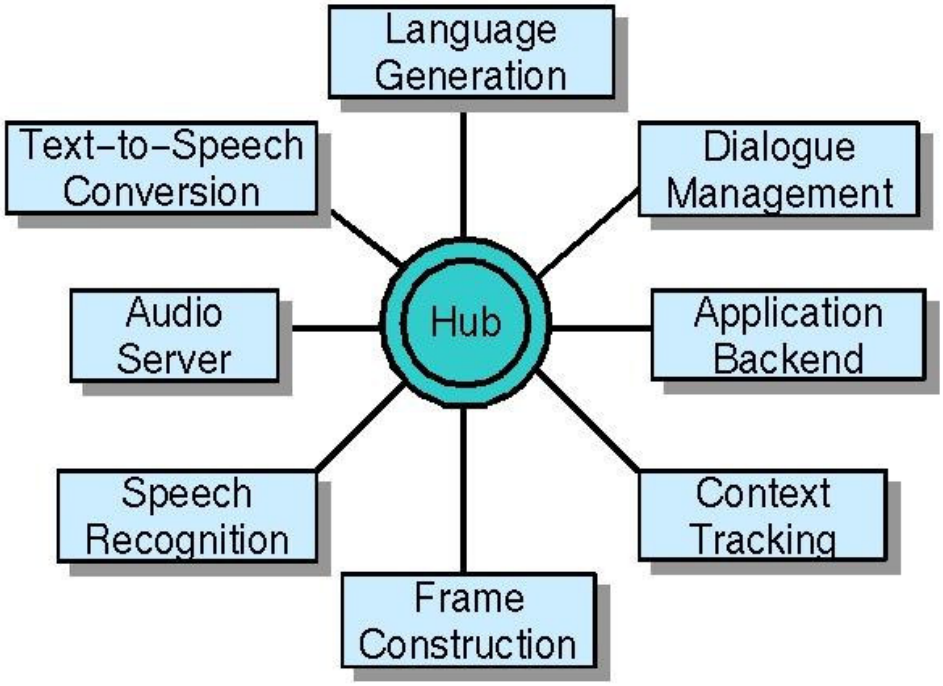


Figure 33 : L'architecture Galaxy Communicator (extrait du site : http://communicator.sourceforge.net).

Le système a été utilisé surtout pour des applications d'accès à l'information (réservations hôtels/avions, météo, etc.). Le projet s'est terminé en 2003, mais une version *code source libre* est disponible.

6.3. L'efficacité du système de dialogue parlé

Après la réalisation et la mise en route du système de dialogue parlé, nous pouvons évaluer sa performance par l'application d'un test de mesure d'efficacité. PARADISE (*PARAdigm for DIalogue System Evaluation*) (Walker et al, 1997) est le plus populaire. Il permet le découpage des tâches du comportement du système. Il permet aussi la comparaison entre plusieurs stratégies de dialogue différentes et la mesure de la contribution relative de plusieurs facteurs pour la performance globale.

PROMISE (*PROcedure for Multimodal Interactive System Evaluation*) (Beringer et al., 2002) est l'extension de PARADISE pour les systèmes multimodaux. Il résout le problème d'évaluation des diverses modalités présentes.

Dans la suite de ce chapitre, nous allons présenter quelques systèmes conversationnels représentatifs des discussions tenues jusqu'ici.

7. Exemples de systèmes conversationnels

Cette section présente un panorama des recherches entreprises dans les systèmes conversationnels. Différentes approches de conception de systèmes conversationnels ont été proposées dans le passé. Nous décrivons ci-après certains de ces systèmes. La plupart des systèmes ont été sélectionnés de façon à privilégier ceux qui sont orientés vers les agents artificiels et/ou les SMA. Nous allons démarrer par le système SPA (Nguyen et Wobcke, 2005), qui a pour but de proposer un agent capable de mener un dialogue entre l'utilisateur et plusieurs agents assistants. Ensuite, nous passons au système TRIPS (Allen et al., 2001a), très connu dans le domaine. Après, nous allons étudier le système HACTAR (Lebarbé, 2002), qui propose un SMA pour l'analyse syntaxique des énoncés. Ensuite, c'est l'approche présentée par (Gaiffe et al., 2004) qui nous intéresse, car d'une part le système est censé gérer plusieurs agents applicatifs (similaires à nos agents de services) et d'autre part, le système de coordination est décentralisé. Un autre système ici présenté est l'agent émotionnel de (Bisognin et al., 2004), représentant des systèmes qui utilisent les agents du type BDI. Ensuite, nous présenterons le système ARTIMIS (Agent Rationnel à base d'une Théorie de l'Interaction mise en œuvre par un Moteur d'Inférence Syntaxique) (Sadek et al., 1997) développé à France Telecom et utilisé dans plusieurs applications commerciales. Finalement, nous allons présenter le système NESPOLE! (Taddei et al., 2002). Ce système n'est pas orienté agents, mais propose une interface multimodale et utilise la théorie de actes de langage pour l'analyse syntaxique et l'analyse sémantique des énoncés.

7.1. Le système SPA

Le SPA (*Smart Personal Assistant*) (Nguyen et Wobcke, 2005) est un système à base d'agents, capables de gérer un dialogue en langage naturel parlé entre l'utilisateur et une collection d'agents assistants spécialisés dans des tâches comme la gestion du courriel ou la gestion d'un calendrier. La Figure 34 illustre la façon dont les agents assistants sont coordonnés par un agent fédérateur : l'agent *Dialogue Manager*.

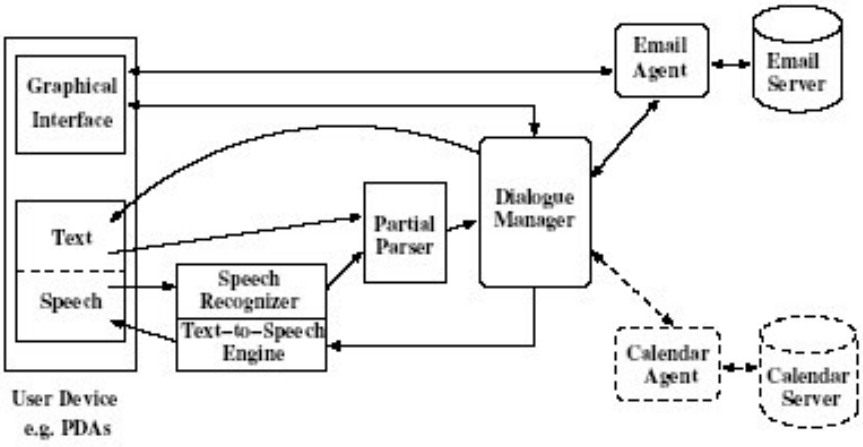


Figure 34 : L'architecture du système SPA (Nguyen et Wobcke, 2005).

Il faut préciser que leur conception d'agent assistant est différente de la nôtre, car pour nous chaque utilisateur a son agent assistant et les tâches sont réalisées par des agents de services. Pour eux, chaque utilisateur peut avoir plusieurs agent assistant, qui n'interfacent pas directement avec l'utilisateur (doivent passer par un agent de contrôle).

L'agent de contrôle du dialogue, basé sur le modèle BDI, a été développé en JACK (*Java Intelligent Agents* (Howden et al., 2001)) comportant les plans suivants :

- identification des actes conversationnels ;
- identification des intentions de l'utilisateur ;
- réalisation de tâches (en les délégants aux agents assistants) ;
- génération de réponses.

En ce qui concerne l'identification des actes conversationnels et des intentions de l'utilisateur, le modèle s'appuie sur les actes de langages de (Searle, 1975) et sur un dictionnaire de termes du domaine.

Les tâches sont réalisées par les agents assistants toujours à la demande de l'agent de contrôle. La communication directe entre les agents assistants n'est pas prévue.

Les auteurs du projet indiquent qu'un prototype du système a été réalisé avec l'agent de contrôle et un seul agent assistant. Ainsi, les modules de coordination du système n'ont pas encore été mis à l'épreuve. D'ailleurs, l'architecture décrite ne comporte pas de véritable politique de présentation et d'affichage d'informations, ce qui est impératif lorsqu'on a plusieurs sources d'informations.

7.2. Le système TRIPS

La Figure 32 montre le système TRIPS (*The Rochester Interactive Planner System*) (Allen et al., 2001a). Il s'agit d'une architecture modulaire, construite pour la planification collaborative. Le trois modules de contrôle : le *Interpretation Manager*, le *Generation Manager* et le *Behavioral Agent* sont appuyés par une dizaines de modules outils: reconnaissance vocale, analyseur grammatical, extracteur d'actes de dialogue, résolution de référence, planificateur, agenda, etc. L'interaction de ces modules est faite par l'envoi de messages. Le TRIPS est utilisé dans des projets d'agents conversationnels dans des domaines pratiques (projet collaboratif d'une cuisine ou recherche d'information).

Le système propose un niveau de traitement grammatical générique et indépendant de l'application. En revanche, dans la phase d'interprétation les connaissances spécifiques requises pour la tâche particulière sont utilisées. Les modules génériques ne manipulent que des entités ayant pour type : objectifs, solutions, ressources, situations. Ceci limite la portée du système, qui ne peut être adapté qu'à l'intérieur des problématiques de planification collaborative.

7.3. Le système HACTAR

Lebarbé (2002) propose un système basé sur les agents, pour faire l'analyse syntaxique, qui permet de paralléliser et de faire coopérer les processus de calcul et les unités linguistiques. La première étape du traitement consiste à *agentifier* chacun des mots de la phrase : chaque mot est intégré par un agent (Figure 35). Une des tâches incombant aux agents est de tenter d'identifier les propriétés des mots qu'ils incarnent.

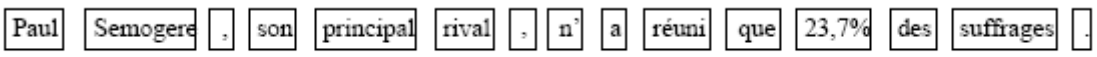


Figure 35 : Les mots sont « agentifiés »

Les agents ensuite communiquent et « s'associent » mutuellement générant de nouveaux agents, qui s'associeront jusqu'au moment où la phrase va être totalement traitée. Une fois l'analyse achevée, il ne reste plus qu'un agent.

Dans ce projet, l'auteur a utilisé le modèle d'agent APA (Architecture de Perception Augmentée) modélisé par (Girault, 1999), qui a été proposé pour être utilisé dans le cadre d'un SMA pour la RoboCup. L'agent APA est non seulement un agent s'insérant dans un système, mais aussi un SMA à part entière.

Nous pensons qu'une distribution du traitement des énoncés, comme dans le projet HACTAR, peut créer des parallélismes qui retardent et compliquent encore plus leur interprétation. Le traitement d'un énoncé est une tâche séquentielle et, ne justifie pas une telle distribution. Pour finir, comme remarqué par (Ottens et al., 2004), il manque dans cette approche une gestion du contexte, fondamentale dans une interface conversationnelle.

7.4. Le Projet OZONE

Dans le cadre du projet OZONE, Gaiffe et collègues (2004) ont travaillé à la conception d'un système de dialogue capable de comprendre et d'interpréter les énoncés d'utilisateur dans un contexte multi-applicatif (architecture montrée dans la Figure 36). En termes de but de projet, cette architecture est la plus proche de notre travail.

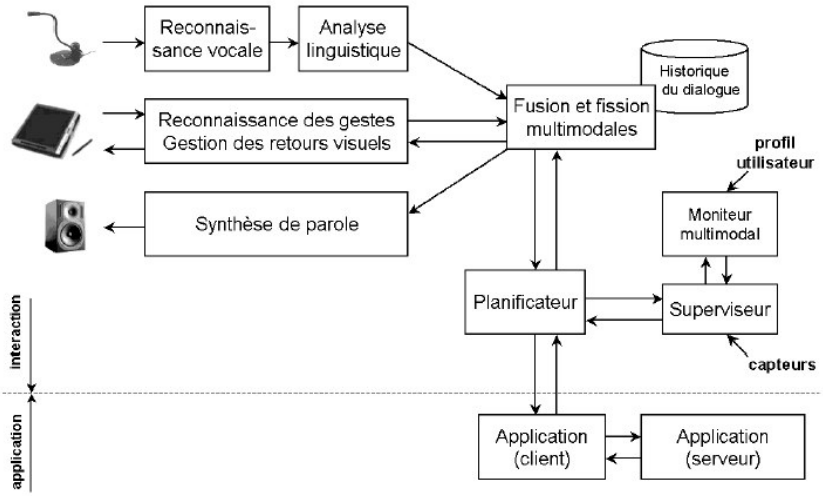


Figure 36 : L'architecture du démonstrateur OZONE (extrait de (Gaiffe et al., 2004)).

Le but est de définir un modèle d'architecture permettant une séparation entre le système de dialogue et les spécificités des tâches applicatives. Pour parvenir à résoudre le problème ils proposent un gestionnaire de dialogue composé de trois agents : un agent « linguistique », un agent « collaboratif » (pour contrôler le contexte du dialogue) et un agent « applicatif » (lié à l'application). Dans leur implémentation, chaque application doit envoyer au gestionnaire du dialogue une ontologie de domaine et une grammaire.

L'affectation d'un agent applicatif lié à chaque application nous paraît tout à fait correcte, similaire à l'utilisation d'un agent de service. En revanche, ce qui nous paraît contraignant est la distinction de rôles entre l'agent linguistique, chargé du contrôle de l'interaction avec l'utilisateur, et l'agent collaboratif, chargé du contrôle du dialogue, cohérence et contexte. Dans une application pratique, l'utilisateur peut changer leurs buts et fournir de nouvelles commandes

à tout moment. Le fait que l'agent collaboratif doive être « alerté » de chaque changement par l'agent linguistique peut entraîner des problèmes de coordination. Dans une situation extrême, nous pouvons nous retrouver avec deux agents ayant des buts contradictoires.

7.5. Une architecture d'agent émotionnel

Bisognin et collègues, (2004) proposent un prototype d'agent émotionnel, dont l'architecture est capable de tenir compte des émotions (comme la joie, de dégoût ou la peur) dans l'interaction avec l'utilisateur. L'architecture est composée de plusieurs modules (voir Figure 37). Le module de perception perçoit, formalise et évalue l'environnement (la situation, les agents, mais aussi des énoncés et leurs caractéristiques non-verbales : expression faciale, intonation, posture, ...). Le module de génération et de révision des croyances ajoute à la base de connaissances de nouvelles croyances générées à partir des données reçues du module de perception. Enfin, un module d'action traduit l'acte résultant dans les diverses modalités de l'agent.

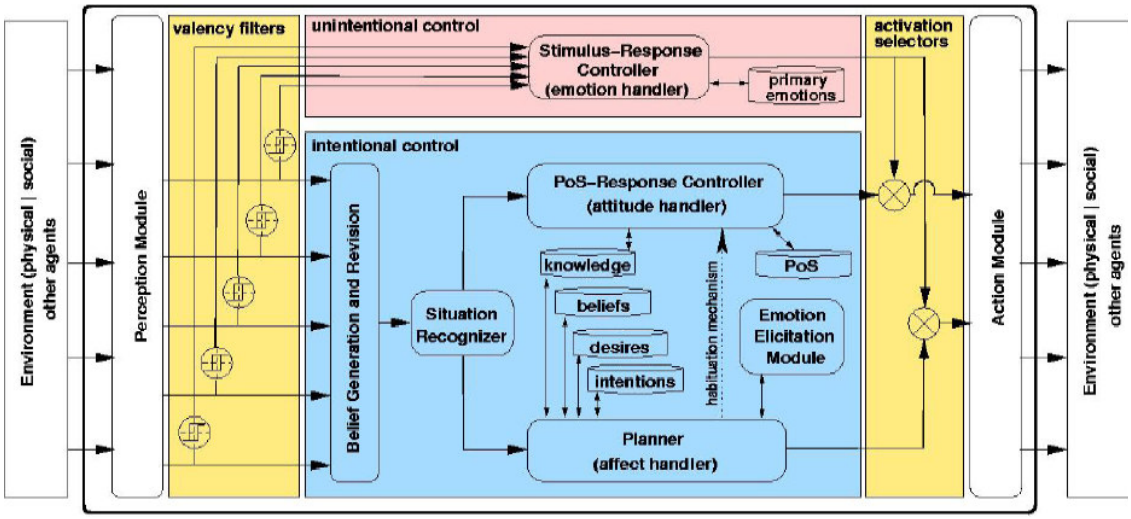


Figure 37 : Architecture d'un agent émotionnel (extrait de (Bisognin et al., 2004)).

L'architecture de l'agent repose sur le modèle BDI, qui ne fait pas son chemin dans les applications d'agent assistant, car elle n'est pas adaptée à l'intégration des mécanismes d'apprentissage et d'adaptation du comportement au sein d'un agent, et ne permet pas de modéliser la composante sociale d'un SMA (les interactions avec les autres agent).

7.6. Le système ARTIMIS

Développé par France Telecom, ARTIMIS (Agent Rationnel à base d'une Théorie de l'Interaction mise en œuvre par un Moteur d'Inférence Syntaxique) (Sadek et al., 1997) définit les

principes d'un agent rationnel dialoguant. Cet agent doit avoir des capacités de réaction coopératives, et être également capable de prendre des initiatives. Pour cela, il se base sur les actes de langage mis en œuvre dans des plans rationnels, exploitant le cadre formel de la logique pour les attitudes mentales (tel que croyance et intention) et l'action. Sadek (1999) propose une théorie formelle de l'interaction par la communication, qui est basée aussi sur les notions de croyances et d'intentions. ARTIMIS essaie d'inférer les intentions de son interlocuteur et d'agir en conformité. Le langage de communication pour dialoguer avec l'utilisateur, ARCOL, est le précurseur de FIPA-ACL, le langage de communication retenu par FIPA.

Les systèmes comme ARTIMIS, passent par l'identification des états psychologiques intentionnels. Le problème qui se pose lors de l'implantation est la gestion de formes logiques correspondant à ces états et pour lesquelles sont à définir des mécanismes de déduction et de présupposition.

7.7. Le système NESPOLE!

NESPOLE! (*Negotiation through SPOken Language in E-commerce*) (Taddei et al., 2002) est un projet qui propose une interface multimodale conçue pour fournir des services de traduction automatique de la parole dans le cadre d'applications de commerce électronique. L'architecture générale du système est montrée sur la Figure 38.

Pour démontrer les fonctionnalités de l'interface, un prototype a été développé et testé au sein du bureau de tourisme de la ville de Trentino, en Italie. Dans ce prototype, un utilisateur parlant anglais, français ou allemand, peut poser des questions, en ligne et en temps réel, à un opérateur humain situé en Italie, qui ne parle que l'italien.

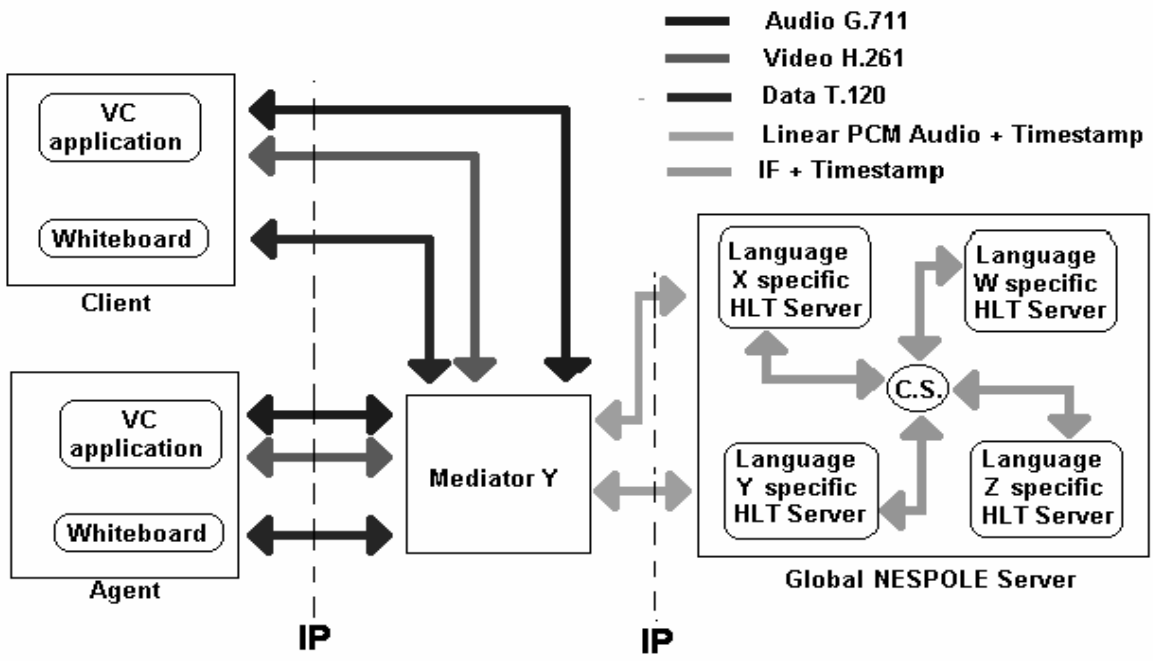


Figure 38 : L'architecture du système NESPOLE! (extrait de (Taddei et al., 2002)).

L'approche de traduction est fondée sur un langage pivot ou *Interchange Format* (IF), qui représente le sens de la phrase indépendamment de la langue (Vu-Minh et al., 2004). Le système est capable de traduire et synthétiser la parole dans les deux sens. À chaque énoncé d'utilisateur, le système génère une représentation intermédiaire, créée à partir d'un acte de langage reconnu et des concepts du domaine. Les concepts sont sub-divisés en attitudes, prédicats principaux et participants du prédicat.

8. Pourquoi une nouvelle architecture ?

Après avoir listé les composants d'une interface conversationnelle et quelques systèmes existants, une question incontournable peut apparaître : pourquoi proposer une nouvelle architecture pour l'interface conversationnelle ? Nous avons bien cherché une interface conversationnelle, adaptable à toutes les fonctionnalités fondamentales de notre application. Malheureusement, nous ne l'avons pas trouvée. Les architectures disponibles entraîneraient un gros investissement sans garantie de résultats satisfaisants. Les principaux défauts trouvés sont :

- le manque d'une véritable politique de présentation et d'affichage d'informations, ce qui est impératif lorsqu'on a plusieurs sources d'informations ;
- la distribution du traitement des énoncés, comme dans le projet HACTAR, créant des parallélismes dangereux ;

- la distribution de la coordination du dialogue entre plusieurs agents, permettant dans une situation extrême que deux agents aient des buts contradictoires ;
- plusieurs d'entre ces systèmes sont fondés sur le modèle BDI, qui n'est pas adapté à l'intégration des mécanismes d'apprentissage et d'adaptation du comportement au sein d'un agent ;
- l'impossibilité d'utiliser les ontologies comme forme de représentation et source de connaissances.

Par ailleurs, aucun de ces systèmes ne propose véritablement une interface dédiée aux agents assistants. La conception d'une telle interface reste inédite.

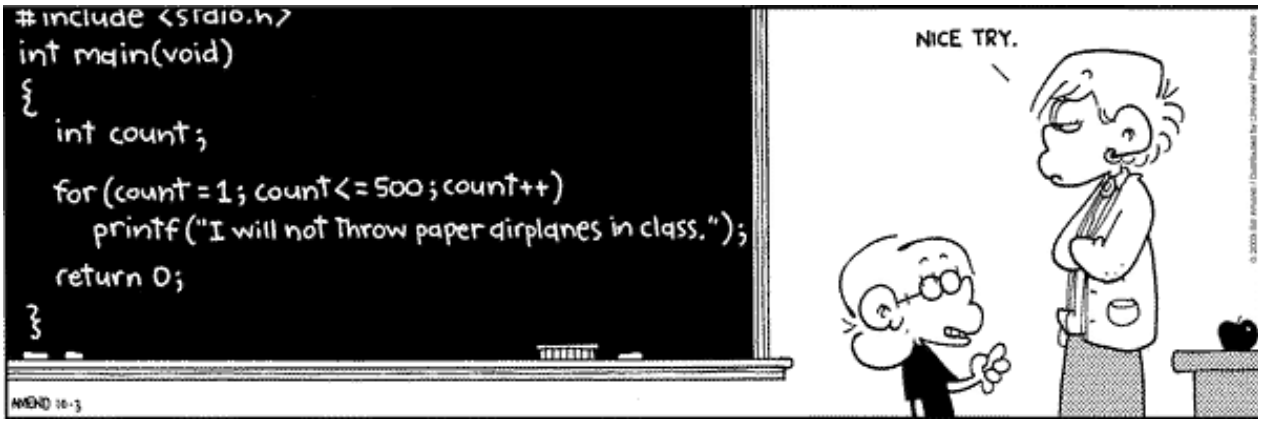
Nous allons présenter au cours du prochain chapitre la proposition d'une interface conversationnelle appropriée et dédiée aux agents assistants.

9. Synthèse et conclusions de ce chapitre

Dans ce chapitre, nous avons identifié et introduit les principaux composants nécessaires à la mise en œuvre d'une interface conversationnelle classique. Nous avons présenté différentes approches de conception de systèmes conversationnels. Nous avons montré qu'à ce jour, il n'existe pas d'architecture complète dédiée aux agents assistants telle que celle que nous proposons. Nous avons listé plusieurs systèmes différents, en remarquant leurs atouts et leurs faiblesses.

Le prochain chapitre présente la nouvelle interface en détails.

CHAPITRE 4



ICAI : Interface conversationnelle pour une aide intelligente

Nous présentons dans ce chapitre l'architecture de l'interface conversationnelle qui nous permettra de réaliser la nouvelle interface conversationnelle de l'agent assistant. Le chapitre aborde toutes les étapes de conception de l'architecture, en prêtant une attention spéciale à la place des ontologies dans le traitement syntaxique et sémantique. Nous introduisons nos contributions à l'évolution du modèle de l'agent assistant, comme la nouvelle politique d'affichage d'informations et la façon dont l'interface est couplée à l'agent.

1. Problématique

Tout au long de cette thèse nous étudions la conception d'une interface conversationnelle pour les agents assistants personnels. Notre travail se place dans un processus de développement d'un modèle générique d'agent assistant personnel initié par Ramos (2000) et poursuivi par Enembreck (2003). Malgré la cohérence des modèles proposés par Ramos, et à la suite d'une première étude de la personnalisation de l'interaction avec l'utilisateur menée par Enembreck, nous avons estimé que l'interface de l'agent assistant méritait une étude plus approfondie. En conséquence, après avoir examiné les divers types d'interfaces utilisées par des agents, nous avons conclu que les interfaces conversationnelles étaient bien adaptées à nos besoins. Nous nous sommes donc focalisés sur la définition et la conception d'une interface conversationnelle pour un agent assistant, en tenant compte de plusieurs contraintes et hypothèses liées à l'agent lui-même, à son pilotage et à la nature du rapport utilisateur-agent, mais aussi aux caractéristiques des applications potentielles. Nous estimons que ce n'est qu'après avoir réalisé ce travail, et après doté l'agent de l'interface conversationnelle correspondante que le modèle générique de l'agent assistant pourra effectivement être validé, puisque les implémentations passées ont privilégié le développement des autres modules de l'agent, comme par exemple, le module soi-même (Enembreck, 2003).

Pour illustrer ce qu'un utilisateur peut attendre de notre interface conversationnelle, considérons l'exemple de conversation du Tableau 6 (*AP* = agent assistant et *USR* = utilisateur).

AP (1):	Welcome to the Voice Personal Assistant.
USR (2):	Open the report on agents.
AP (3):	*AP rajoute le message « The report is open. » dans la file de message de basse priorité.*
AP (4):	You have a meeting scheduled for today: participant: Mike, time: 11:00, place: office, description: agent platform.
USR (5):	Send an email to Mike.
AP (6):	What is the subject?
USR (7):	Our meeting today.
AP (8):	What is the content?
USR (9):	We have a meeting today. Please bring your report on agent platforms.

AP (10):	You have a new email: sender: Mary.
USR (11):	OK, could you open my email account?
AP (12):	Yes.
AP (13):	*AP rajout le message « Email account loaded. » dans la file de message de basse priorité.*
USR (14):	List all meetings in my office tomorrow morning.
AP (15):	You have no meetings scheduled for tomorrow morning.
USR (16):	What is the email address of Mary?
AP (17):	The email is: mary@company.com.
USR (18):	Send an email to her and to Mike.
AP (19):	What is the subject?
USR (20):	Setting a meeting for tomorrow.
AP (21):	What is the content?
USR (22):	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?
AP (23):	You have a new email: sender: Paul.

Tableau 6 : Un exemple de dialogue.

Le contexte de cette conversation a été extrait du cas issu d'expérimentations du système de capitalisation et de gestion de connaissances, développé par Tacla (2003) à l'UTC. Dans son approche, les utilisateurs sont immergés dans un espace de travail collaboratif (un environnement de gestion des connaissances - EGC), où toutes leurs actions peuvent être prises en compte pour la capitalisation des connaissances, comme l'envoi d'un courriel, la rédaction d'un rapport, la recherche sur Internet. L'agent assistant et son staff d'agents sont capables de mettre à jour des mémoires personnelles au fur et à mesure de ses activités. Il est capable aussi d'identifier deux ou plusieurs personnes intéressées par les mêmes sujets et de les mettre en relation.

Étant donné ce contexte, supposons un groupe de trois personnes (John, Mike et Mary) travaillant dans le cadre d'un même projet. Chacun d'entre eux a son propre agent assistant capable d'accomplir les tâches décrites dans le paragraphe précédent. Le dialogue du Tableau 6 illustre une conversation entre John et son agent assistant. Après le message qui marque le début de la session (1), l'utilisateur demande à son agent assistant l'ouverture d'un rapport sur les

agents. Cette tâche est exécutée par un des agents de service. Une fois le rapport ouvert, John y consacre quelques instants. L'agent de service responsable de la gestion de l'agenda, envoie un message d'alerte pour l'informer de la proximité d'une réunion (4). Ensuite, John décide d'envoyer un courriel à Mike pour lui demander d'apporter le rapport qu'il prépare sur les plates-formes d'agents (5-9). En interprétant le contenu du message, l'agent assistant constate que le sujet du message, les plates-formes d'agents, a un intérêt pour un autre participant du groupe de projet : Mary. Sans que Mike en soit averti, son agent assistant notifie l'agent assistant de Mary qui à son tour imprime un message d'information. Mary rédige alors un courriel à John pour lui proposer une présentation de ses activités. Une fois le message reçu, l'agent assistant de John l'informe (10) tout de suite. John pose une question à l'agent assistant (11) qui lui répond affirmativement. On note la stratégie coopérative du dialogue, qui mène l'agent assistant à répondre à la question posée et tout de suite après, à ouvrir la boite à lettre de John. À travers l'énoncé (14) l'utilisateur cherche à savoir s'il y aura des réunions le lendemain matin. La réponse arrive et l'agent assistant l'imprime immédiatement. À ce stade du dialogue, John cherche à savoir s'il peut proposer une réunion entre Mary, Mike et lui même. John demande à son agent assistant s'il connaît l'adresse électronique de Mary. En recevant une réponse positive, John écrit un courriel à Mike et à Mary. Au cours de son travail de saisie du courriel (18-22), un message d'alerte arrive (23) annonçant un nouveau courriel. Le message restera dans la file d'attente jusqu'au moment où la pile de tâches sera vide, ce qui va se produire après l'envoi du courriel que John est en train d'éditer.

Ce dialogue est un exemple typique des conversations possibles entre le maître et son assistant. Il illustre le potentiel de l'interface conversationnelle appliquée dans le cadre des systèmes de capitalisation de connaissances qui exploitent la saisie des traces des activités de l'utilisateur, plus facilement gérables que les interfaces purement graphiques. À la fin de ce chapitre, nous retournerons à cet extrait de conversation pour détailler comment notre interface et l'agent assistant l'interprètent et donnent suite aux demandes de l'utilisateur.

La conception et la réalisation d'une interface capable de mener des conversations comme celle ci-dessus, dans le cadre d'un agent assistant, n'est pas évidente, comme nous l'avons signalé lors du chapitre précédent. Les principaux défis liés au développement d'une telle interface comprennent les éléments suivants :

- la conception d'un mécanisme d'analyse syntaxique robuste, capable de traiter des énoncés parlés, pas forcément bien reconnus par le moteur de reconnaissance vocale et parfois mal formés syntaxiquement ;

- la conception d'un analyseur sémantique, capable d'interpréter les énoncés, pas toujours liés au domaine de l'application ;
- la conception d'un gestionnaire de dialogue capable de gérer des conversations sur des domaines spécifiques, capable de déclencher plusieurs tâches simultanément et capable de bien gérer l'arrivée de messages provenant de plusieurs sources différentes (d'autres agents) ;
- la mise en place d'une structure de mémoires, capable de fournir des informations à la gestion du contexte de la conversation ;
- la conception d'une politique d'affichage d'informations, gérée par l'agent assistant, pour organiser la façon d'interrompre l'utilisateur : pour lui poser des questions et pour lui présenter des informations diverses, comme des résultats d'exécution des tâches ;
- la capacité d'ancrer le raisonnement de l'agent sur les ontologies dont l'agent dispose.

Nous espérons montrer dans ce chapitre que l'interface conversationnelle de l'agent assistant que nous avons conçue comporte des éléments suffisants pour faire face à ces défis.

2. Nos propositions et nos objectifs

Nous estimons qu'une interface conversationnelle spécialement conçue pour l'agent assistant, simplifiera son pilotage, entraînant une amélioration de la qualité de l'aide apportée par rapport aux interfaces classiques purement graphiques. Ceci diminuera la charge cognitive de l'utilisateur, puisque le travail d'interprétation et de reconnaissance des intentions de ce dernier devient la responsabilité de la machine. Nous postulons aussi que le comportement intelligent de l'agent peut contribuer à l'amélioration de la performance liée à une telle interface, grâce à une politique d'affichage d'informations. Pour cela, nous proposons le concept d'interface conversationnelle pour une aide intelligente - ICAI :

« Une interface conversationnelle pour une aide intelligente est le résultat de la conjonction d'un mécanisme conversationnel en langage naturel parlé et de la gestion intelligente de ce mécanisme, permettant le déroulement d'un dialogue coopératif et capable de gérer le déclenchement de plusieurs tâches à la demande de l'utilisateur, avec un minimum d'effort de la part de ce dernier. »

Nous insistons sur l'utilisation du terme « gestion intelligente » dans la définition présentée. Le rôle « intelligent » de l'interface est joué par l'agent assistant, en s'appuyant sur sa capacité de gestion et de coordination d'une politique d'affichage d'informations, d'exploration d'une ontologie du domaine, entre autres.

La mention « intelligente » relève du besoin évident d'une interface qui se comporte intelligemment. Pour cela, nous listons ci-dessous les éléments que nous croyons déterminants pour construire une interface intelligente :

- Adaptation à l'utilisateur (*User Adaptativity*): permettre que l'interaction (utilisateur X assistant) soit adaptable à différents utilisateurs en différentes situations (dans notre cas, cela se fait grâce notamment aux agents du staff, qui réalisent des tâches propres à chaque utilisateur, en fonction du modèle de l'utilisateur) ;
- Modèle de l'utilisateur (*User Model*): maintenir des connaissances et préférences de l'utilisateur ;
- Langage Naturel: interagir avec les utilisateurs en langage naturel écrit ou oral au clavier;
- Gestion de dialogue: maintenir un dialogue avec l'utilisateur, en tenant compte du contexte ;
- Politique d'affichage d'informations : gestion des interruptions et des impressions de messages, minimisant les interventions de l'utilisateur.

L'intelligence de l'interface est garantie par une politique d'affichage adaptée (développée dans la section 5.6), par une stratégie coopérative de gestion du dialogue et par une gestion des énoncés, à travers un numéro limité d'actes de langage.

Nous mettons l'accent sur l'utilisation des ontologies comme support à l'analyse syntaxique et sémantique, vu que les ontologies sont déjà présentes dans le modèle générique de l'agent. Les ontologies sont le fil conducteur de nos travaux, puisque d'une part elles nous aident au cours de toutes les interprétations des messages, provenant de l'utilisateur ou d'ailleurs. Les ontologies sont la seule forme de représentation de connaissances dont nous avons besoin. Les ontologies sont une contrainte transformée en atout.

Dans notre thèse de départ, l'agent assistant est inséré dans un environnement professionnel où les individus (les agents humains) effectuent des activités de plus en plus complexes, dans un délai de plus en plus court. Notre rôle est de prendre en compte ces contraintes en proposant un système capable de rendre les activités de l'utilisateur plus aisées. Comme le soutient Ramos, ce concept d'insertion nous indique que la relation entre maître et assistant sera forcement basée sur :

1. un couplage fort entre utilisateur et assistant, fondamental pour une bonne adaptation de l'assistant à son utilisateur. Une assistance effective est toujours basée sur l'identité entre les deux parties et sur un certain degré de connaissances partagées (habitudes, préférences, jargon commun, etc.) ;
2. la communication, base d'un processus de collaboration surtout dans la relation utilisateur-assistant, ce qui nous amène à privilégier la communication en langage naturel ;
3. la distribution de l'exécution des tâches spécifiques entre des agents de service, qui va spécialiser l'agent assistant uniquement dans la tâche d'assistance à son utilisateur.

2.1. Notre contribution

Notre groupe de recherche a récemment expérimenté l'utilisation du langage naturel comme forme d'entrée des informations. Ces expérimentations, bien que préliminaires et superficielles, ont montré un bon potentiel. Nous voulons dans cette thèse aller de l'avant, proposant la conception d'une nouvelle interface, où les entrées se font en langage parlé, à travers un mécanisme conversationnel. Notre principale contribution est la conception de l'interface conversationnelle pour une aide intelligente, qui contiendra :

- un mécanisme propre d'analyse syntaxique, basé sur les grammaires de liaisons ;
- un système de dialogue basé sur les actes de langage directifs (ordre, question et réponse) ;
- une mémoire de conversation basée sur des instances de concepts ontologiques ;
- une nouvelle politique d'affichage d'informations ;

Cette interface est aussi basée sur :

- l'adoption d'une stratégie coopérative pour le système de dialogue ;

- l'ancrage sémantique à travers les ontologies ;
- la séparation physique des connaissances de domaine et des tâches.

2.2. Nos objectifs

En implémentant la nouvelle interface, nous avons les objectifs suivants :

1. montrer la faisabilité de notre approche ;
2. montrer que la charge cognitive de l'utilisateur est diminuée par rapport aux interfaces graphiques traditionnelles, puisque le travail d'interprétation et reconnaissance des intentions est de la responsabilité de la machine ;
3. mesurer l'amélioration de la qualité de l'aide de l'agent assistant ;
4. apporter nos contributions à l'évolution du modèle de l'agent assistant.

Les sections suivantes présentent des études préliminaires que nous avons menées avant la conception de l'interface. Ensuite nous présentons notre proposition d'interface, que nous appellerons interface conversationnelle pour une aide intelligente - ICAI.

3. Les pré-requis et les hypothèses de départ

Dans cette section, nous présentons les pré-requis et les hypothèses de départ qui ont été prises en compte pour la conception de l'interface conversationnelle. Tout d'abord, nous allons présenter les pré-requis spécifiques fonctionnels de l'agent et ensuite nous allons lister les hypothèses de départ. Pour chaque pré-requis recensé nous précisons comment nous l'avons pris en compte.

3.1. Les pré-requis spécifiques fonctionnels de l'agent assistant

Nous avons présenté dans le Chapitre 2, le concept d'agent assistant et ses attributions. À cette liste d'attributions nous allons rajouter un certain nombre de pré-requis spécifiques propres aux agents assistants qui secondent un travailleur intellectuel dans son travail quotidien. Nous les classons en 2 groupes : les fonctionnalités liées à la gestion de l'environnement dans lequel l'agent assistant est immergé et les fonctionnalités liées à l'adaptation à l'utilisateur et le domaine de l'application. Cette liste de fonctionnalités nous a permis de baliser quelques uns de nos choix de conception.

3.1.1 Fonctionnalités liées à la gestion de l'environnement

En fonction de l'exécution distribuée et partagée des tâches, l'agent assistant doit gérer l'arrivée constante de messages, concernant l'utilisateur, provenant des agents de service. Comme conséquence, l'agent assistant doit être capable d'interrompre une conversation avec l'utilisateur pour lui poser une question ou pour lui présenter le résultat de l'exécution d'une tâche. Pour cela, le système de dialogue doit permettre des interruptions sans perdre le contexte. La solution que nous proposons est la mise en place d'une politique d'affichage d'informations (présentée dans la section 5.6) capable de classer ces messages selon leur degré de priorité, et la mise en place d'un système de dialogue « préemptif » (sans pour autant perdre le contexte de la conversation).

La délégation de l'exécution des tâches de la part de l'agent assistant entraînera aussi la gestion de temps d'exécution différents, obligeant l'agent assistant à maintenir le contrôle de plusieurs tâches qui, parfois, peuvent ne pas être terminées. Notre choix passe par l'intégration d'une structure de gestion de tâches, où chacune est traitée de façon indépendante.

3.1.2 Fonctionnalités liées à l'adaptation à l'utilisateur et le domaine de l'application

Le deuxième groupe de fonctionnalités est lié à l'utilisateur et à son environnent de travail. Il ne faut jamais oublier la concurrence de l'agent assistant par rapport aux autres applications de l'utilisateur : l'attention de l'utilisateur est partagée entre plusieurs « processus » différents. L'agent assistant doit « profiter » des moments d'attention et ne pas trop « déranger » l'utilisateur avec des énoncés inutiles. Ainsi, le système de contrôle du dialogue va prioritairement assumer une stratégie coopérative, limitant, autant que possible, les échanges avec l'utilisateur. Pour cela, nous avons limité l'espace des dialogues à ceux ne contenant que des actes de langage du type illocutoires et, parmi les classes proposées par (Searle, 1975), à la classe *Directives* : *inform*, *question* ou *answer*. D'ailleurs, le lexique de ce système de dialogue doit être enrichi avec les termes du domaine (« jargons »). De plus, comme signalé par (Yankelovich, 2001), les énoncés générés par le système doivent être les plus concis et informatifs possibles.

Grâce au mécanisme d'apprentissage (Enembreck, 2003) présent dans l'agent assistant, les préférences personnelles de l'utilisateur sont disponibles et peuvent être utilisées pour améliorer la performance de l'interface, en profitant par exemple des données extraites des conversations passées. Comme résultat, l'agent assistant pourra personnaliser ses actions en fonction du maître.

Finalement, l'interface conversationnelle doit être capable de se servir d'une ontologie comme source de connaissances du domaine de l'application. Cela conduit à accepter la structure de l'ontologie qui existe déjà. Dans ce chapitre, nous présentons une proposition de structure de l'ontologie pour l'interface conversationnelle à partir de l'ontologie du domaine préalablement écrite.

3.2. Les principes de base et les hypothèses de départ

Pour la conception de l'interface, nous avons étudié d'abord, la relation entre le maître et l'agent assistant dans le contexte des applications visées.

3.2.1 Les contraintes et les hypothèses de travail

Pour le design de notre approche il a fallu prendre en compte quelques contraintes et hypothèses liées à l'agent assistant lui-même, à son pilotage et à la nature du rapport maître-agent. Le but était toujours de développer un agent assistant qui améliore le travail de l'utilisateur.

Les contraintes liées à l'opération de l'agent assistant

Nous avons identifié trois contraintes principales :

- l'agent assistant est en concurrence avec toutes les autres applications pour capter l'attention de l'utilisateur ;
- l'agent assistant s'impose une stratégie coopérative, ce qui peut l'amener à poser des questions ou à démarrer une conversation avec son maître si une commande n'a pas pu être finalisée, ou si elle a été mal comprise, ou si l'agent assistant a besoin de nouvelles informations pour bien finir une tâche ;
- l'agent assistant peut alerter l'utilisateur pour signaliser un événement majeur (par exemple le résultat de l'exécution d'une tâche).

Nous discutons la politique d'affichage d'informations fortement influencée par ces contraintes plus loin dans ce chapitre.

Les contraintes liées au rapport agent-maître

En ce qui concerne le rapport agent-maître, nous avons identifié les contraintes suivantes :

- la nature de nos applications cibles nous guident vers un rapport « maître-esclave » entre le maître et son agent assistant, le maître contrôlant l'agent ;

- le maître fait son travail et, de temps en temps, interagit avec son agent assistant ;
- le maître est libre de changer le sujet de conversation, plusieurs fois au cours d'une session ;
- la dimension émotionnelle et affective du rapport est considérée comme étant moins importante, à cause du caractère strictement professionnel de nos applications (voir les travaux (Hyötyniemi et al., 2004) ou (Meyer, 2004) pour plus de détails sur ce domaine de recherche).

La limitation de l'espace de dialogue, la politique d'affichage d'informations, la stratégie coopérative de dialogue, entre autres, ont été intégrées à l'architecture finale en tenant compte ces contraintes.

Pour compléter cette analyse de l'environnement de travail, nous rappelons les critères de sélection des applications potentielles, déjà présentés dans le Chapitre 2 section 4. Pour ces applications :

- le domaine est limité et bien connu, comme par exemple la gestion des connaissances ou les environnements d'apprentissage ;
- les actions de l'utilisateur sont complexes et exigent une connaissance préalable du domaine ;
- l'utilisateur doit être guidé pour accomplir une tâche ;
- une interface graphique traditionnelle peut ennuyer l'utilisateur, à cause d'un grand nombre de composants tels que les menus, les sous-menus, les boutons, etc. ;
- l'utilisateur doit « mémoriser » les actions passées pour piloter l'application (selon Larson (2003), la mémoire humaine à court terme ne peut retenir qu'environ 7 actions passées pour la même tâche) ;
- un écran n'est pas toujours disponible ;
- l'utilisateur peut avoir des limitations physiques, comme les mains occupées ou un handicap moteur.

Étant donné ces hypothèses de départ, et du point de vue du concepteur de l'interface, la question que nous nous posons maintenant est de savoir quel rôle est dévolu à l'interface, par rapport à l'ensemble de l'agent assistant.

3.2.2 Le rôle de l'interface dans l'agent assistant

Parmi toutes les étapes de préparation de la conception de l'interface, il est fondamental de savoir quel est le rôle de l'interface dans l'agent assistant, c'est-à-dire, avoir une idée claire sur où commencent les compétences de l'interface et où elles s'arrêtent. Comme nous l'avons vu précédemment, l'interface est partie d'un ensemble, et doit agir sous le contrôle du module responsable du pilotage de l'agent assistant (voir Figure 10 du Chapitre 2). Ainsi le Tableau 7 délimite les compétences individuelles de l'ensemble : l'interface et « le reste » de l'agent assistant.

Attributions de l'agent assistant	**Attributions de l'interface**
Gestion de l'interaction avec les agents artificiels (agents de service)	Gestion de l'interaction avec l'utilisateur : saisie et interprétation des énoncés
Définition de la stratégie de dialogue	Gestion du système de dialogue
Contrôle de l'exécution des tâches	Contrôle de la mémoire de conversation
Gestion de l'ontologie	Gestion des fichiers du lexique
Gestion de la politique de présentation et d'affichage d'informations	Synthèse des résultats
Gestion du modèle de l'utilisateur	Exclusion des entrées inutiles
Apprentissage	

Tableau 7 : Les attributions de l'agent assistant et de l'interface.

L'échange d'informations entre l'interface et les autres éléments est géré par le module de contrôle de l'agent assistant, donc il n'y a pas de messages intra-agent.

Le comportement intelligent émerge de l'union de toutes les attributions ci-dessus listées. Ainsi comme la politique d'affichage d'informations est primordiale pour permettre à l'agent assistant de savoir quand et comment interrompre le maître, la gestion de la mémoire de la conversation est-elle aussi importante puisqu'elle permettra la gestion du contexte, permettant que l'utilisation de l'agent assistant devienne plus efficace aux yeux de l'utilisateur.

Avant de présenter l'architecture de l'interface conversationnelle, nous allons étudier et présenter les avantages et les limitations d'un système de dialogue développé par Ramos pour la plate-forme OMAS-WA basé sur des graphes de conversation. Cette étude nous fournira aussi des éléments pour la conception de note interface.

4. Le système de dialogue dans OMAS-WA

Le système de dialogue présenté ici est issu de la plate-forme OMAS-WA – « *Open Multi-Agent Systems with Assistants* ». OMAS-WA est la spécialisation de la plate-forme OMAS, aux agents assistants. La mise en œuvre de cette spécialisation sur le modèle d'agent générique dans OMAS a été développée par Ramos (2000).

La conversation entre le maître et son agent assistant, commence normalement par une demande d'information ou une assertion tapée par le maître. Le système est basé sur l'approche développée par (Chan-Lam, 1979) utilisant des graphes de conversation et des machines à états finis. L'idée principale est de disposer de plans déjà élaborés contenus dans une bibliothèque de plans. Ces plans vont constituer les compétences particulières de l'agent assistant et le problème principal va consister à choisir un plan particulier, qui donnera naissance à des actions, en fonction de ce que souhaite faire le maître.

Un graphe de conversation est un graphe orienté dont les nœuds correspondent à des états de la conversation et dont les arcs correspondent à des transitions (Figure 39). Dans l'approche de Chan-Lam chaque nœud est un système expert qui possède un certain nombre de règles, qui vont s'appliquer à une base de faits qui représentent le contexte de la conversation. Chaque nœud a pour objectif de recueillir certaines informations et lorsque celles-ci sont obtenues une transition s'effectue sur un noeud suivant jusqu'à arriver à un nœud qui marquera la fin de la conversation. Au moment de la transition, la base de faits est transférée au nœud suivant et les règles associées au nœud suivant s'appliquent sur celle-ci. La conversation débute à un nœud particulier du graphe. Les états du dialogue sont implantés à l'aide de MOSS (Barthès, 2005). Un modèle d'état a été défini. Un graphe de conversation est donc constitué d'une série d'instances d'états. Les méthodes exécutées lors du traitement de ces états sont associées aux instances des états. Un modèle de contexte a été également défini. Chaque instance d'état aura donc une instance de contexte associée pour chaque conversation. Les transitions ne sont pas explicitées, mais définies dans les méthodes. Au moment d'une transition, l'objet contexte est cloné, et la copie est attachée au nouvel état. Chaque état possède au moins une méthode qui s'appelle *execute*. Au moment d'une transition la méthode *execute* est invoquée sur l'instance d'état concerné. En général, un état possède une méthode *resume* qui est invoquée lorsque le maître fournit une réponse ou de nouveaux éléments d'information.

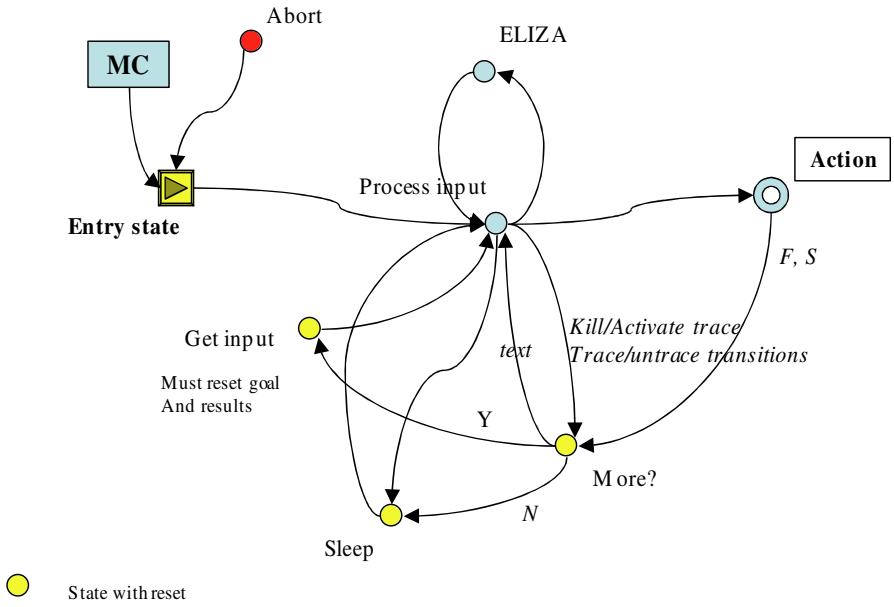


Figure 39 : Un graphe de conversation (Barthès, 2005).

4.1. Le déroulement du dialogue

Tout le processus d'interprétation des entrées passe pour l'identification des mots-clés dans chaque entrée tapée par l'utilisateur. Ces mots-clés, ainsi que l'identification du type de question, déclencheront les transitions dans le graphe de conversation. Un graphe de conversation est défini au préalable dans une étape de construction d'une base qui contient toutes les questions « possibles » pour le domaine d'expertise de l'agent assistant. Cette étape est cruciale puisqu'elle permettra de définir tous les dialogues traités par l'agent. Dans (Barthès, 2005) nous avons trouvé quelques exemples de questions dans le cadre d'un système de documentation en ligne concernant MOSS :

What is MOSS?

What is an entity?

What is an orphan?

How can I create a method?

How can I modify an object?

Le concepteur du système doit faire preuve de créativité en listant toutes les questions possibles. Il pourra aussi affiner sa base de dialogues par un processus de tests successifs, en invitant des utilisateurs extérieurs au domaine, qui poseront des questions au système. En

trouvant de questions non couvertes par la base active, il pourra les rajouter pour générer une nouvelle base.

Le graphe de conversation est composé de plusieurs états décrits en langage MOSS (en utilisant une *macro* Lisp appelée *m-defstate* – voir détails techniques dans (Barthès, 2005)). Un des états le plus important est le *Process Input* State (qui est précède de l'état *_q-GET-INPUT* responsable de la saisie de l'entrée). C'est lui le responsable du traitement de l'entrée. Il est formé par plusieurs slots dont *:transitions* qui contient les patterns applicables à l'entrée. Dans le pattern listé ci-après, le sous-dialogue *_what_conversation* (voir Figure 40) sera déclanché si les mots « what is/are » ou « explain/meaning/mean/means » sont trouvés.

```
(:patterns   ( ("what" "is" (?* ?y))
               ("what" "are" (?* ?y))
               ((?* ?x) "explain" (?* ?y))
               ((?* ?x) "meaning" (?* ?y))
               ((?* ?x) "mean" (?* ?y))
               ((?* ?x) "means" (?* ?y))
             )
             :keep ?y
             :sub-dialog _what-conversation
             :success _q-more? :failure _q-more?
)
```

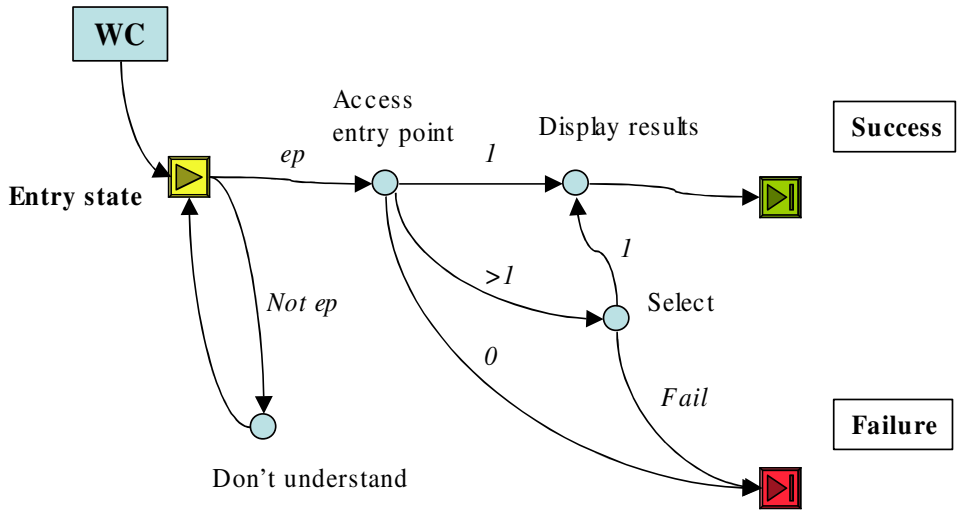


Figure 40 : Le sous-dialogue *What Conversation* (Barthès, 2005).

Le sous-dialogue est composé de plusieurs transitions qui arrivent dans l'un des deux états : *Sucess* ou *Failure*.

4.2. Les limites et les avantages de cette approche

La section précédente nous a présenté succinctement le système de dialogue présent dans la plate-forme OMAS-WA. De cette présentation ressortent quelques points de réflexions pour la suite de ce chapitre :

Ce système répond-il à nos besoins ?

Pour répondre à cette question, nous analysons le système de dialogue OMAS-WA dans les aspects liés au traitement de l'entrée, au contrôle et à la stratégie de dialogue et de gestion des entrées.

4.2.1 Traitement de l'entrée

Nous avons noté un certain inconfort lié au traitement des entrées, dû à l'interprétation approximative du système MOSS *Dialogs*. Il n'y a pas de grammaire pour interpréter les entrées ni pour classer les mots et les groupes de mots. En conséquence, des expressions plus « détaillées » ne sont pas facilement traitées, retardant le déclenchement d'une action identifiée. Une entrée du type « *List all reports on agents written by Mike Palmer* » pose des problèmes au système, car il serait capable d'identifier la tâche, mais il pourrait y avoir des difficultés pour identifier les paramètres présents (sujet : agents ou auteur : Mike Palmer). D'ailleurs ce point est vraiment important puisque certaines de nos applications cibles, comme les applications de gestion de connaissances (voir (Tacla, 2003)), utilisent la saisie des traces des activités de l'utilisateur, ce qui nous oblige à bien comprendre chaque expression donnée par l'utilisateur.

En fonction de cette approche basée sur une base de questions prédéfinies, l'ontologie de domaine reste sous-utilisée et reléguée à un deuxième plan, vu que les informations pour la sélection d'une action, sont déclarées dans la partie *:patterns* de chaque état. De plus, cette sous-utilisation de l'ontologie, nous paraît tout à fait dommage et surtout risquée car des contradictions pourront apparaître à un moment donné.

4.2.2 Contrôle et stratégie de dialogue

Le moteur de dialogue choisit une stratégie directive (selon la taxonomie présentée dans la section 6.1 du Chapitre 3) : il garde l'initiative pour conduire le dialogue, ce qui oblige l'utilisateur à accepter le schéma de conversation défini par le concepteur du dialogue. Cela a pour conséquence d'imposer une réponse réactive et de limiter ainsi la variété de ses stratégies. Nous pouvons le constater par l'analyse de la Figure 41, où à partir du point d'entrée (*Entry*

state), plusieurs chemins sont possibles, mais une fois la conversation engagée, le retour à l'arrière est impossible.

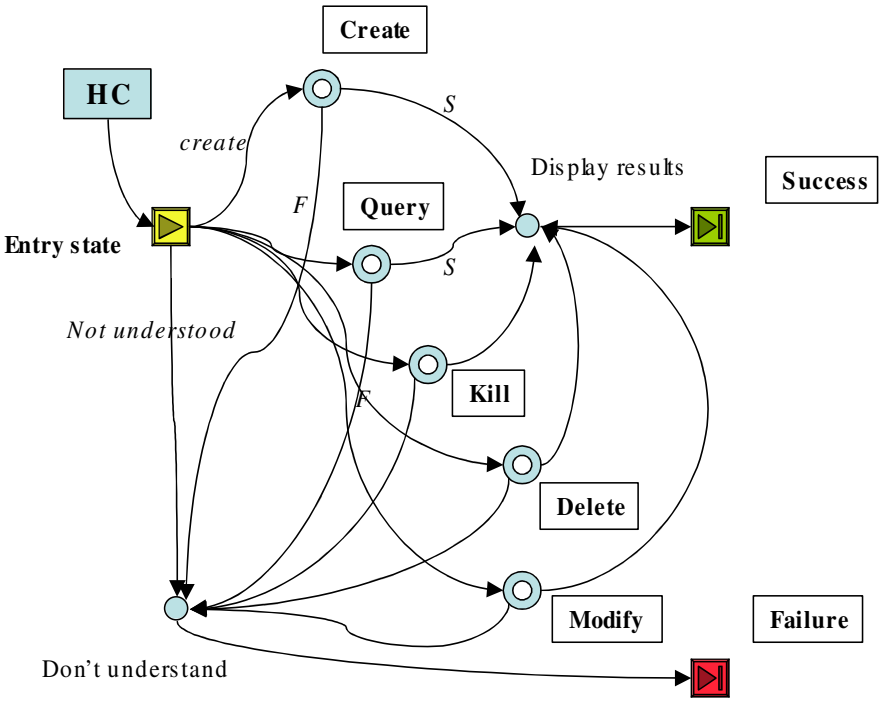


Figure 41 : Le sous-dialogue pour traiter une question du type *How* (Barthes, 2005).

Concernant le choix des états par le moteur de dialogue, l'enchaînement des états peut provoquer des boucles infinies. Si par erreur le concepteur du dialogue lie deux patterns de dialogue entre eux mêmes (par le champ *:sub-dialog* de *:patterns*), une boucle est créée.

4.2.3 Gestion des entrées

Nous avons compris dans le chapitre précédent l'importance du traitement de la référence pour la gestion d'un dialogue. Dans le système de dialogue OMAS-WA, la structure proposée dans ce modèle est appropriée : au moment de la transition, la base de faits est transférée au nœud suivant et les règles associées au nœud suivant s'appliquent sur celle-ci. Néanmoins, comme la version actuelle n'a pas de règles de traitement, il n'y a pas de gestion de contexte. La mise en place d'un tel mécanisme serait envisageable à condition d'inclure un analyseur lexico-syntaxique plus performant, pour faire face aux problèmes d'anaphore, d'ellipses, etc.

La réflexion menée jusqu'ici indique que ce système est plus approprié aux dialogues du type « questions fréquemment posées » (en anglais, FAQ - *Frequently Asked Questions*) ou même aux systèmes d'apprentissage basés sur ordinateur comme ceux proposés par Fried et collègues (2003) ou Cheng et collègues (2002), fondés sur une base de questions-réponses. Ce type de système possède une base de questions et de réponses enregistrées qui peuvent être

facilement exploitées par le moteur de dialogue. En revanche, si le nombre de questions est limité, l'utilisateur aura tendance à ignorer le système.

Du point de vue de l'implémentation, son langage de spécification d'un graphe de conversation, basé sur MOSS, permet une structure souple, contenant plusieurs objets ayant des rôles spécifiques, et un traitement d'exceptions efficace.

Avant de terminer cette section, un dernier point : que faire lorsqu'une ontologie n'est pas disponible ? Est-il plus facile de créer une ontologie ou de construire un système contenant toutes les interactions possibles ? Milward et Beveride (2003) ont posé les mêmes questions. Selon leur expérience il est préférable d'écrire une ontologie dans les cas où le traitement de la référence est nécessaire, ce dont nous sommes aussi partisans.

Le « cahier de charges » décrit au début de ce chapitre nous permet de dégager le besoin d'une autre architecture, capable de faire face aux conditions de fonctionnement de l'agent assistant et qui n'ait pas les faiblesses du modèle OMAS-WA que nous avons indiquées dans cette section. Les sections suivantes présentent donc la nouvelle l'interface conversationnelle de l'agent assistant.

5. Architecture générale de l'interface conversationnelle

Compte tenu des contraintes et des hypothèses de travail, nous avons développé une architecture générale de l'interface conversationnelle, montrée sur la Figure 42. La Figure 43 indique l'emplacement de l'interface conversationnelle dans le modèle de l'agent assistant.

L'architecture montrée ici est le fruit de toutes les considérations, fonctionnalités, hypothèses de travail, etc., indiquées précédemment. Dans cette section nous décrirons ces composants en fonction de leur place dans le flux de traitement des informations. Nous présenterons d'abord les aspects liés à la saisie de la parole pour ensuite progresser vers les mécanismes d'analyse sémantique et de gestion du dialogue (exactement comme nous avons fait dans le chapitre précédent).

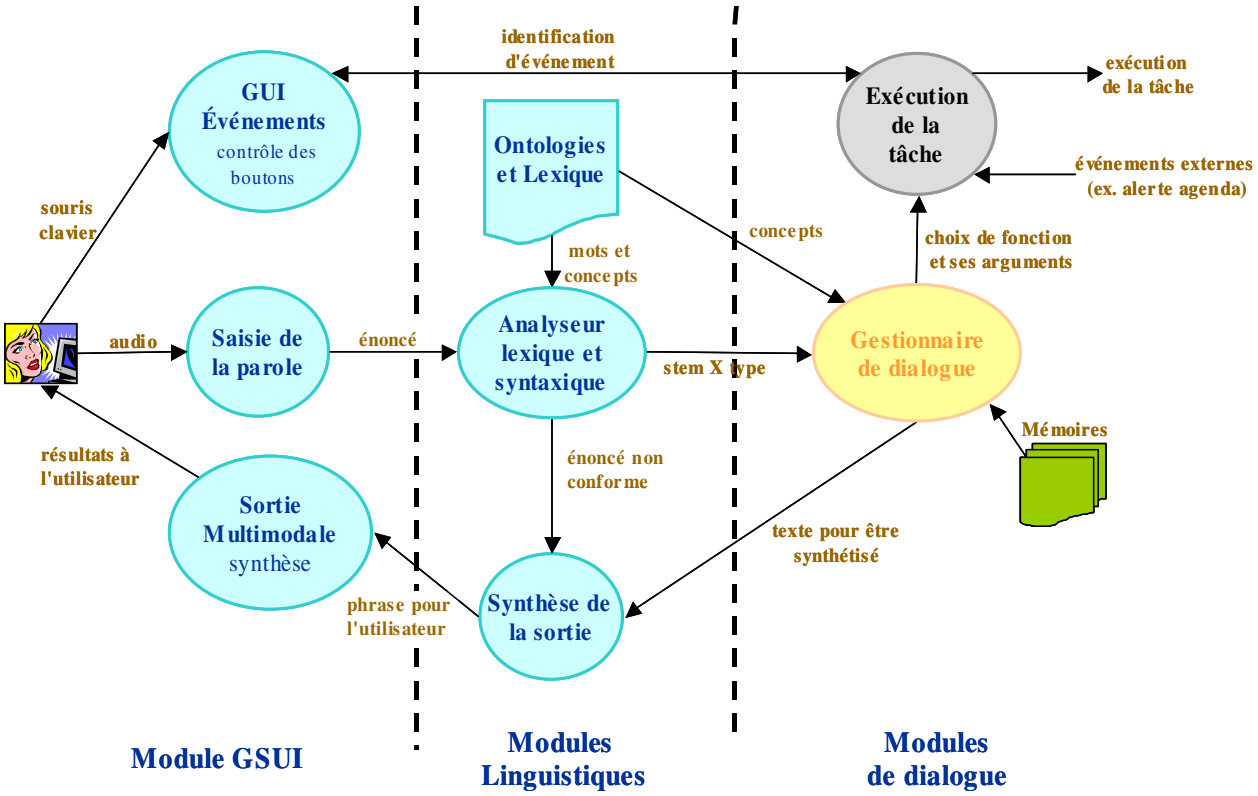


Figure 42 : L'architecture de l'interface conversationnelle.

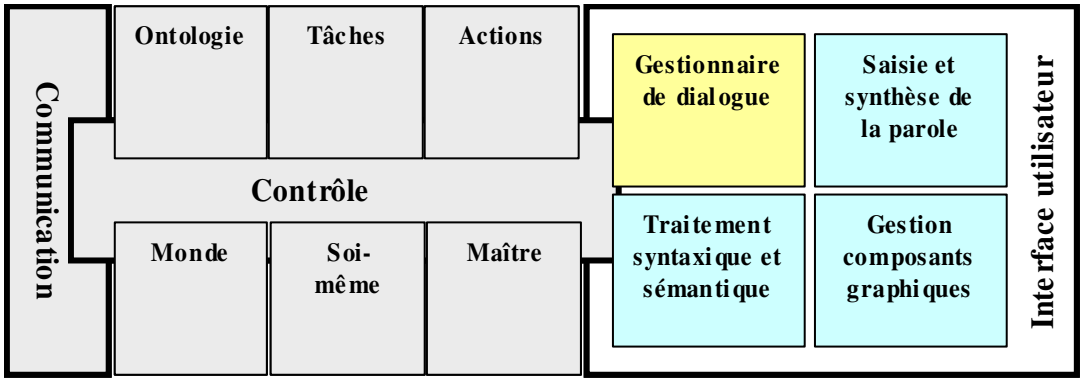


Figure 43 : L'agent assistant et son interface.

L'architecture de l'interface est composée de trois parties : une première partie (*Module GSUI*) responsable de la saisie et synthèse de la parole, ainsi que de la gestion des composants graphiques. Une deuxième partie (*Modules linguistiques*) responsable du traitement linguistique et de la compréhension. Enfin, une troisième partie (*Modules de dialogue*) est responsable de la gestion du dialogue.

Cette architecture centralisée nous permet de concentrer la gestion du dialogue dans un seul agent. Nous ne sommes pas partisans des architectures qui gèrent le dialogue de façon distribuée comme dans les travaux de (Lebarbé, 2002) ou de (Eriksson, 2001), tout simplement à cause du caractère séquentiel et dépendant des étapes du traitement.

5.1. La saisie et la synthèse de la parole

Nous avons souligné dans le chapitre précédent (section 2) la complexité du moteur de reconnaissance de la parole. Dans notre système nous utilisons un moteur de reconnaissance et de synthèse de la parole disponible commercialement (plus de détails dans le chapitre d'expérimentations et dans l'Annexe I). Ce moteur nous fournit la traduction de chaque mot prononcé par l'utilisateur. Notre système est responsable de l'enchaînement de ces mots, formant l'énoncé. Une fois l'énoncé acquis, démarre l'étape suivante de traitement linguistique.

Le processus de synthèse de la parole dans notre architecture est limité. Même si l'agent assistant est capable de synthétiser ses énoncés, ses réponses restent limitées à des phrases préalablement définies, comme des messages d'alerte, des résultats d'exécution d'une tâche et des messages provenant des agents de service.

5.2. Le traitement linguistique

La Figure 44 montre le flux et la relation existant entre les analyseurs impliqués dans le traitement de chaque énoncé.

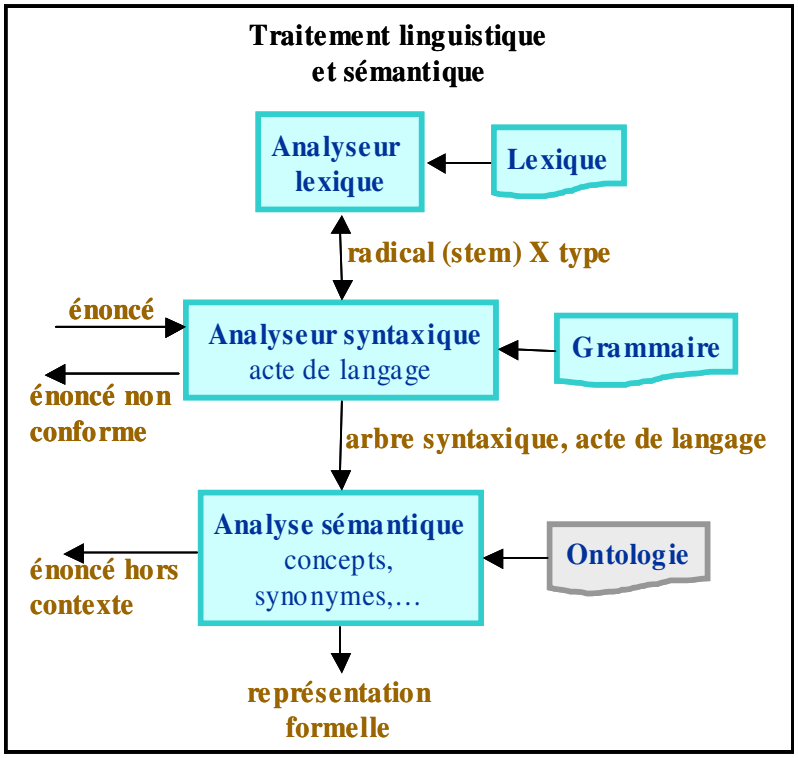


Figure 44 : Les étapes de traitement linguistique et sémantique.

L'énoncé saisi est envoyé aux modules d'analyse lexico-syntaxiques. Tout d'abord le mécanisme d'analyse syntaxique recherche une règle ou un ensemble de règles permettant la

construction de l'arbre syntaxique. À la sortie, chaque mot est classé selon son type et groupé selon sa position et ses interrelations (correspondant à une organisation hiérarchique de constituants). La Figure 45 montre le résultat pour l'énoncé :

USR (1): List all meetings at my office for the next three days

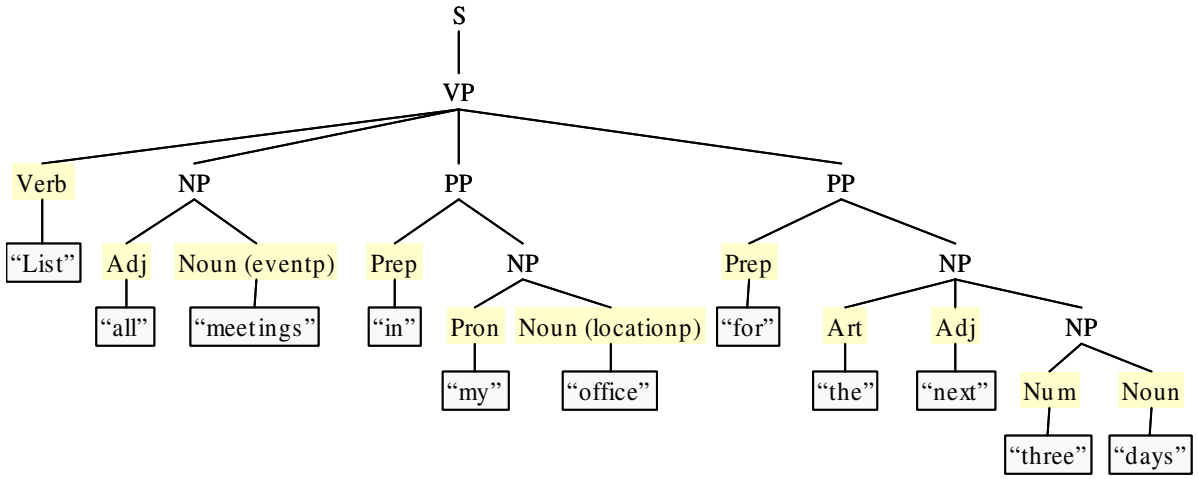


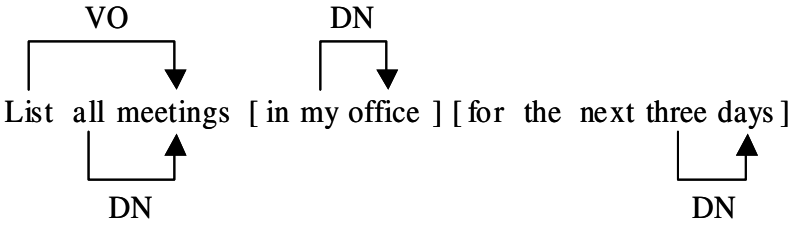
Figure 45 : L'arbre syntaxique résultant.

L'analyseur utilise une stratégie descendante, essayant de trouver les règles grammaticales qui combinées, permettent d'arriver aux feuilles. Cette grammaire ne contient pas de règles dépendant du domaine. Le lexique du système est composé de plusieurs fichiers contenant les mots groupés selon leur type (verbe, nom, article, etc.). Le lexique contient aussi quelques types spéciaux, quasi tous dérivés du type nom, comme par exemple un type groupant les mois de l'année (ces types seront mieux décrits dans le chapitre sur les expérimentations).

Pour enrichir cet arbre syntaxique, nous avons développé un mécanisme propre d'analyse syntaxique qui s'appuie sur l'algorithme *Link Parser* (Grinberg et al., 1995). Au départ, le *Link Parser* a été développé pour le langage écrit. Nous l'avons donc adapté au langage parlé. Une stratégie similaire a été développée par Antoine dans le système ROMUS (2003), dans le domaine de renseignement touristiques.

Nous avons « cassé » la rigidité de l'analyse faite par *Link Parser*, pour traiter les énoncés parlés. Nous avons réduit le nombre de liens et adapté l'analyseur pour qu'il accepte des énoncés contenant des expressions moins courantes à l'écrit. Le résultat de l'analyse est une suite de relations binaires, qui, assemblées, peuvent être visualisées sous forme d'un graphe. Cet algorithme est capable de fournir des informations précieuses comme des relations de quantités, de qualités, de genres, etc., entre les mots. Il identifie les constituants de l'énoncé, surtout les compléments des verbes, utiles dans l'identification des attributs des concepts. Pour l'énoncé de

l'exemple, nous obtenons la représentation succincte, montrée sur la Figure 46, qui est aussitôt annexée à l'arbre syntaxique de la Figure 45:



VO: verbe-objet
DN: déterminant-nom

Figure 46 : Les informations complémentaires à l'arbre syntaxique.

Notons que le système a identifié deux compléments, ici placé entre crochets. Grâce à ces compléments, l'analyse sémantique est facilitée puisque les compléments dégagent les attributs liés à l'objet en question, qui très probablement seront les attributs de la tâche à déclencher. Les énoncés suivants en sont d'autres exemples :

Who is the responsible [for updating the lotus database]?
Open the report [on agents] and send an email [to Mike]
Send an email [[to Paul] and [to Mike]]

La Figure 46 montre aussi les liens construits par l'analyseur. Nous avons défini quatre groupes des liens qui nous intéressaient plus particulièrement : Verbe-objet (VO), Déterminant-nom (DN), Adjectif-nom (AN) et Time-nom (TN). Dans le chapitre sur les expérimentations, ces groupes seront présentés en détails.

5.2.1 Les actes de langage

Après l'analyse syntaxique, nous trions les énoncés suivant une politique « d'encadrement » de l'espace de dialogue. L'objectif est de caractériser les buts de l'utilisateur en identifiant les actes de langage présents dans l'énoncé. Dans notre travail, nous nous intéressons plus particulièrement aux actes illocutoires, car sont eux qui plus facilement explicitent les intentions d'utilisateur. Nous avons restreint l'analyse des énoncés à trois catégories d'actes de langages : question, réponse et ordre (Paraiso et Barthès, 2004). L'analyseur syntaxique, en fonction des règles grammaticales utilisées pour analyser l'énoncé, va définir de quelle catégorie il s'agit. Comme nous l'avons montré dans la section 4 du Chapitre 2, un certain nombre d'échanges peuvent être évités, accélérant la conversation et l'obtention des résultats.

Les énoncés classés comme un ordre sont systématiquement envoyés au traitement sémantique, pour la construction d'une requête formelle. Les énoncés classés comme un ordre sont déclaratifs, contenant un verbe qui exprime la volonté de voir une action être exécutée. Dans le cadre de nos applications, en voici quelques exemples :

Search the latest version of my article on logics.
List all meetings for tomorrow morning.
Open my email account.
Please, send an email to Mike Palmer.

Pour les énoncés du type question, le travail est plus ardu, puisque elles peuvent être *directes* ou *conditionnelles*. Les questions directes mènent forcement à l'exécution d'une action et, par conséquent, à l'affichage des résultats. Par exemple, admettons la question suivante :

How many meetings do I have today?

Pour y répondre, le gestionnaire de dialogue aura obligatoirement besoin de construire une requête formelle et de questionner l'agent de service concerné, puisque lui-même n'est pas capable d'exécuter les tâches demandées par l'utilisateur.

Pour les questions conditionnelles, la réponse sera un *oui* ou *non*, suivi de l'affichage d'un résultat, selon le cas. Pour la question « Do I have meetings today? » le gestionnaire doit répondre par un oui suivi par les détails, ou par un non. Pour cela, un agent de service peut être consulté aussi. D'ailleurs, ce dernier pourra conditionner une réponse à une autre information provenant de l'utilisateur, ce qui entraînera le gestionnaire à poser une nouvelle question avant de répondre à la première question posée. Cette distinction entre les questions est détectée par l'analyseur syntaxique, qui sait quelles sont les règles grammaticales qui traitent les types différents existants.

Enfin, pour les énoncés classés comme réponse, le gestionnaire va d'abord essayer de trouver une question en attente de réponse. Dans ce cas, le gestionnaire de dialogue a un rôle crucial, car c'est lui qui connaît exactement la nature des dernières questions posées. Il y a les questions posées pour demander la saisie d'un paramètre d'une tâche en cours de remplissage. Il y a les questions de contrôle du mécanisme de dialogue, comme celle qui demande si l'utilisateur autorise ou non l'exécution d'une tâche. Et il y a aussi les questions provenant des agents de service. Dans tous les cas, le gestionnaire sait quel genre de réponse est attendu : un oui ou un non, un prénom, une adresse électronique, etc.

5.3. L'interprétation sémantique des énoncés à l'aide des ontologies

Comme nous l'avons indiqué dans le chapitre précédent, l'interprétation sémantique est un processus complexe. Plusieurs chercheurs ont appliqué des approches sémantiques, cherchant des mots-clés dans l'énoncé. Notre approche d'interprétation sémantique est basée sur la notion selon laquelle le sens d'un énoncé peut être déduit de la recherche de concepts et de leurs attributs dans chaque énoncé. Nous croyons que cette approche est plus indiquée pour des applications où le domaine est bien connu et assez restreint. De plus, notre approche peut aisément être utilisée pour d'autres langues que l'anglais, comme le français ou le portugais, puisqu'elle intervient après le traitement linguistique.

Nous utilisons les ontologies comme forme de représentation des connaissances du domaine. Dans notre architecture, les ontologies jouent deux rôles principaux. En particulier elles servent :

i) à aider l'interprétation du contexte des messages envoyés par d'autres agents ou par l'utilisateur, et;

ii) à garder une représentation informatique des connaissances.

La gestion des connaissances du domaine est cruciale pour la réussite de notre agent assistant. Les ontologies ont été choisies pour cela, et nous préconisons que :

« La spécification d'une ontologie dans un domaine particulier doit couvrir le plus précisément possible, le monde qui entoure l'utilisateur, en termes des entités et de leurs relations. De plus, nous soutenons que l'ontologie du domaine peut faciliter le processus d'interprétation syntaxique et sémantique, en fournissant des éléments linguistiques à l'analyseur lexico-syntaxique, comme les catégories sémantiques : synonymie, hyponymie/hyperonymie, entre autres. »

Même si la construction d'une ontologie n'est pas l'objet principal de ce travail, nous avons dégagé les éléments minimaux qu'elle doit contenir pour être utile dans l'interface conversationnelle de l'agent assistant.

5.3.1 Les composants de base de l'ontologie du domaine

Récemment, Eriksson (2003) a publié une étude listant les éléments pour la conception d'une ontologie dans les systèmes de dialogue, traçant un parallèle entre ces derniers et les systèmes d'extraction de connaissances. À partir de cette étude et de notre expérience empirique,

nous présentons une liste de principes de base à respecter pour élaborer une ontologie qui sera utilisée dans la gestion du dialogue entre l'utilisateur et l'agent assistant (Paraiso et Barthès, 2005). Comme nous le montrons tout au long de ce texte, l'ontologie est utile dans plusieurs étapes et endroits différents. Les principes ici définis ont été conçus essentiellement pour les étapes de l'analyse syntaxique et sémantique du traitement des énoncés.

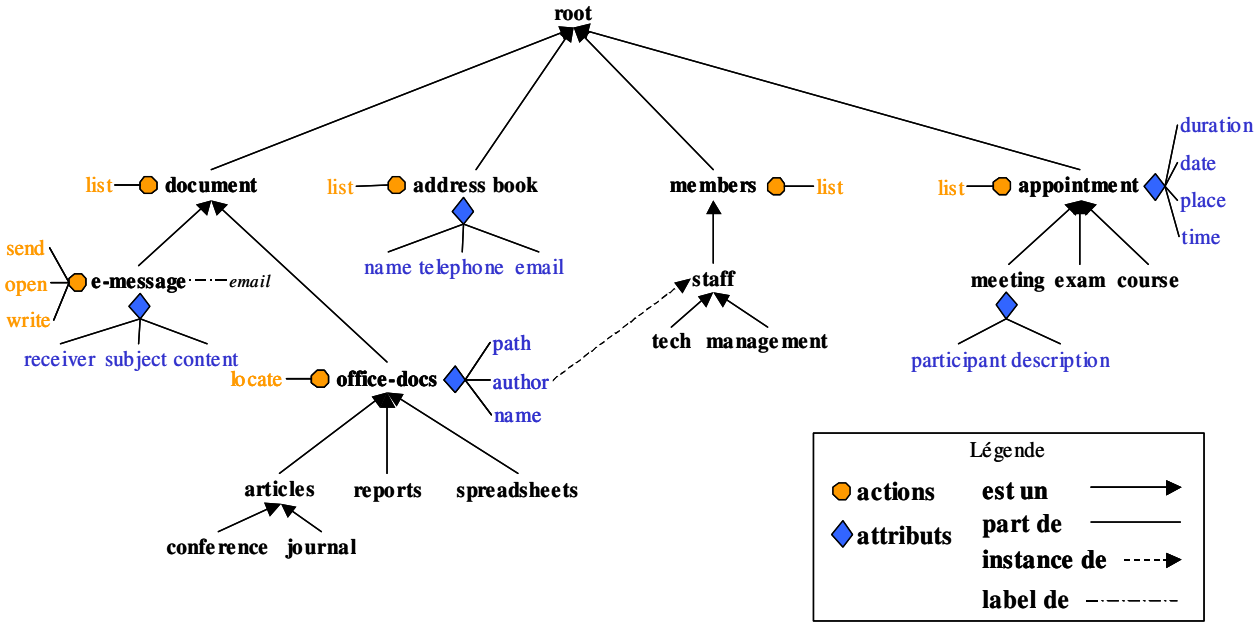


Figure 47 : Un extrait de l'ontologie du domaine.

La Figure 47 montre un extrait d'ontologie élaborée en suivant les principes décrits ci-après. Pour faciliter la lecture de cette figure nous avons supprimé quelques concepts, relations, attributs (propriétés) et étiquettes.

Définition des concepts et de leurs relations

Les concepts et leurs relations (binaires) sont les éléments de base qui constituent une ontologie. Nous indiquons l'utilisation des relations d'hyponymie/hyperonymie (*is-a*) et de méronymie (*has-a*), qui aideront l'interprétation d'énoncés qui ne sont pas dans le champ des réponses prévues, mais qui peuvent être déduites. Prenons l'exemple suivant :

USR (1): List all articles on agents.

Du point de vue de l'interprétation de l'énoncé il serait plus facile pour l'analyseur de trouver le terme *conference article* ou *journal article*. Par contre la relation d'hyperonymie entre *articles* et *conference/journal* permettra à l'analyseur de formuler une requête formelle et arriver au même résultat. De la même façon, la résolution d'une référence sera facilitée, comme dans l'extrait de dialogue suivant :

AP (1): Which documents?

USR (2): Articles.

AP (3): Do you mean conference articles or journal articles?

L'analyseur correctement identifie *articles* comme hyponyme de *documents* grâce à la relation *est-un*.

Pour chaque concept il faut prévoir une liste de mots synonymes qui aidera l'interprétation syntaxique.

Définition des attributs des concepts

Pour chaque attribut d'un concept nous recommandons la précision de son type, d'une liste de mots synonymes et la restriction de domaine appropriée. Nous recensons quatre types de restrictions de domaine : *time*, *space*, *people* et *general*. La restriction *time* est utilisée pour les attribut temporels comme par exemple les attributs du concept *appointment* : *time*, *duration* ou *hour*. La restriction *space* est utilisée pour les attributs liés à une localisation géographique. La restriction *people* définit les attributs liés aux personnes. Et finalement, la restriction *general* est utilisée pour les autres attributs. La restriction est particulièrement utile lors de l'interprétation des questions (plus de détails dans la 5.4.2).

Les actions applicables à chaque concept

Les actions applicables à chaque concept doivent être explicitées. Elles permettront aux agents de service de dégager parmi les demandes d'exécution de tâches, celles qui les concernent. Les actions sont héritables et doivent contenir une liste de mots synonymes. La Figure 48 montre par exemple le cas de l'action *list*, liée au concept *document*, mais qui est héritée par tous les concepts en dessous (*e-message* ou encore *office-docs*).

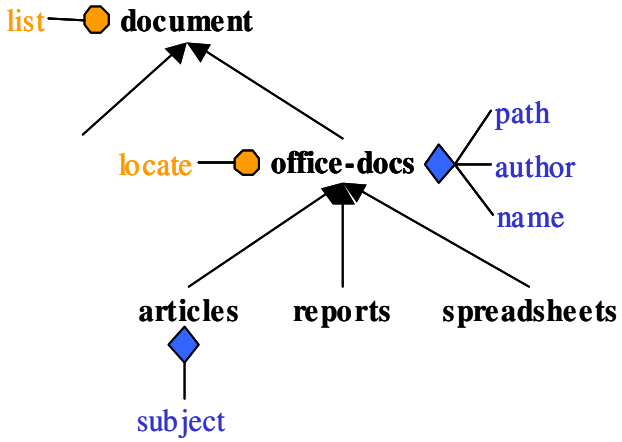


Figure 48 : Un extrait de l'ontologie.

Définition des instances

Concernant les instances, l'instanciation multiple sera notamment utile quand nous avons plusieurs sources d'informations différentes ou dans la définition de plusieurs tâches différentes à partir d'un même modèle.

L'ontologie résultante contient les informations du domaine, mais aussi des informations lexicales et des synonymes. Néanmoins, il faut rappeler que la structure des concepts et de leurs relations est tout à fait définie par le domaine de l'application et non par son utilisation dans le système de gestion des énoncés.

5.3.2 La construction de la requête formelle

Le résultat de l'analyse sémantique est une représentation formelle de l'énoncé. Donc, la construction de la requête formelle est l'analyse sémantique. Comme source d'information pour la construire nous avons l'arbre syntaxique, les liens existants entre les mots et les compléments trouvés, et l'ontologie du domaine. La requête formelle aura la forme d'une liste, selon la représentation BNF suivante :

```
<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
```

Une requête formelle est composée forcement d'une action et d'au moins un concept.

Pour comprendre la construction de la requête formelle, prenons quelques exemples. La Figure 47 présente une ontologie que nous utiliserons pour montrer le fonctionnement de l'analyse sémantique.

Supposons l'entrée suivante, émise par l'utilisateur qui veut savoir s'il a une réunion fixée avec *Mike*.

USR (1): Do I have a meeting with Mike in my office?

Le processus d'analyse syntaxique produit l'arbre syntaxique montré sur la Figure 49 et les informations complémentaires représentées sur la Figure 50.

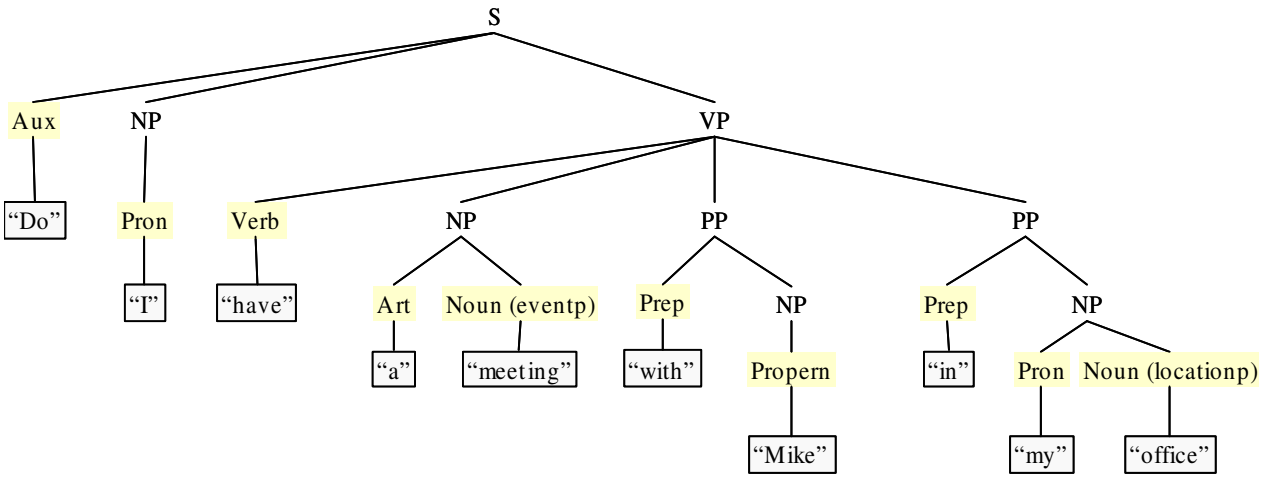


Figure 49 : L'arbre syntaxique.

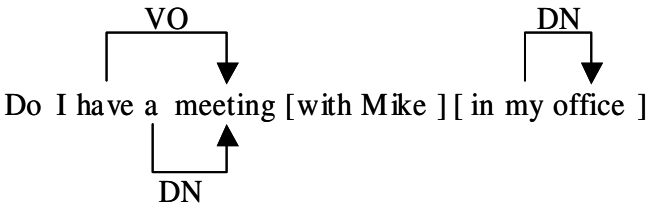


Figure 50 : La structure syntaxique utile pour la construction de la représentation formelle.

Il s'agit bien d'une question conditionnelle qui amène le gestionnaire de dialogue à « chercher » la réponse avant de pouvoir donner suite à la conversation. Pour cela, le gestionnaire construit la représentation suivante, qu'il utilisera pour consulter l'agent de service concerné :

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

Cette liste est créée à partir d'un algorithme qui cherche à trouver d'abord un concept lié au domaine d'application. Le lien VO entre les mots *have* et *meeting* nous indique l'objet de la phrase. Le mécanisme cherche alors un concept équivalent. Il trouve le concept *Meeting* (hyperonyme du concept *Appointment*, voir Figure 51). Il faut savoir que dans l'ontologie chaque concept est lié à une étiquette et à une liste de synonymes utiles dans cette phase de l'interprétation. Les comparaisons entre les mots et les concepts (et ses synonymes) sont faites à l'aide d'un mécanisme de calcul de similarité entre mots : le Trigram.

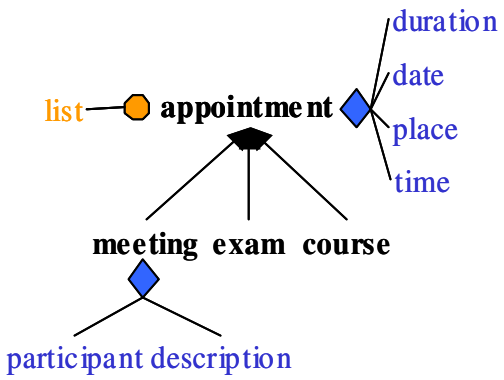


Figure 51 : Le concept *Appointment* et ses sous-concepts.

À partir de la sélection d'un concept, l'algorithme démarre une boucle cherchant à trouver une valeur compatible avec chaque attribut du concept. Le concept *Meeting* possède six attributs : *duration*, *date*, *place*, *time*, *participant* et *description*. Chaque attribut a son propre type explicité dans l'ontologie. Ce processus est facilité par les trois compléments trouvés dans l'énoncé. Pour le premier complément « *in my office* », le lien DN indique la référence au mot *office*, classé par mécanisme d'analyse lexique comme *locationp*, un type spécial pour distinguer les « lieu de travail ». Le slot *place* est donc logiquement concerné. En remplissant les attributs trouvés, la requête s'enrichit, ce qui va aider le processus d'inférence de l'agent. L'action liée à l'objet est, elle, placée au début de la requête, construite sous forme d'une liste.

Pour la paire énoncé-représentation formelle suivante, le processus est le même :

Locate the report on agents

```
(locate (Report (:subject "agents")))
```

Le nom *report* est une indication sur le concept à choisir : *Report*. Une fois le concept choisi, l'analyseur cherche les attributs du concept. Le fragment *on agents* est un complément, dégagé par l'analyser syntaxique, ce qui va faciliter l'identification de l'attribut. Le mot *locate* est un verbe, liée au nom *report*, grâce un lien VO identifié par l'analyser syntaxique. La requête formelle peut alors être construite.

L'énoncé suivant est d'une nature particulière : l'utilisateur veut connaître la valeur d'un attribut d'un concept, en posant la question conditionnelle :

Do you have the address of Mike?

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

Dans ce cas, l'analyseur sémantique va d'abord rechercher le concept compatible avec l'attribut et toutes les autres informations trouvées dans la phrase. La requête maintenant démarre toujours avec l'action *list* suivie de l'attribut désiré et des informations du concept.

Le contexte passé de la conversation peut jouer un rôle dans la construction de la requête. Si par exemple l'utilisateur fait une référence à *Mary* dans un énoncé et juste après il la cite indirectement (par une anaphore) la requête bien évidemment sera complétée. Analysons la conversation suivante :

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

La référence à *Mary* dans (3) sera traitée et l'attribut *receiver* du concept *e-message* sera rempli avec *Mary*.

5.4. Quelques exemples d'interprétation

Nous profitons de cette section pour illustrer avec plus de détails le processus d'interprétation sémantique à l'aide des ontologies. Nous avons déjà montré dans la section 5.3.2 comment les ontologies sont utilisées pour la construction de la représentation formelle de l'énoncé. Maintenant, nous approfondissons cette approche, rajoutant le contexte de la conversation.

5.4.1 Sélection d'une tâche

La Figure 52 présente un énoncé (à gauche) classé comme un ordre par l'analyseur syntaxique, qui peut être interprété à l'aide de l'extrait d'ontologie affichée à côté (à droite).

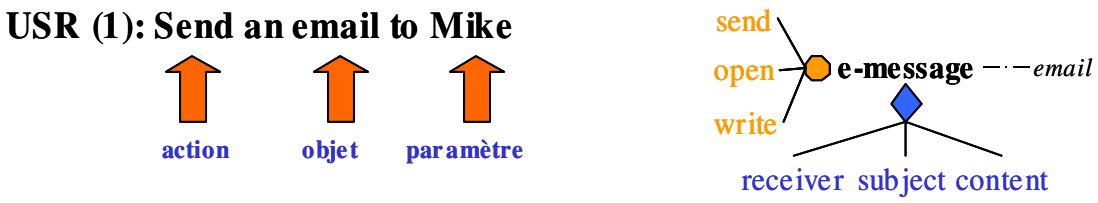


Figure 52 : Exemple de sélection de tâche.

En fonction de l'arbre syntaxique construit par l'analyseur syntaxique, nous pouvons facilement sélectionner trois éléments principaux : un objet (*email,* étiquette liée *à e-message*) lié au concept *e-message*, un paramètre (*receiver*) et une action (*send*). Suivant le processus présenté dans la 5.2, le résultat de l'analyse lexico-syntaxique est montré sur la Figure 53.

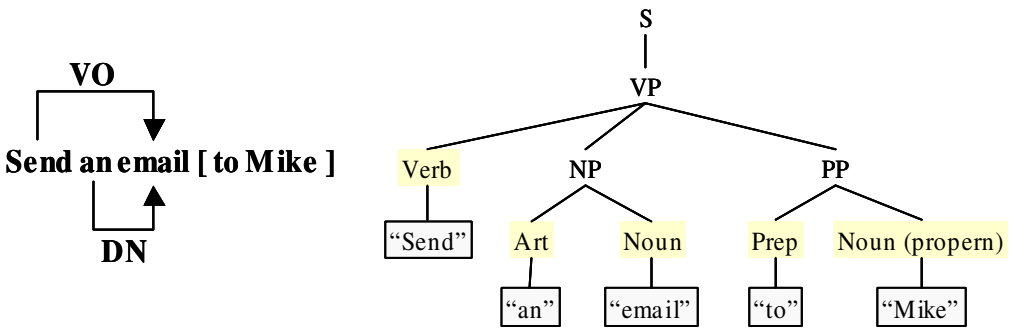


Figure 53 : Le résultat de l'analyse lexico-syntaxique.

L'attribut *receiver* est facilement trouvé dans le complément *to Mike*. À partir de cet énoncé, classé comme un ordre, une représentation formelle est dérivée :

```
(send (E-Message (:receiver "Mike")))
```

Nous rappelons que les tâches sont effectivement exécutées par des agents de service. Ainsi, le processus d'identification de la tâche correspondant à cette représentation formelle, peut se faire selon deux modèles différents. Un premier modèle, que nous appelons *Assistant facilitateur*, est caractérisé par la centralisation, où l'agent assistant connaît a priori les tâches accomplies par chaque agent de service. Il ne lui reste qu'à choisir l'agent qui l'exécutera. Le deuxième modèle, que nous appelons de *Centre de services*, est caractérise par une plus forte distribution, où l'agent assistant ne connaît pas les tâches ni les agents de service disponibles. Dans ce cas, l'agent fait une demande en *broadcast* aux agents de service présents. En envoyant la requête formelle comme elle a été créée, l'agent espère que un d'entre eux se manifestera en donnant suite à la demande. Ces modèles seront présentés en détails dans le chapitre des expérimentations. Dans cette section, à titre d'exemple, nous allons adopter le deuxième modèle : à chaque demande de l'utilisateur, il doit chercher un agent de service capable de l'exécuter, en envoyant la requête formelle en *broadcast* à tous les agents. L'agent de service concerné lui retournera la description de la tâche sélectionnée, contenant tout ce qui doit être récolté auprès de l'utilisateur. L'agent de service pourra retourner la réponse à la requête, s'il est capable de la résoudre avec les informations déjà disponibles. Ce processus de négociation et de sélection sera montré dans le chapitre des réalisations.

Parallèlement à construction de la requête formelle, une instance du concept *E-Message* sera créée en mémoire, contenant tous ses attributs :

```
(E-Message (:receiver "Mike")
           (:subject nil)
           (:content nil))
```

Notons que les attributs sont, bien évidemment partiellement remplis (seul l'attribut *receiver* a été trouvé). Une fois que l'agent assistant a reçu la description de la tâche, le gestionnaire de dialogue démarre le processus de remplissage des paramètres de la tâche sélectionnée. Au fur et à mesure des entrées, les paramètres sont remplis jusqu'au moment où la tâche sera prête pour l'exécution. Les attributs du concept concerné sont aussi remplis (nous rappelons que ces informations sont utiles pour garder le contexte de la conversation). L'extrait de dialogue ci-après présent cette négociation :

AP (2): What is the subject?

USR (3): Setting a meeting for tomorrow

```
(E-Message (:receiver "Mike")
           (:subject "Setting a meeting for tomorrow")
           (:content nil))
```

AP (4): What is the content?

USR (3): Dear Mike, Mrs. Mary Smith is working …

```
(E-Message (:receiver "Mike")
           (:subject "Setting a meeting…")
           (:content "Dear Mike, Mrs. Mary Smith is working …"))
```

L'étape suivante est l'exécution, où l'agent assistant envoie à l'agent de service les informations obtenues.

En suivant la même procédure, l'énoncé montré sur la Figure 54 sera traité, produisant la requête formelle et la charge d'une instance de concept, toutes les deux montrées juste après.

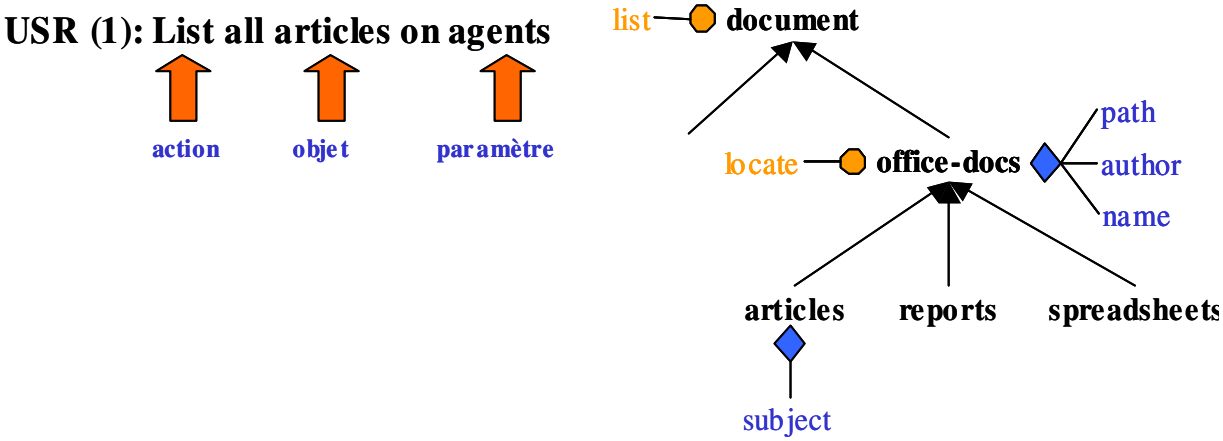


Figure 54 : Un autre exemple de sélection de tâche.

```
(list (Article (:subject "agents")))
```

```
(Article (:path nil)
         (:author nil)
         (:name nil)
         (:title nil)
         (:subject "agents"))
```

À partir de cette instance, le gestionnaire de dialogue trouvera des informations précieuses pour la gestion de la conversation.

5.4.2 Les questions posées par l'utilisateur

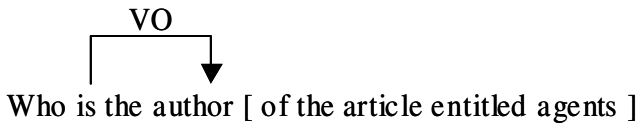
Les questions ont un traitement similaire aux ordres. Après l'analyse syntaxique, il faut d'abord classer la question soit comme directe soit comme conditionnelle. Les questions directes, celles du type *what, where, when, how*, etc., mènent à l'exécution directe d'une action. Les questions conditionnelles, celles du type *can/could, may/might, do/does, is/are*, etc., amènent le gestionnaire à interpréter le résultat avant de répondre à la requête de l'utilisateur.

Directe

La construction de la requête formelle d'une question directe est particulière, puisque le gestionnaire ne cherche pas dans l'énoncé une référence à l'action à exécuter, mais plutôt, l'indication de l'objet désiré par l'utilisateur. Analysons la question suivante :

Who is the author of the article entitled agents?

Cette question est composée d'un complément, contenant l'indication du concept (*Article*) et d'un attribut (*name*) et d'un lien entre le verbe *be* et l'objet *author*, comme illustré ci-dessous :



Dans ce genre de situation, l'action *list* est systématiquement ajoutée, générant la requête formelle :

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

Le gestionnaire après l'envoi et la réception des résultats, pourra les afficher à l'utilisateur.

Pour la question ci-dessous, l'interprétation est plus complexe :

Where is the meeting with Paul?

L'information cherchée par l'utilisateur est indirectement référencée. En utilisant la question *where*, l'utilisateur indique qu'il veut savoir où ce déroula la réunion. Dans ce cas, l'analyseur sémantique utilisera les restrictions de chaque attribut du concept trouvé (*Meeting*) pour construire la requête formelle :

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

L'attribut *place* a la restriction *space*, qui indique la localisation de la réunion.

Pour compliquer un peu plus, la question suivante n'indique pas « explicitement » l'information désirée et, en plus, en cherchant un attribut de restriction compatible avec le type de question *when*, l'analyseur sémantique trouvera trois attributs avec la restriction *time* : *date*, *duration* et *time*.

When is the meeting at my office with Paul?

La requête formelle résultante aura les trois attributs, comme données manquantes à être listées par l'agent de service qui l'exécutera.

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

Cette liste est envoyée aux agents de service, en utilisant un protocole propre, décrit en détails dans le chapitre suivant.

Conditionnelle

Le cas présenté sur la Figure 55 est un peut différent, car avant de pouvoir répondre à la question posée, l'agent assistant doit d'abord déterminer s'il a la réponse.

USR (1): Do I have a meeting today?

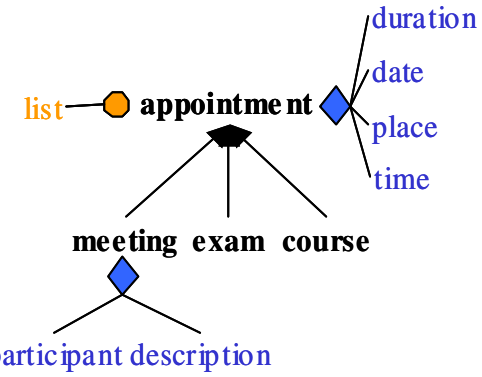


Figure 55 : Le traitement d'une question conditionnelle posée par l'utilisateur.

Pour cela, la requête montrée ci-dessous est créée.

```
(list (Meeting (:date "28-03-2005")
               (:participant "Emerson")))
```

Le gestionnaire répondra *oui*, suivi du résultat reçu de l'agent de service, si ce dernier lui a envoyé une réponse positive (une réunion dans ce cas précis), ou *non* si l'agent de service ne trouve pas de réunion.

5.4.3 Les réponses

Les énoncés classés comme réponses par l'analyseur syntaxique sont traités de façon analogue. Dans l'exemple suivant, à l'aide de l'ontologie du domaine et du modèle de tâches l'identifiant du *receiver* est trouvé dans l'énoncé (2) :

USR (1): Who is the receiver?
AP (2): The receiver is Mike
USR (3): What is the subject?

En trouvant l'information désirée, l'agent donne suite à conversation. En cas de défaut d'information, la même question serait posée.

5.4.4 La référence

Nous traitons la référence localement, vu que toutes les conversations sont faites par l'agent assistant.

La Figure 56 illustre l'occurrence d'une anaphore existante entre les énoncés (1) et (3).

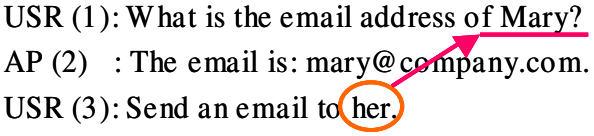


Figure 56 : Un exemple d'anaphore.

Après la question posée par l'utilisateur et l'exécution de la requête ci-après :

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

le gestionnaire pourra résoudre la référence à *Mary* dans (3) (le pronom *her*) grâce à l'instance du concept, créée aussitôt l'énoncé (1) traité :

```
(E-Message (:receiver "Mary")
           (:subject nil)
           (:content nil))
```

Les instances sont stockées dans une mémoire sur forme d'une pile, ce qui facilite l'identification de la référence, puisque forcement l'objet référencé est apparu dans les derniers énoncés.

Le même se passera pour la référence trouvée entre les lignes (1) et (3) de la conversation montrée sur la Figure 57.

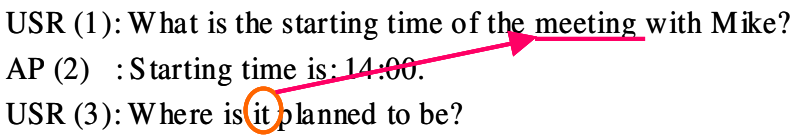


Figure 57 : Un autre exemple de traitement de la référence.

```
(Meeting (:date "17-05-2005")
         (:time "14-00")
         (:place "office")
         (:duration "1-00")
         (:participant "Mike")
         (:description nil))
```

La présence du pronom *it* dans l'énoncé (3) force le gestionnaire à chercher une instance de concept compatible. Dans le sommet de la pile de concepts il trouve l'instance *Meeting*, tout juste créée. À partir de cette instance il pourra formuler une requête pour répondre à la question posée en (3).

5.4.5 Exploration d'un énoncé incomplet

La conversation montrée ci-dessous nous présente un autre avantage de l'utilisation des ontologies dans l'analyse sémantique. À travers l'énoncé (4) l'utilisateur demande à l'agent assistant une action que celui-ci ne peut pas exécuter : effacer un courriel.

AP (1) : You have a new email: sender Mike Palmer.

USR (2) : I would like to read it

AP (3) : *AP empile le message « Email account loaded. » dans la file de message de basse priorité.*

USR (4) : Erase this email

AP (5) : I can't do what you want. However I can: send an electronic message or open the email account. These are tasks related to email.

Selon le bout d'ontologie montré sur la Figure 58, « *erase* » n'est pas une action disponible pour le concept *e-message*. L'agent assistant n'a pas d'autre choix et refuse l'ordre de

l'utilisateur. Par contre, toujours grâce à l'ontologie, le gestionnaire pourra suggérer un certain nombre d'actions applicables au concept dégagé de la conversation : « *e-message* », soit : les tâches liées à l'action *send* ou *open*.

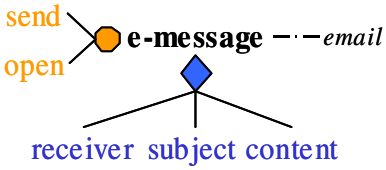


Figure 58 : Le concept *e-message*.

5.5. Le gestionnaire de dialogue

La coordination du dialogue entre l'agent assistant et son maître est une tâche déléguée au gestionnaire de dialogue[7]. Le gestionnaire de dialogue est responsable, entre autres, du remplissage des paramètres des tâches reconnues, de la gestion des instances de concepts en mémoire, ou encore de la présentation des résultats provenant des agents de service. Le gestionnaire a été conçu pour déclencher et gérer plusieurs tâches en mémoire (à travers une pile de tâches). Les tâches sont stockées dans la pile de tâches, au fur et à mesure qu'elles sont identifiées. La stratégie coopérative a été choisie pour permettre l'agent de tenir compte du but de son maître et de toujours faire le maximum pour l'atteindre. Le gestionnaire de dialogue met en place cette stratégie, démarrant une conversation entre l'agent et le maître chaque fois qu'il le faut. Ainsi, à partir d'un énoncé, le gestionnaire donne suite à la conversation par plusieurs chemins possibles. Pour illustrer une partie de ces chemins, prenons le début d'une conversation : après le message de bienvenue, qui marque le début d'une session de conversation, en fonction de l'énoncé donné par l'utilisateur, le gestionnaire peut :

1. charger une nouvelle tâche en mémoire ;
2. démarrer le processus d'aide à l'utilisateur ;
3. refuser l'énoncé ;
4. fermer la session (si l'utilisateur le souhaite).

À partir de la deuxième interaction, s'ajoutent :

1. la gestion d'une réponse à une question posée ;

[7] Certains auteurs, comme Rouillard (2002) utilisent l'expression *dialoguer* pour désigner le gestionnaire de dialogue.

2. le changement d'un paramètre d'une tâche de la pile de tâches ;
3. l'annulation d'une tâche ;
4. l'exécution d'une tâche.

Cela peut encore se compliquer si l'on pense à toutes les interventions possibles de la part de l'agent assistant, comme l'arrivée d'un résultat d'exécution de tâche, d'une question provenant d'un agent de service ou d'un message d'alerte.

Le Tableau 8 résume toutes les tâches attribuées au gestionnaire de dialogue et les facteurs qui déclenchent leur exécution.

Fonctionnalités	**Facteur déclencher**
Identifier et charger une tâche	Enoncé d'utilisateur
Envoyer une tâche à l'exécution	L'état de la tâche est : prête
Suspendre une tâche	Enoncé d'utilisateur
Déclencher l'aide à l'utilisateur	Enoncé d'utilisateur
Changer un paramètre d'une tâche	Enoncé d'utilisateur
Effacer une tâche en mémoire	Enoncé d'utilisateur
Remplir les paramètres d'une tâche	Enoncé d'utilisateur
Gérer plusieurs tâches dans une pile de tâches	En continu
Gérer la pile d'énoncés à traiter	En continu
Gérer les mémoires de conversation	En continu
Refuser un énoncé	Enoncé d'utilisateur
Fermer la session de dialogue	Enoncé d'utilisateur

Tableau 8 : Les fonctionnalités du gestionnaire de dialogue.

Au sein de ce dispositif un groupe de structures (Figure 59) maintiennent les informations de contexte (pile de concepts), des tâches (pile de tâches), les énoncés (pile d'énoncés) et la corbeille d'entrées.

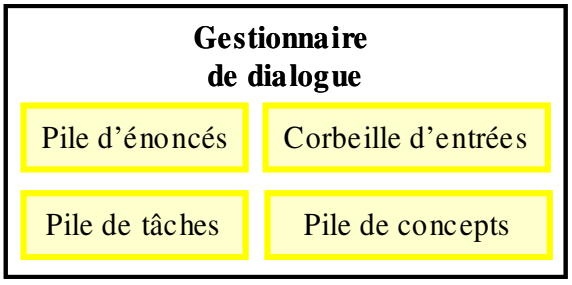


Figure 59 : Les structures de données du gestionnaire de dialogue.

Identifier et charger une tâche

Une tâche est chargée en mémoire, c'est-à-dire stockée au sommet de la pile de tâches, si après les analyses syntaxique et sémantique, la requête formelle résultante a été reconnue par un

agent de service comme liée à une tâche dans le domaine de l'application. L'agent de service retourne à l'agent assistant la description de la tâche. Au cours du chargement, tous les paramètres de la tâche, éventuellement présents dans l'énoncé, sont remplis. Organisée sous forme d'une pile, cette mémoire contient la description et les valeurs des paramètres déjà connus de chaque tâche déclenchée après une requête d'utilisateur.

Les systèmes de dialogue génériques, comme le nôtre, emploient des modèles explicites de tâches pour guider le dialogue avec l'utilisateur. Bien entendu, le but est de permettre que de nouvelles tâches soient ajoutées au système sans que l'on ait à changer le mécanisme du dialogue. Pour cela il est nécessaire d'avoir un modèle de tâche comme l'illustre la Figure 60.

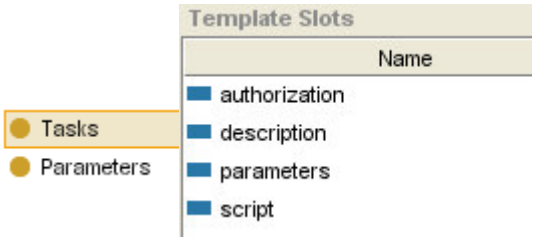


Figure 60 : Le modèle de tâches.

La structure d'une tâche est formée de quatre champs (plus un cinquième créé au moment de l'exécution) :

- *authorization* « autorisation » : il s'agit d'un champ booléen signalant si le moteur de dialogue doit ou non demander à l'utilisateur l'autorisation pour envoyer la tâche à l'exécution ;
- *description* « description » : information textuelle sur la tâche ;
- *parameters* « paramètres » : liste de paramètres de la tâche. Chaque paramètre est décrit selon une liste d'arguments (Figure 61) ;
- *script* « script » : liste d'arguments pour déclancher une fonction d'un des agents de service (celui qui exécutera la tâche). Ce champ est rempli par le moteur du dialogue (le format de ce champ est présenté dans la section 5.4) ;
- *status* « état » : état de la tâche dans la pile des tâches en exécution. Ce champ est créé au moment de l'exécution.

Les arguments d'un paramètre sont montrés sur la Figure 61 et décrits juste après :

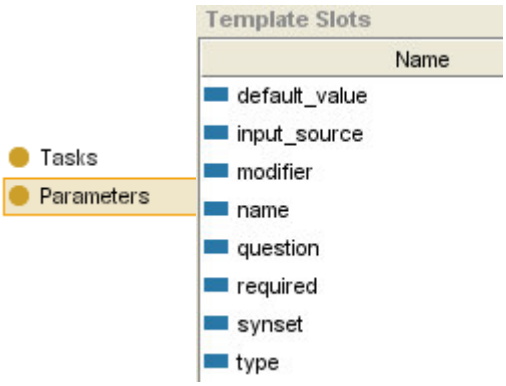


Figure 61 : Les champs d'un paramètre.

- *default_value* « valeur défaut » : contient la valeur défaut du paramètre ;
- *input_source* « source de l'information » : indique si l'entrée doit être saisie par clavier ou par le micro ;
- *modifier* « fonction modificatrice » : une fonction qui doit être exécutée sur une valeur donnée par l'utilisateur. Utile pour adapter une valeur aux exigences de l'exécution, pour exemple : changer « *today* » en 15/07/2005;
- *name* « identificateur » : l'identificateur de la variable qui contiendra la valeur ;
- pstatus « état » : état de saisie de la valeur : *true* indiquant déjà saisie et *false* indiquant ne pas saisir (créé au moment de l'exécution) ;
- *synset* « liste de synonymes » : liste de synonymes utilisés au cours du traitement syntaxique ;
- *question* « question » : question à présenter à l'utilisateur lors de la saisie ;
- *required* « obligatoire » : booléen qui indique si la valeur doit impérativement être remplie ou non par l'utilisateur avant l'exécution de la tâche. Si cette variable de contrôle est *false* et l'utilisateur n'a pas donné une valeur (même sans demande de la part du système), la valeur par défaut sera appliquée ;
- *type* « type » : type de la valeur ;
- *value* « valeur » : la valeur de l'argument (créée au moment de l'exécution).

Gérer plusieurs tâches dans une pile de tâches

Une observation importante : l'interface conversationnelle dispose d'une pile de tâches où sont stockées les tâches en cours d'exécution du point de vue du système de dialogue. L'agent assistant a une autre pile de tâches dans le module Tâches. Cette distinction est importante, car à

partir de l'envoi d'une tâche aux agents de service, le contrôle de la tâche est « perdu » et devient la responsabilité de l'agent de service qui doit retourner le résultat le plus tôt possible. L'agent assistant peut éventuellement rappeler où même relancer une tâche s'il ne reçoit pas de réponse de la part d'un agent de service. Ce contrôle est fait hors interface conversationnelle, en utilisant les informations stockées dans le module de tâches.

Envoyer une tâche à l'exécution

Une fois la tâche prête à être exécutée, elle est stockée dans la mémoire à court terme (module de tâches) de l'agent assistant qui, ensuite, l'envoie à l'agent de service responsable.

Suspendre une tâche

Le concepteur de la tâche peut décider de toujours demander à l'utilisateur une autorisation pour l'exécution de la tâche. Si c'est le cas, au moment où le gestionnaire de dialogue est prêt à envoyer la tâche pour exécution, il va demander l'autorisation de l'utilisateur, qui pourra la refuser, ce qui entraînera la suspension de la tâche. Le gestionnaire consultera à nouveau l'utilisateur, quand toutes les autres tâches de la pile auront été terminées.

Déclencher l'aide à l'utilisateur

Après une demande d'aide de la part de l'utilisateur, le gestionnaire lancera le processus de collecte d'information auprès des agents de service. L'aide à l'utilisateur est fournie à deux niveaux : d'abord une explication textuelle sur le fonctionnement de l'agent assistant (opération de l'agent assistant), ensuite une présentation des informations récoltées auprès de chaque agent de service disponible. Cela permet à l'utilisateur d'avoir une vision générale de l'ensemble du SMA et de ses fonctionnalités.

Changer un paramètre d'une tâche

Après une demande explicite de l'utilisateur, le gestionnaire permettra la modification d'un paramètre déjà rempli.

Effacer une tâche en mémoire

L'effacement d'une tâche en mémoire (le gestionnaire l'efface de la pile de tâches) est réalisée un ordre explicite et confirmation de la part de l'utilisateur.

Remplir les paramètres d'une tâche

Le remplissage des paramètres des tâches est fait à partir des énoncés de l'utilisateur et à l'aide de l'ontologie du domaine.

Gérer la pile des énoncés à traiter

L'analyseur syntaxique peut trouver dans un même énoncé plus d'une tâche à lancer (exemple : « *find the article on agents and open my email account* »). Dans ce cas, la première tâche est lancée et la deuxième empilée.

Gérer la mémoire de conversation

La mémoire de conversation contient les énoncés de la session et les instances des concepts utilisés.

Refuser un énoncé

Un énoncé passé par l'analyseur syntaxique et sémantique, peut être refusé. Plusieurs situations peuvent mener le gestionnaire à refuser un énoncé :

- l'utilisateur demande la suppression d'une tâche, alors que la pile de tâches est vide ;
- l'utilisateur demande le changement d'un paramètre d'une tâche, alors que la pile de tâches est vide ;
- l'énoncé est une réponse à une question posée, mais le type de la réponse n'est pas compatible. Exemple : le gestionnaire attend une réponse du type « oui » ou « non » et la réponse est d'un autre type.

Fermer la session de dialogue

Finalement, la fermeture d'une session est faite à partir d'un ordre explicite de l'utilisateur.

5.5.1 Choix de la forme d'entrée des énoncés

Le concepteur de chaque tâche doit définir la source des entrées : soit le micro (voix) soit le clavier. En fonction du type d'information attendue, il est suggéré l'entrée par clavier : une adresse électronique ou un mot de passe par exemple. L'agent assistant fera le traitement adéquat selon la forme choisie. En cas d'entrée via le clavier, une petite fenêtre est ouverte. Cela peut

être l'alternative lorsqu'une information confidentielle doit être saisie (en conséquence personne ne l'entendra).

5.5.2 Une demande d'aide

L'agent assistant peut répondre à une demande d'aide de la part de l'utilisateur. Cette demande déclenchera un envoi, en *broadcast*, à tous les agents de service afin de collecter leurs propres informations pour les présenter à l'utilisateur.

5.5.3 La corbeille des messages jetés

Une corbeille de messages non désirés est mise à disposition de l'utilisateur. L'agent assistant utilise la corbeille dans le cas suivant :

- les énoncés fruits des bruits de l'environnement : un certain nombre de sources de bruits dans l'environnement où se trouve l'assistant (un téléphone qui sonne ou une discussion entre plusieurs personnes) peuvent générer à l'entrée du micro quelques énoncés parasites. Ils sont facilement détectables par l'assistant. Le parseur classe ce type d'entrée comme une réponse. S'il n'y a pas de question en attente d'une réponse, l'assistant va mettre cette entrée à la corbeille. À tout moment l'utilisateur peut vérifier le contenu de la corbeille.

La corbeille sert à l'agent assistant, mais elle est sous le contrôle du maître, qui peut ainsi vérifier quelles sont les choses jetées par l'agent assistant. Par contre, une fois que quelque chose a été jetée, on ne peut pas la récupérer.

5.6. Politique d'affichage d'informations

Un agent assistant personnel qui interrompt continuellement son utilisateur avec un torrent de questions, deviendra rapidement ennuyeux, et sera, probablement, tout simplement ignoré, perdant complètement sa fonction en tant qu'assistant (Ramos, 2000). L'agent assistant doit avoir des moyens de développer ses activités, de clarifier ses doutes et de résoudre ses problèmes de compréhension, en dérangeant l'utilisateur uniquement quand la situation le justifie, comme par exemple le retour d'information (résultats de tâche) ou dans des situations d'urgence déclarées par l'utilisateur ou par d'autres agents. En fait, il doit avoir une politique de présentation. La politique de présentation est plus qu'un mécanisme d'impression de résultats, elle doit permettre l'agent assistant de traiter les informations qu'arrivent d'utilisateur et des autres agents, en prenant compte leurs contexte et l'état actuel d'interaction avec l'utilisateur. Si par exemple

l'agent assistant reçoit un message informant la mise à jour d'un rapport technique concernant les agents, il pourra différer la présentation de cette information pour la présenter dès qu'il s'apercevoir que l'utilisateur travaille dans une activité concernée par les agents.

À l'origine, la définition de politique de présentation présentée par Ramos ne clarifiée pas comment la mettre en place, ce qui est tout à fait compréhensible car la politique de présentation par elle-même est un sujet complet de recherche. Pour donner un exemple, la politique de présentation proposée par Ramos ne prévoyait pas de critères de sélection des messages qui arrivent, laissant cette décision à l'utilisateur qui devrait régulièrement surveiller le contenu des mémoires qui les stockent.

Nous n'avons pas largement étudié la politique de présentation de l'agent assistant. Cependant, nous avons mis en place une politique d'affichage d'informations que nous considérons un premier pas vers la politique de présentation. La politique d'affichage d'informations permettra à l'agent assistant de savoir quand et comment interrompre le maître, pour poser une question ou pour présenter les résultats de l'exécution d'une tâche. Cette politique empêche l'agent assistant d'interrompre continuellement son maître évitant que ce dernier l'ignore. Elle résoudre le problème de traitement de messages qui arrivent cité ci-dessus, en classant les messages (en leur donnant une priorité) et en les stockant dans des mémoires spécifiques. Ainsi, c'est l'agent lui-même qui décidera quand et comment les présenter à l'utilisateur. De plus, Ramos n'avait pas prévu les interrogations provenant des agents de service, situation que nous avons traitée dans la politique d'affichage d'informations.

La politique d'affichage d'informations agira directement sur la façon dont les énoncés sont imprimés sur la fenêtre principale. Nous montrons sur la Figure 62 la fenêtre principale de l'agent assistant. Nous préciserons dans le chapitre des réalisations comment elle a été réalisée. Ce qui nous intéresse ici est de savoir quelles sont les zones d'affichage d'information concernées par cette politique. La fenêtre principale possède trois zones d'affichage : la zone d'interaction (occupant la majorité de la surface exploitable), où les énoncés de l'utilisateur et de l'agent assistant sont affichés, et les zones d'affichage des derniers énoncés de chaque participant du dialogue (à gauche).

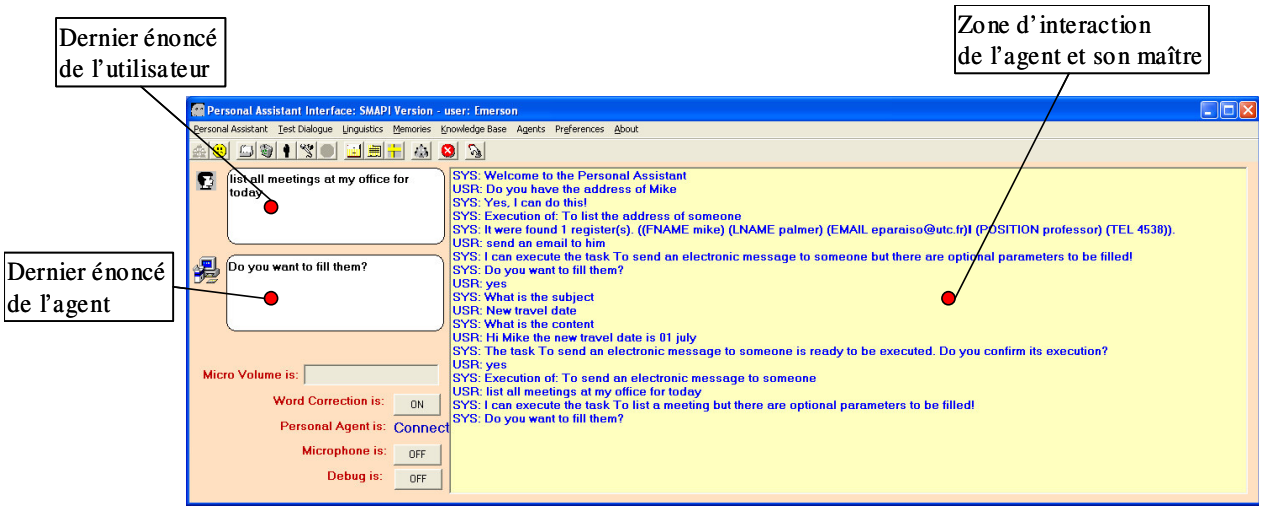


Figure 62 : La fenêtre principale.

Chaque énoncé est imprimé dans les zones correspondantes, une seule fois, selon l'ordre dans lequel il est produit. Les messages qui arrivent sont synthétisés et imprimés dans ces mêmes zones.

Concernant les règles de la politique d'affichage d'informations, nous envisageons deux situations distinctes au quelles l'agent assistant doit faire face : les échanges avec le maître en fonction de ses propres besoins (nous les appelons événements locaux) et les interruptions du maître en fonction de requêtes provenant des agents de service (les événements extérieurs).

5.6.1 Les événements locaux

Les événements locaux sont des interruptions produites par l'agent assistant dans le but d'obtenir des informations complémentaires pour résoudre une tâche qu'il est en train de traiter. Un cas typique est le processus de négociation des paramètres pour l'exécution d'une tâche par un agent de service. L'agent assistant pose des questions à l'utilisateur, l'une après l'autre, afin de remplir tous les paramètres nécessaires à l'exécution effective de la tâche.

D'autres interruptions, moins communes, peuvent être produites à partir des problèmes dus à la connexion réseau ou au manque de mémoire.

Les événements locaux sont des interruptions incontournables, donc seront toujours affichés dans la région de dialogue.

5.6.2 Les événements extérieurs

La seconde classe d'interruptions (dont les structures de contrôle sont montrées sur la Figure 63) concerne tous les messages et requêtes provenant d'autres agents que l'agent

assistant. L'agent assistant va classer ces interruptions en fonction d'un critère de priorité, établi au préalable, à savoir : *basse priorité* et *haute priorité*.

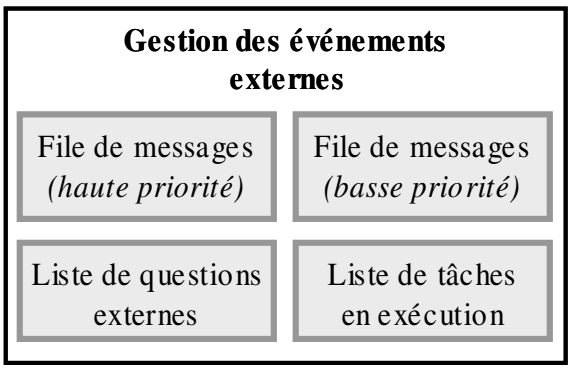


Figure 63 : Les structures de gestion des événements externes.

Basse priorité

Ces sont des interruptions sans intérêts évident comme la confirmation d'exécution d'une tâche (par exemple : « le courriel a été envoyé »), des messages inattendus de format ou d'origine inconnue. L'agent assistant va stocker ces informations dans la région des messages reçus (et non traités) en attendant que le maître vienne les lire. Donc, ce messages ne seront pas affichés (ni lus bien entendu) dans la région de dialogue.

Haute priorité

Ces sont des interruptions importantes comme les résultats d'exécution (réussie ou pas) d'une tâche ou une question qui doit être posée à l'utilisateur, toutes les deux provenant d'un agent de service (y compris des messages d'alerte comme « *You have a new email.* » ou « *You have a meeting today with Mr. Barthès.* »). Dans ce cas, l'agent assistant les affiche dans la région de dialogue lorsqu'il s'aperçoit que c'est possible. Il va attendre que la pile de tâches du système de dialogue soit vide, pour ne pas ennuyer l'utilisateur avec des informations de contextes différents. Ces messages externes resteront donc stockés dans la région de messages reçus et non traités.

Il faut noter aussi que les agents de service peuvent envoyer des questions qui seront posées à l'utilisateur. Elles resteront dans une liste de requêtes et recevront une haute priorité. Dans le cas où il y aurait des requêtes et des messages de haute priorité en attente, les requêtes seront traitées d'abord.

Cette politique d'affichage d'informations est la conséquence de notre hypothèse initiale qui est que l'agent assistant est l'interface privilégiée avec l'utilisateur.

Pour finir, la situation où un agent de service désire interagir directement avec l'utilisateur, en utilisant sa propre interface n'est pas exclue et doit être traitée. Elle se fera dans les conditions suivantes :

1. l'agent du staff annoncera à l'utilisateur, par l'intermédiaire de l'agent assistant, le chargement de sa propre interface ;
2. l'agent assistant appliquera à cette demande une priorité basse ;
3. l'utilisateur doit basculer lui-même sur la nouvelle interface[8].

6. Analyse détaillée de quelques conversations

Nous analysons dans cette section, deux exemples de conversations hypothétiques capables d'être menés en utilisant l'agent assistant avec sa nouvelle interface.

Pour le premier exemple, nous reprenons la conversation du Tableau 6 présentée au début de ce chapitre. La conversation se déroule entre trois personnes (John, Mike et Mary) travaillant dans le cadre d'un même projet. Chacune d'elles a son propre agent assistant capable d'accomplir les tâches décrites dans le paragraphe précédent. Le dialogue du Tableau 6 illustre une conversation entre John et son agent assistant. Après le message qui marque le début de la session (1), l'utilisateur demande à sont agent assistant l'ouverture d'un rapport sur les agents. Cette tâche est exécutée par un des agents de service. Une fois le rapport ouvert, John y consacre quelques instants. L'agent de service responsable de la gestion de l'agenda, envoie un message d'alerte pour l'informer de la proximité d'une réunion (4). Comme la pile de tâches en exécution est vide, l'agent assistant imprime immédiatement le message. Ensuite, John décide d'envoyer un courriel à Mike pour lui demander d'apporter le rapport qu'il prépare sur les plates-formes d'agents (5-9). En interprétant le contenu du message, l'agent assistant constate que le sujet du message, les plates-formes d'agents, a un intérêt pour un autre participant du groupe de projet : Mary. Sans que Mike en soit averti, son agent assistant a notifie l'agent assistant de Mary qui à son tour imprime un message d'information[9]. Mary rédige alors un courriel à Mike pour lui proposer une présentation de ses activités. Une fois le message reçu, l'agent assistant de John

[8] Dans nos implémentations actuelles, les agents de service tournent toujours dans la même machine que l'agent assistant.

[9] Nous rappelons que l'agent assistant peut avoir ses propres tâches, stockées dans le module de tâches, et qui tournent indépendamment du contrôle de l'interface de l'agent.

l'informe (10) tout de suite. John pose une question à l'agent assistant (11) qui lui répond affirmativement. On note la stratégie coopérative du dialogue, qui a mène l'agent assistant à répondre à la question posée et tout de suite après, à ouvrir la boite à lettre de John. À travers l'énoncé (14) l'utilisateur cherche à savoir s'il y aura des réunions le lendemain matin. Cette requête, après l'interprétation, est envoyée à l'agent de service approprié. La réponse arrive et l'agent assistant peut l'imprimer immédiatement puisque la pile de tâches du système de dialogue est vide. À ce point du dialogue, John cherche à savoir s'il peut proposer une réunion entre Mary, Mike et lui même. John demande à son agent assistant s'il connaît l'adresse électronique de Mary. En recevant une réponse positive, John écrit un courriel à Mike et à Mary. On note que Mike à indirectement référencé Mary par le pronom *her* (anaphore). Au cours de son travail de saisie du courriel (18-22), un message d'alerte arrive (23) annonçant un nouveau courriel. Le message restera dans la file d'attente jusqu'au moment où la pile de tâches sera vide. C'est ce qui va se passer après l'envoi du courriel que John est en train d'éditer.

Voyons maintenant un deuxième cas, illustré par l'extrait de dialogue ci-dessous.

AP (1):	Welcome to the Voice Personal Assistant.
USR (2):	Could you book a ticket to Sao Paulo?
AP (3):	Your request is out of my domain.
USR (4):	What can you do?
AP (5):	*AP imprime un message explicatif et tous les messages explicatifs provenant des agents de service.*
USR (6):	Find all documents on relational databases.
AP (7):	There are no documents on this subject.
AP (8):	You have a new email: sender Mary Palmer.
USR (9):	I would like to read it.
AP (10):	*AP rajout le message « Email account loaded. » dans la file de message de basse priorité.*
USR (11):	Erase this email.
AP (12):	I can't do what you want. However I can: send an electronic message or open the email account. These are tasks related to email.
USR (13):	Can you send an email for me?
AP (14):	Yes.

AP (15):	Who is the receiver?
USR (16):	Bill Johnson.
AP (17):	What is the subject?
USR (18):	Cancel this message.
AP (19):	The task to send an electronic message was canceled.
USR (20):	Send an email to Mike.
AP (21):	What is the subject?
USR (22):	My new email address.
AP (23):	What is the content?
USR (24):	Hi Mike. Here is my new email address.
AP (25):	What is the email address of Mike?
USR (26):	Mike.philips@company.com.

Tableau 9 : Deuxième exemple de dialogue.

Cet extrait de dialogue est le fruit d'une conversation hypothétique entre un utilisateur tout juste arrivé, donc inexpérimenté, et son agent assistant. Après le message d'ouverture, l'utilisateur fait une demande hors du champ de l'application. L'agent assistant n'a pas d'autre alternative que de refuser l'entrée. L'utilisateur à travers l'entrée (4) questionne l'agent assistant pour savoir ce que ce dernier peut faire. Ensuite, l'utilisateur formule une requête concernant la recherche de documents sur les bases de données relationnelles. L'exécution de la tâche est réussie mais il n'y avait aucun document sur ce sujet. Dans l'énoncé (8) l'agent assistant annonce l'arrivée d'un courriel. L'utilisateur enchaîne par l'entrée (9) qui mène l'ouverture de sa boite postale électronique. Après la lecture du message, l'utilisateur demande à l'agent assistant de l'effacer. Selon notre ontologie de travail, montrée sur la Figure 47, effacer un message n'est pas une action possible pour le concept *e-message*. Par contre, l'agent assistant identifie que l'entrée est tout à fait dans son domaine d'action. Il suggère, à travers (12), les actions applicables sur ce concept. On note le rôle de l'ontologie qui contient les actions applicables à chaque concept. Ensuite, l'utilisateur démarre l'écriture d'un courriel pour *Bill Johnson*. Avant de finir, il change d'avis et annule la tâche (18), pour démarrer une nouvelle tâche d'écriture d'un message, cette fois pour *Mike*. Après tous les paramètres aient été remplis, l'agent assistant

envoie la tâche à l'agent de service responsable. Ce dernier demande l'adresse électronique de *Mike* à l'agent *AddressBook* qui ne l'a pas[10]. L'agent *Courrier* envoi une requête à l'agent assistant en demandant l'adresse de Mike, qui la présente à son maître. Cette demande externe arrive à l'agent assistant qui la rajoute à la file de messages de haute priorité. Comme la pile de tâches est vide, il peut l'afficher tout de suite.

7. Contribution à l'évolution du modèle de l'agent assistant

Avant de passer au chapitre des expérimentations, arrêtons nous un instant pour apporter des nouveaux éléments à la détermination du modèle générique de l'agent assistant. Dans la section 2.2 du Chapitre 2, nous avons montré l'évolution du modèle d'agent assistant depuis sa première proposition par Ramos et les améliorations qui ont été apportées par Enembreck. Rappelons que ces améliorations ont fait évoluer l'agent substantiellement, surtout pour en diminuer sa complexité, en fonction de la mise en place d'un staff d'agents qui tournent sur la machine de l'utilisateur et qui lui rendrent des services personnalisés.

Cependant, il y a quelques points qui nous paraissent encore « instables » dans l'architecture utilisée jusqu'ici. Ces points sensibles ont été dégagés grâce à nos réflexions au cours de la conception et l'implémentation de l'interface conversationnelle. Nous essayerons bien entendu, d'apporter nos contributions pour les résoudre et faire évoluer le modèle de l'agent assistant.

7.1. L'emplacement du système de dialogue et la politique d'affichage d'informations

Le modèle évolué d'agent assistant proposé par Enembreck est capable de traiter des expressions en langage naturel plus complexes que celles traitées par Ramos et comporte un système de dialogue du type orienté tâches. Le moteur de dialogue a été placé dans le module de contrôle de l'agent, mélangent le contrôle de l'agent et le contrôle du dialogue. À notre avis, cette décision compromet la stabilité de l'agent assistant car ce module est le gestionnaire de tous les autres modules de l'agent. Ce module est déjà trop chargé, donc il doit rester libre le plus possible. Nous croyons que la structure que nous avons adoptée dans notre architecture peut

[10] Le processus de négociation d'informations entre les agents artificiels sera présenté dans le chapitre des expérimentations.

résoudre cette incompatibilité, car le système de dialogue (moteur de dialogue compris) est un composant de l'interface conversationnelle (Figure 64). De plus, le module de traitement de la parole est lié au module de traitement linguistique. Ainsi, si nous décidons d'utiliser les entrées au clavier, il faudra seulement enlever ce module et utiliser les composants GUI classiques.

La politique de présentation, n'était pas l'objet d'étude principal d'Enembreck, donc elle était restée à l'état de la proposition initiale de Ramos. Pour bien traiter les interruptions externes, nous avons eu besoin de formuler une politique d'affichage, avec la création des structures propres de gestion des messages (de basse et de haute priorité), ainsi que l'inclusion du traitement de questions (pour être posées à l'utilisateur) provenant des agents de service.

7.2. Le rôle du *Module de tâches*

Dans la proposition d'Enembreck, les tâches du système de dialogue ont été directement codées dans le module de Tâches, ce qui entraîne une confusion « conceptuelle » vu que le module de Tâches a été conçu pour stocker les tâches que l'agent assistant exécute à un moment donné (la capitalisation des connaissances en est un exemple). Lorsque les tâches, du point de vue du système de dialogue, ne sont pas prêtes pour l'exécution, à cause du manque d'informations (paramètres non remplis), elles doivent rester le plus « prés » possible de l'interface et être envoyées au mécanisme responsable du l'exécution seulement lorsqu'elles sont exécutables. À cette fin, nous avons créé une pile de tâches gérée par le gestionnaire de dialogue.

7.3. Le modèle résultant

La Figure 64 présente le modèle de l'agent assistant résultante.

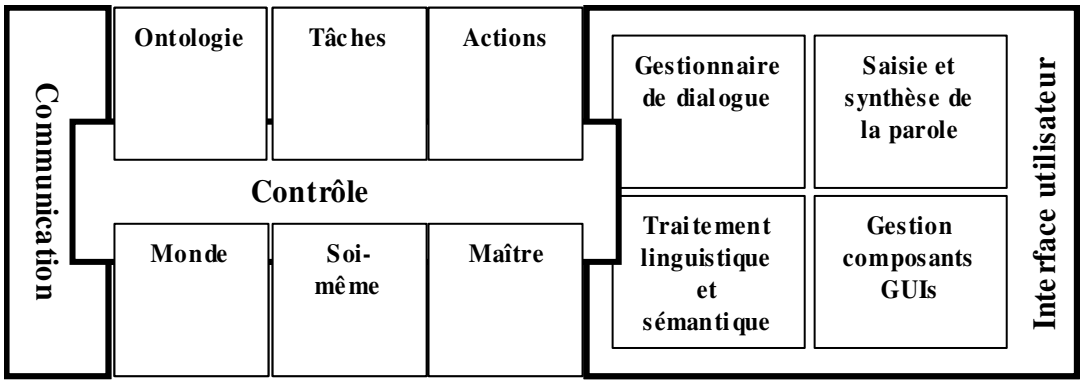


Figure 64 : Le modèle résultant de l'agent assistant.

On note, par rapport à la structure proposée par Enembreck (Figure 65), un couplage plus léger entre l'interface et le reste de l'agent. Les modules de gestion du dialogue sont tous insérés

dans l'interface. Le module de contrôle de l'agent assistant n'a pas de tâches supplémentaires à surveiller.

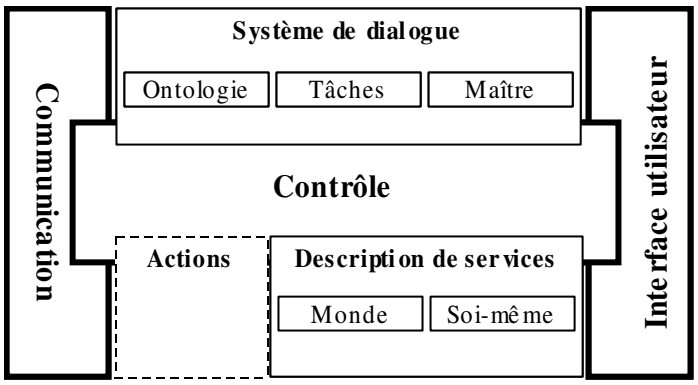


Figure 65 : La structure de l'agent assistant selon Enembreck (2003).

Tous les autres modules (ontologie, tâches, actions, monde, soi-même et maître) du modèle restent les mêmes. L'interface est connectée à l'agent par le module de contrôle.

8. Synthèse et conclusions de ce chapitre

Le but principal de ce chapitre était de présenter la nouvelle interface conversationnelle de l'agent assistant. Nous avons montré que l'agent assistant exige de la part de l'interface un certain nombre de fonctionnalités spécifiques, liées à l'adaptation à l'utilisateur et au domaine d'application. Cela nous a permis de dégager le besoin d'une architecture dédiée aux agents assistants et de proposer le concept ICAI - Interface conversationnelle pour une aide intelligente : le résultat de la conjonction d'un mécanisme conversationnel en langage naturel parlé et de la gestion intelligente de ce mécanisme, permettant le déroulement d'un dialogue coopératif et capable de gérer le déclenchement de plusieurs tâches à la demande de l'utilisateur, avec un minimum d'effort de la part de ce dernier.

Nous avons aussi discuté le fonctionnement d'un ancien système de dialogue disponible dans OMAS-WA basé sur des graphes de conversation. Nous croyons que ce système est plus indiqué pour les applications du type « questions fréquemment posées » ou les systèmes d'apprentissage basés sur ordinateur.

Le restant du chapitre a décrit notre proposition, en portant une attention spéciale à la place des ontologies dans le traitement syntaxique et sémantique. Nous avons aussi montré la politique d'affichage d'informations.

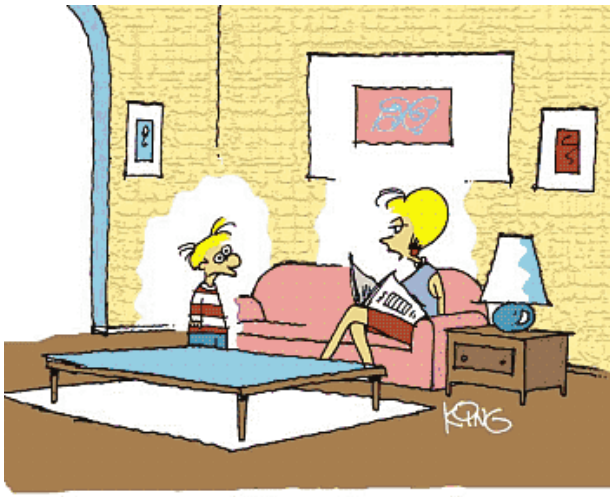
Dans ce chapitre, nous avons présenté nos principales contributions de recherche :

- la conception de l'interface conversationnelle, basée sur :
    - la définition d'un système de dialogue fondé sur les actes de langage directifs (ordre, question et réponse) ;
    - l'adoption d'une stratégie coopérative pour le système de dialogue ;
    - l'ancrage sémantique à travers les ontologies ;
    - la séparation physique des connaissances de domaine et de tâches.
- la mise en place d'une politique d'affichage d'informations ;
- une liste de principes de base à respecter pour élaborer une ontologie pour la gestion de dialogues avec un agent assistant.

À la fin du chapitre, nous présentons nos contributions à l'évolution du modèle générique de l'agent assistant.

Le chapitre suivant est consacré aux réalisations et aux expérimentations.

CHAPITRE 5



"No, you weren't downloaded.
Your were born."

Réalisations et expérimentations

Ce chapitre est consacré au côté expérimental de la recherche. Notre but est essentiellement de construire un agent assistant personnel avec l'interface conversationnelle fondée sur les caractéristiques et modèles proposés dans les chapitres précédents. Nous avons conçu et réalisé deux prototypes de SMA pour valider l'agent assistant avec sa nouvelle interface.

1. La nature des réalisations

Nous avons montré dans le chapitre précédent (section 3) les principaux défis et difficultés lors de la conception et de la réalisation de l'interface conversationnelle de l'agent assistant. Pour surmonter ces difficultés, la conception des composants de l'interface était toujours suivie de sa réalisation, permettant de vérifier sa faisabilité. Ces réalisations intermédiaires sont moins importantes et ne seront pas décrites, car une description détaillée de l'ensemble de l'interface et de l'agent assistant est ici présentée. Les réalisations menées au cours du travail de thèse ont suivi l'idée développée depuis les travaux de Ramos, d'après laquelle l'utilisateur doit interagir avec un seul agent assistant selon le principe du « majordome » qui coordonne les fonctionnalités du système. Dans cette optique, nous avons porté une attention spéciale à la réalisation de l'interface conversationnelle de l'agent assistant, implémentant seulement les aspects minimaux de chaque prototype, nécessaires et suffisants pour valider notre approche. Nous allons débuter par la réalisation de l'agent assistant et son interface, pour ensuite présenter deux prototypes de systèmes multi-agents réalisés pour valider l'interface conversationnelle.

2. L'agent assistant

Le diagramme de classes de la Figure 66 illustre le positionnement de l'interface et de ses composants par rapport à l'agent assistant. Plusieurs modules de l'agent assistant n'ont pas été volontairement représentés.

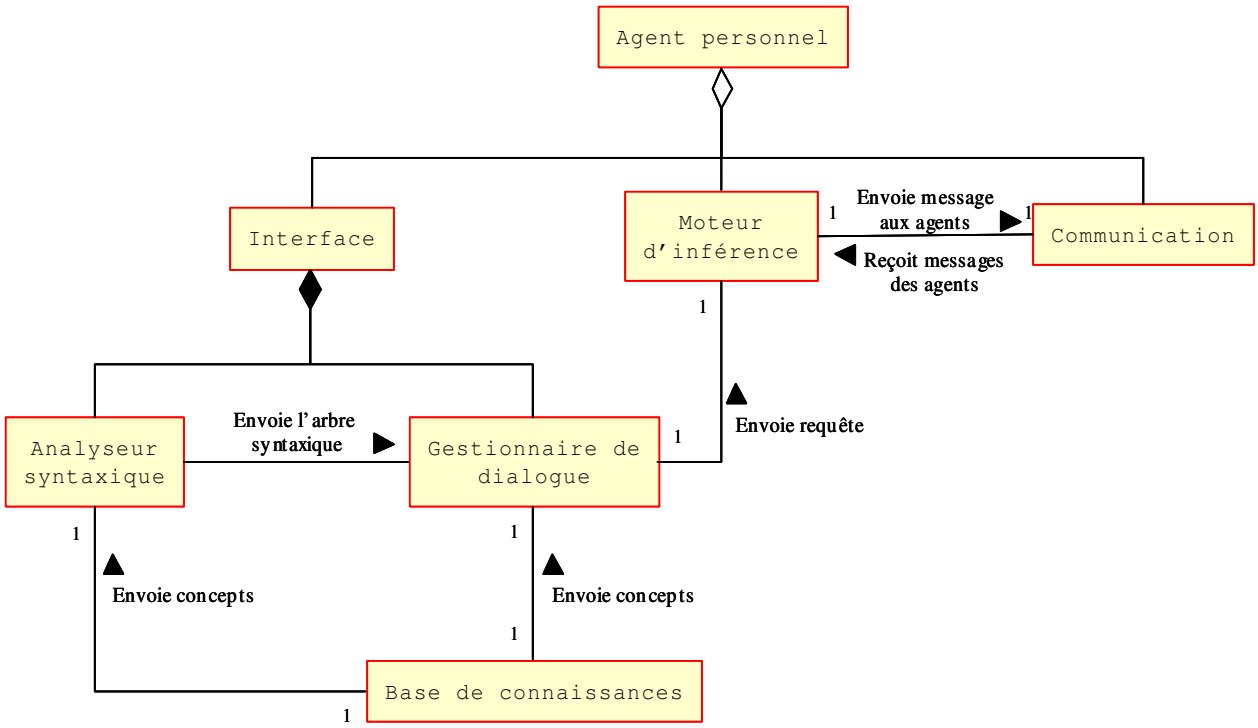


Figure 66 : Diagramme de classes simplifié de l'agent assistant.

Conceptuellement, l'agent assistant est composé de son interface, d'un moteur d'inférence et d'un module de communication. En fait, la réalisation de chacune des ces 3 parties mène à la spécification d'autres modules, formant le modèle d'agent présenté dans le Chapitre 2 (Figure 10). Comme l'interface de l'agent nous intéresse plus particulièrement, nous lui donnons plus d'importance.

L'interface conversationnelle réalisée est montrée dans le diagramme de classes de la Figure 67. Les classes sont représentées sans ses membres.

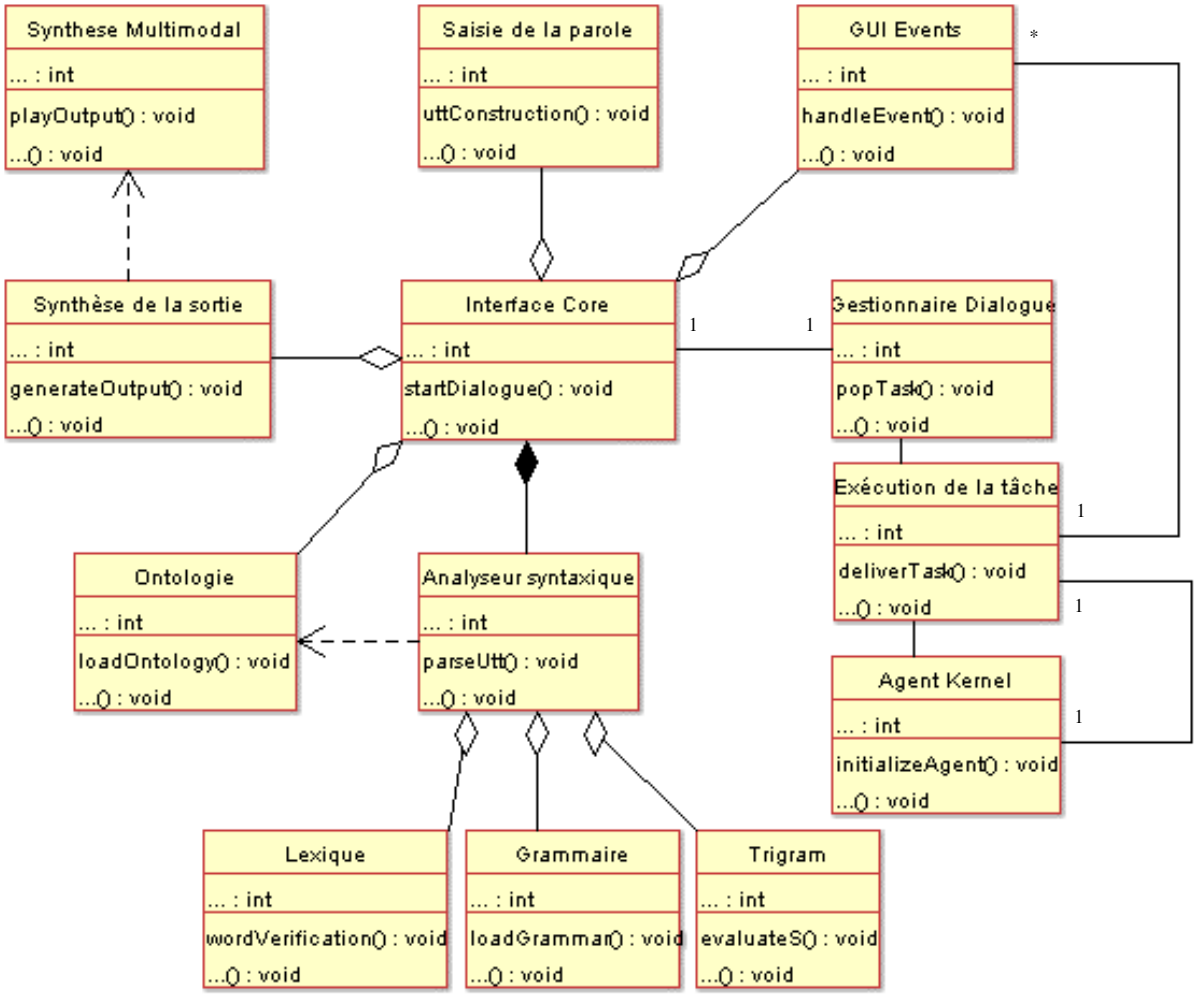


Figure 67 : Diagramme de classes de l'interface conversationnelle.

À partir des spécifications présentées ci-dessus, nous avons réalisé l'agent assistant, dont la fenêtre principale est montrée sur la Figure 68. Elle possède trois zones d'affichage : la zone d'interaction (occupant la majorité de la surface exploitable) où les énoncés de l'utilisateur et de l'agent assistant sont affichés, et les zones d'affichage des derniers énoncés de chaque participant du dialogue (à gauche). Chaque énoncé est imprimé dans les zones correspondantes, une seule fois, selon l'ordre dans lequel il est produit.

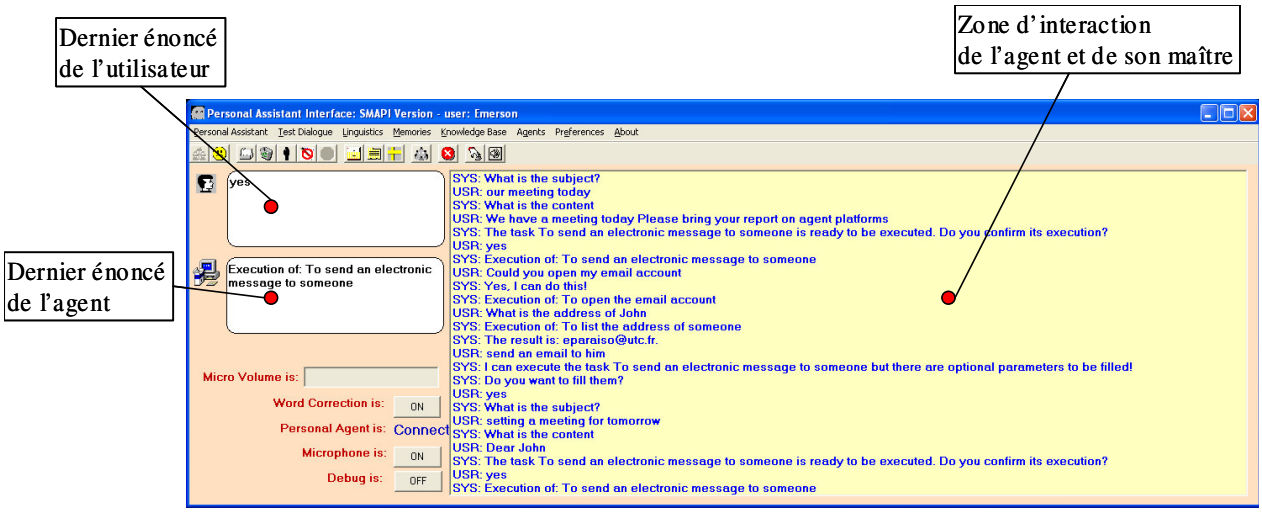


Figure 68 : La fenêtre principale de l'agent assistant.

Une fois l'agent assistant chargé, l'utilisateur peut démarrer une session de dialogue en appuyant sur le deuxième bouton (de la gauche vers la droite) de la barre d'outils. La fenêtre principale comporte aussi un bouton pour activer/arrêter le micro et un contrôleur de volume.

2.1. Outils de développement, réseaux et stockage de données

Tous les agents réalisés au cours de nos expérimentations ont été codés en Allegro-Common Lisp sur la plate-forme OMAS (voir section 3). Toutefois, en raison de l'utilisation des composants ActiveX de gestion des moteurs de reconnaissance de la parole, nous avons été obligé de choisir un deuxième outil de développement pour l'agent assistant, vu que notre version Allegro-Common Lisp ne manipule pas ces genre de composants. Nous avons choisi l'outil Visual Studio, qui nous a facilité d'ailleurs l'intégration des codes sources d'autres outils comme le WordNet et le Link Parser (disponibles en code source en C++). Donc, le noyau de l'agent assistant a été codé en Allegro-Common Lisp (en utilisant la plate-forme OMAS) et l'interface en Visual Studio.

Tout échange de message inter-agents est fait via OMAS, en utilisant le protocole UDP. Les communications sont faites via des *sockets*, où chaque agent à son adresse IP et une porte qui lui est réservée.

La façon dont les données utilisateur sont stockées varie en fonction de l'application (en général de façon distribuée). Chaque agent de service dispose d'un espace de stockage, un dossier où il stocke ses fichiers textuels. L'agent assistant stocke localement les fichiers du lexique, l'ontologie et d'autres fichiers de contrôle (toujours en format textuel).

2.2. Saisie des entrées

Le processus de saisie de chaque énoncé est illustré sur la Figure 69. L'état S0 marque le début de la saisie. Après l'entrée du premier jeton (un mot), une boucle est démarrée (état S1) pour enchaîner tous les mots saisis, formant l'énoncé. Un temporisateur est alors déclenché pour déterminer si la fin de l'énoncé a été atteinte. Ce temporisateur est mit à zéro à chaque fois qu'un mot est détecté à l'entrée du micro. Si la valeur de ce temporisateur dépasse un seuil fixé, une période de silence est reconnue, ce qui amène la transition à l'état S2, où l'énoncé est envoyé au traitement linguistique.

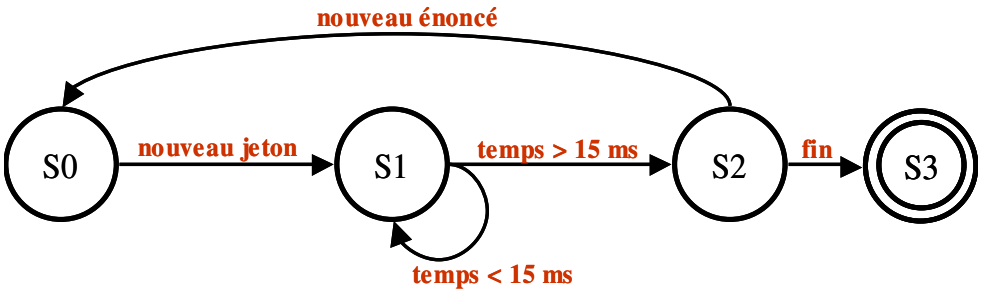


Figure 69 : Le processus de saisie des énoncés.

Nous avons utilisé initialement le moteur Microsoft English Engine 5.1 comme moteur de reconnaissance vocale, puis testé DragonSpeaking de la société Scansoft. Les moteurs sont contrôlés directement par le système d'exploitation. Ils ont plusieurs fonctionnalités : la reconnaissance et la synthèse en sont, bien entendu, les principales. Une API (*Application Programming Interface*) a été développée par Microsoft donnant accès à ces fonctionnalités : la SAPI (*Speech Application Programming Interface*). Tandis que l'usage de ces fonctionnalités est relativement complexe, des contrôles ActiveX/COM ont été développés des différentes sociétés, encapsulant la complexité, et permettant d'interfacer les applications à la SAPI. Ces ensembles de contrôles, comme l'outil Voice Tools de la société Wizzard Software ou le Dragon NaturallySpeaking SDK de la société Scansoft, peuvent être utilisés pour le développement d'applications locales ou pour le WEB, à travers les langages de programmation qui gèrent les contrôles COM (l'Annexe I contient une description plus détaillée).

2.3. L'analyseur syntaxique

Comme nous l'avons montré dans la section de traitement linguistique du chapitre précédent, à la fin du processus d'analyse syntaxique, nous avons un arbre syntaxique et des informations complémentaires pour chaque énoncé. Une grammaire et un lexique appuient le

déroulement de ce processus. Les règles grammaticales (environ 52 règles) ont été divisées en deux groupes : les règles de base et les règles de phrase. Les règles de base groupent les phrases nominales, verbales et prépositionnelles, permettant d'arriver jusqu'aux feuilles. Les règles de phrase groupent les règles syntaxiques qui sont utilisées par le mécanisme d'analyse descendent qui finalement construit l'arbre syntaxique. Même si les règles sont indépendantes du domaine, nous avons eu le besoin de créer quelques types syntaxiques spéciaux, quasi tous dérivés du type nom :

- *datep* : pour les jours de la semaine et les mois de l'année ;
- *timep* : pour l'identification des parties de la journée (matin, après-midi et soir) ;
- *locationp* : pour des lieux géographiques spécifiques comme le bureau, l'université, une salle, et. ;
- *documentp* : pour décrire les types différents de documents : rapport, article, livre, etc.

L'étape suivant la construction de l'arbre syntaxique est l'identification des informations syntaxiques complémentaires. Pour cela, nous avons développé un algorithme basé sur *Link Parser* pour identifier les liens syntaxiques entre les mots qui forment l'énoncé. Cet algorithme codé en C++ sous forme d'une bibliothèque dynamique DLL (*Dynamic Link Library*) fournit les liens entre les mots et les constituants de l'énoncé. Le système *Link Parser*, développé par Sleator et Temperley (1993) réalise l'analyse syntaxique à partir des liens existants entre paires de mots (exemple sur la Figure 70). Chaque lien entre deux mots est typé (adjectif-nom, sujet-verbe, etc.) et répond à des contraintes décrites dans le dictionnaire. Les mots y sont organisés par type de comportement (par exemple verbes transitifs ou non). À chaque famille de mots s'appliquent des contraintes particulières définies dans des règles. Nous pouvons constater sur la Figure 70 les liens créés par l'analyseur. Le lecteur trouvera plus de détails sur ce système dans la référence (Grinberg et al., 1995).

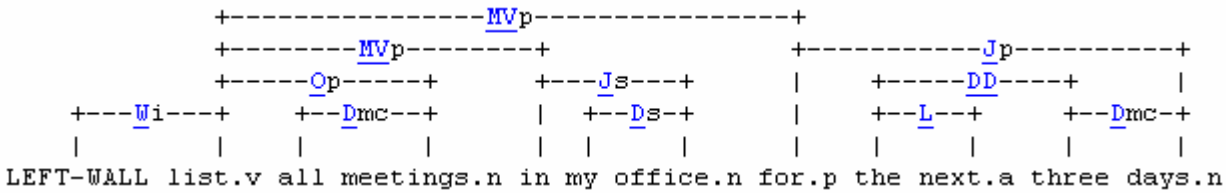
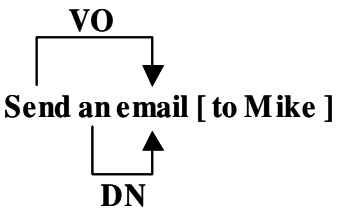


Figure 70 : Le résultat obtenu par *Link Parser* (extrait du site WEB : http://www.link.cs.cmu.edu/link/index.html).

Parmi cette multitude de liens, nous avons défini quatre groupes des liens qui nous intéressaient plus particulièrement.

Verbe-objet (VO)

Le lien verbe-objet lie le verbe à l'objet de la phrase. En général, le verbe indique l'action applicable à l'objet. Par inspection de l'objet, nous cherchons les concepts de l'ontologie du domaine et ses attributs. L'exemple ci-dessous illustre ce lien :



Déterminant-nom (DN)

Le lien déterminant-nom regroupe tous les modificateurs quantitatifs applicables au nom, comme : *many*, *all*, *some*, entre autres.

Adjectif-nom (AN)

Ce lien est créé entre un adjectif et un nom.

Time-nom (TN)

Ce lien exprime les relations de temps entre deux mots.

Un autre support à l'analyseur syntaxique est offert par WordNet (nous avons encapsulé l'outil WordNet[11] dans une bibliothèque dynamique). D'abord, nous avons récupéré le lexique et les ensembles de synonymes des concepts et des attributs de l'ontologie, à partir des fichiers textuels de la base de données WordNet. Le lexique de l'agent est, en réalité, un ensemble de fichiers textuels, où chacun de ces fichiers contient une liste séquentielle en ordre croissant alphabétique des mots de même type syntaxique.

La deuxième utilité de WordNet apparaît au moment de l'exécution de l'agent. Pour faciliter les comparaisons entre deux chaînes de caractères, à chaque fois qu'un verbe est traité, nous obtenons sa forme infinitive à travers l'opération d'*entailment*.

[11] Le code source de l'outil WordNet peut être téléchargé sur http://wordnet.princeton.edu/.

2.4. La gestion de l'ontologie du domaine

Pour gérer la base de connaissances du système, nous avons implémenté une bibliothèque dynamique de gestion de l'ontologie du domaine. La raison de ce choix est purement technique. Au début de nos implémentations, nous avons cherché un mécanisme de gestion de l'ontologie. Nous avons alors commencé par utiliser Protégé (Gennari et al., 2002). Au fur et à mesure de nos recherches, le format et le contenu physique de l'ontologie variaient, rendant de plus en plus pénible son utilisation. De plus, nous voulions un mécanisme aisément intégrable aux agents Allegro-Common Lisp développés sur la plate-forme OMAS. Protégé fournit une API en Java et pour l'appeler à partir d'un programme Allegro-Common Lisp, il faut utiliser le *JLinker*, ce qui ralentissait l'exécution. En conséquence, nous avons développé une bibliothèque en C++. Pour représenter l'ontologie, nous avons choisi le langage XML.

La bibliothèque dynamique est composée de trois classes objets (voir diagramme de classes sur la Figure 71) : la classe *Ontology*, la classe *Concept* et la classe *Attribute*.

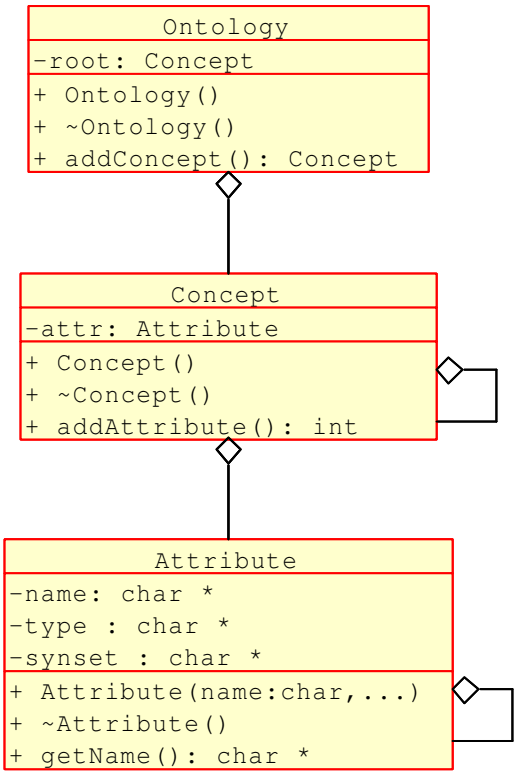


Figure 71 : Diagramme de classes du gestionnaire de l'ontologie.

Une fois une instance de la classe *Ontology* créée, le processus de chargement et d'utilisation de l'ontologie est garanti par une série de méthodes (environ 26), dont quelques exemples sont listés dans le Tableau 10.

Méthode	Description
RetTreeOfConcept	Retourne l'arbre de concepts en dessous d'un concept donné
RetAttributes	Retourne la liste d'attributs attachés à un concept donné
RetActions	Retourne la liste d'actions attachées à un concept donné
RetConceptsOfAttribute	Retourne la liste de concepts qui ont un attribut spécifié
RetConceptsOfActions	Retourne tous les concepts concernés par une action donnée
RetAllSynsetOfConcept	Retourne la liste de synonymes attachée à un concept donné
RetTypeOfAttribute	Retourne le type d'un attribut

Tableau 10 : Les principales méthodes de la classe *Ontology*.

Le gestionnaire de dialogue et les analyseurs syntaxique et sémantique appellent ces méthodes plusieurs fois au cours du traitement d'un énoncé.

Dès que l'ontologie est chargée en mémoire nous avons besoin d'un certain nombre d'algorithmes qui aident sa gestion et son exploration. Nous les présentons dans les sections suivantes.

2.4.1 La comparaison entre chaînes

Les étapes d'analyse syntaxique et sémantique demandent à plusieurs reprises la comparaison entre mots. Lors de la comparaison entre un mot trouvé dans l'énoncé et un concept ou attribut de l'ontologie, une différence lexicale doit être gérée (comme par exemple entre le pluriel et le singulier). Nous avons réalisé un système de calcul de similarité entre mots sous forme d'une bibliothèque dynamique. Cette bibliothèque assure le calcul de la similarité entre deux mots (deux chaînes de caractères) ou entre deux chaînes de mots, utilisant l'algorithme du *N-gram*, où *N* est le nombre de caractères ou de mots consécutifs rencontrés dans les vecteurs concernés par les comparaisons, obtenues comme montré dans l'exemple qui suit.

Dans nos expérimentations, nous avons fixé *N* à 3, ce qui fait que chaque groupe de caractères est connu comme un trigramme (Hylton, 1996). Pour donner un exemple de calcul de similarité entre deux chaînes de caractères, supposons les deux chaînes distinctes :

C1 = « Machine Vision » et C2 = « Machine Learning »

D'abord, il faut générer les trigrammes pour chaque chaîne, combinant les caractères en groupe de trois lettres, comme montré :

TC1 = {mac, ach, chi, hin, ine, ***nev, evi, vis, isi, sio, ion***}

TC2 = {mac, ach, chi, hin, ine, ***nel, ele, lea, ear, arn, rni, nig***}

Après, il faut identifier les trigrammes uniques (une seule occurrence) dans chaque ensemble de trigrammes (formant les ensembles UC1 et UC2). Pour les ensembles TC1 et TC2, tous les trigrammes sont uniques. Ensuite, séparer ceux qui sont uniques et partagés (présents) entre ces deux ensembles (formant l'ensemble aligné dans UP). Le trigramme *mac* est unique et partagé par les deux ensembles (TC1 et TC2). Ce n'est pas le cas du trigramme *nev* trouvé que dans TC1. Tous les trigrammes en TC1 et TC2 imprimés en gras italique sont alors exclus.

UC1 = {mac, ach, chi, hin, ine, nev, evi, vis, isi, sio, ion}

UC2 = {mac, ach, chi, hin, ine, nel, ele, lea, ear, arn, rni, nig}

UP = {mac, ach, chi, hin, ine}

Maintenant, nous sommes prêts à calculer la similarité entre les chaînes C1 et C2, à travers la formule[12] :

$$S = \frac{2 * C}{A + B}$$

Où,
A = nombre de trigrammes uniques en TC1
B = nombre de trigrammes uniques en TC2
C = nombre de trigrammes uniques partagés entre TC1 et TC2
S = (0..1)

$$S = \frac{2 * 5}{11 + 12} = 0{,}43$$

, où :

A = 11 (nombre de trigrammes uniques en UC1) ;

B = 12 (nombre de trigrammes uniques en UC2) ;

C = 5 (nombre de trigrammes en UP)

La valeur de similarité varie entre 0 et 1 (plus proche de 1, plus similaires sont les deux chaînes). Nous avons empiriquement fixé comme seuil minimal la valeur 0,60. Pour ces deux autres exemples, la valeur de similarité est :

[12] Formule de calcul de similarité réalisée selon les notes de cours de Xia Lin sur http://faculty.cis.drexel.edu/~xlin/main.html.

C1 = « Intelligence » C2 = « Intelligences » $S = \frac{2 * 11}{10 + 11} = 0,95$

C1 = « Intelligence » C2 = « Indulgence » $S = \frac{2 * 2}{10 + 8} = 0,33$

2.4.2 La gestion des instances des concepts

La pile d'instances des concepts est un composant clef pour la gestion de la conversation. Elle est alimentée par le gestionnaire de dialogue lors de l'interprétation sémantique. Les énoncés contiennent les références aux concepts de l'ontologie du domaine. Une fois le concept identifié, une instance est créée et aussitôt empilée. Les instances peuvent être visualisées au moment de l'exécution à travers la fenêtre montrée sur la Figure 77. Chaque instance est formée par trois champs groupés dans les structures suivantes :

```
Public Type ConceptInstance
    name As String                      // nom du concept
    n_prms As Integer                   // nombre de paramètres
    vprms() As ArgumentConceptInstance // vecteur de paramètres
End Type
Public Type ArgumentConceptInstance
    prm_name As String      // nom du paramètre
    prm_type As String      // type du paramètre
    prm_value As Type       // valeur du paramètre
End Type
```

Les instances sont mises à jour par le gestionnaire de dialogue en fonction des échanges avec l'utilisateur. En général, le processus de remplissage d'une tâche déclenchée par le gestionnaire de dialogue est la principale source d'informations, puisque les instances du sommet de la pile sont liées à la tâche en remplissage. Si par exemple l'utilisateur fait la demande suivante :

USR (1): List all meetings with Mike in my office.

une instance de concept sera créée (à titre d'exemple, nous avons utilisé l'ontologie montrée sur la Figure 47 du chapitre précédent) :

```
(Meeting (:date nil)
         (:time nil)
         (:place "office")
         (:duration nil)
```

```
            (:participant "Mike")
            (:description nil))
```

Supposons que pour l'exécution de cette demande la tâche illustrée ci-dessous (Figure 72) doive être remplie:

```
- <Task>
    <description>To list meetings</description>
  - <prm prm_id="1">
      <name>DATE</name>
      <synset>date of</synset>
      <question>Give the date</question>
      <type>datep</type>
      <modifier>TreatDate</modifier>
      <required>true</required>
      <input_source>vocal</input_source>
      <default_value>none</default_value>
    </prm>
  - <prm prm_id="2">
      <name>PLACE</name>
      <synset>location at place</synset>
      <question>Give the location</question>
      <type>locationp</type>
      <modifier>none</modifier>
      <required>true</required>
      <input_source>vocal</input_source>
      <default_value>none</default_value>
    </prm>
  - <prm prm_id="3">
      <name>TIME</name>
      <synset>time hour</synset>
      <question>Enter the programed time</question>
      <type>timep</type>
      <modifier>none</modifier>
      <required>true</required>
      <input_source>keyboard</input_source>
      <default_value>none</default_value>
    </prm>
    <script>@AGENDA @ALLREGISTERFROMFIELDS (:event "meeting")</script>
    <authorization>false</authorization>
  </Task>
```

Figure 72 : Description d'une tâche.

Après toutes les négociations, l'instance aura les informations suivantes :

```
(Meeting (:date "20-07-2005")
         (:time "14-00")
         (:place "office")
         (:duration nil)
         (:participant "Mike")
         (:description nil))
```

On note que, en suivant la description de la tâche de la Figure 72, les attributs *duration* et *description* ne seront pas remplis, car ils ne sont pas présents dans la description de la tâche.

2.4.3 Résolution de la référence

Comme nous l'avons montré dans le Chapitre 3, le traitement de la référence n'est pas simple. Nous avons réalisé une partie de la référence, celle qui résout l'anaphore inter-phrases, où la référence est résolue par inspection d'un historique d'objets mentionnés précédemment. Dans notre cas, l'historique est bien évidemment la liste des instances des concepts en mémoire et l'anaphore correspond à la présence de pronoms qui peuvent si rapporter à des objets présents dans l'historique, comme l'illustre l'exemple du chapitre précédent que nous reprenons ici :

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

Après la question posée par l'utilisateur et l'exécution de la requête :

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

le gestionnaire pourra résoudre la référence à *Mary* dans (3) (le pronom *her*) grâce à l'instance du concept, créée aussitôt que l'énoncé (1) est traité :

```
(ElectronicMessage (:receiver "Mary")
                   (:subject nil)
                   (:content nil))
```

L'algorithme réalisé traite la référence entre pronoms non réflexifs (*her, him, it*).

Néanmoins, l'énoncé (3) présente un problème pour notre algorithme : le fait que l'utilisateur ait cité deux personnes différentes crée un complément avec deux sous compléments :

Send an email [[to her] and [to Mike]].

La version actuelle de notre analyseur sémantique va créer deux requêtes distinctes :

`(send (E-Message (:receiver "Mary")))` et

`(send (E-Message (:receiver "Mike")))`

Cela se produit car l'algorithme interprète cette situation comme une action applicable à deux objets distincts. Nous avons l'intention de résoudre ce problème dans la prochaine version du système.

2.5. Le gestionnaire de dialogue

Le gestionnaire de dialogue est responsable, entre autres, du remplissage des paramètres des tâches du point de vue du système de dialogue. Nous avons présenté le modèle de tâches que nous avons conçu dans la section 5.5 du chapitre précédent. La Figure 73 montre la tâche d'envoi d'un courriel au format XML, respectant le modèle présenté[13]. Le modèle de la tâche en mémoire est le même. Au fur et à mesure des échanges, les informations sont mises à jour. Une tâche est composée de plusieurs arguments et paramètres. Les paramètres sont des données utiles pour l'exécution de la tâche, et doivent être obtenus par le gestionnaire de dialogue, à partir d'une conversation avec l'utilisateur. Dans l'exemple, les 3 paramètres : *receiver*, *subject* et *content* seront envoyés à l'agent de service concerné dès que les conditions d'envoi sont atteintes : les paramètres sont remplis et l'utilisateur a donné son autorisation (l'autorisation est optionnelle et est définie par le concepteur de la tâche).

[13] On note que les arguments *value* et *pstatus* d'un paramètre de la tâche sont des variables de contrôle, utiles au cours de remplissage de la tâche, et que à ce titre, elles n'existent pas avant le chargement de la tâche en mémoire. Idem pour le champ *status* de la tâche.

```
- <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>
```

Figure 73 : Un exemple de tâche en format XML

Le gestionnaire stocke la tâche dans la pile de tâches en exécution et au cours de son remplissage, un état s'appliquera :

- *executable* « exécutable » : la tâche est prête à être envoyée à l'exécution, néanmoins, il y a des paramètres (non obligatoire, *required* = *false*) non remplis ;
- *blocked* « bloqué » : état initiale de chaque tâche ;
- *pending* « en suspens » : en attente d'une entrée de l'utilisateur (notamment pour le remplissage d'un paramètre) ;
- *ready* « prête » : la tâche est prête à être envoyée à l'exécution ;

- *executed* « exécutée » : la tâche est finie (au moins du point de vue du gestionnaire de dialogue) et peut être retirée de la pile de tâches ;
- *suspended* « suspendu » : la tâche est prête, mais l'utilisateur a refusé son exécution ;
- *waitingAuthorization* « en attente d'autorisation » : la tâche est prête, mais l'utilisateur n'a pas encore autorisé l'exécution ;
- *canceled* « annulée » : la tâche a été annulée par l'utilisateur ;
- *changing* « en modification » : l'utilisateur est en train de modifier un paramètre de la tâche.

Les états se succèdent en fonction des échanges. Lorsque la tâche est prête pour l'exécution (*ready*) le gestionnaire du dialogue remplira le champ *script* du modèle de tâche, concaténant tous les arguments *value* de chaque paramètre de la tâche. Par exemple :

```
(@SENDEMAIL (:receiver "Marc")
            (:subject "Setting a meeting…")
            (:content "Dear Marc, Mrs. Mary Smith is working …"))
```

Dans le cadre de l'exécution de la tâche de la Figure 72, après de négociations, le gestionnaire va créer la liste suivante :

```
(@AGENDA @ALLREGISTERFROMFIELDS (:event "meeting")
                                (:date "20-07-2005")
                                (:place "office")
                                (:time "14-00"))
```

On note que l'argument *participant* n'est pas présent, et pourtant il est connu. La spécification de la tâche ne l'inclut pas. Toutefois, si ultérieurement l'agent de service le demande, l'agent assistant pourra l'envoyer en consultant l'instance du concept en mémoire.

L'envoi d'une tâche n'est pas un rôle du gestionnaire de dialogue. Cela est une tâche de l'agent assistant à travers le module *tâches*.

2.6. Les mémoires de l'agent assistant

L'agent assistant a besoin de conserver des informations. Le contexte des conversations avec l'utilisateur et les autres agents, les tâches en exécution, l'historique d'une conversation, en sont quelques exemples. Ces informations sont utiles à court terme et à long terme. Pour cela,

l'agent assistant s'appuie sur un ensemble de mémoires : les mémoires à court, à moyen et à long terme. Dans le cadre de l'interface conversationnelle, ces mémoires seront surtout utiles au mécanisme de contrôle de dialogue et de l'interaction avec l'utilisateur.

2.6.1 La mémoire à court terme

La mémoire à court terme contient les tâches envoyées pour exécution aux agents de service. Lorsque la tâche est prête à être exécutée, du point de vue du système de dialogue, elle est stockée dans la mémoire à court terme qui l'enverra à l'exécution. Lorsque la tâche est exécutée, cette position de la pile sera libérée et son contenu déplacé dans la mémoire à moyen terme.

2.6.2 La mémoire à moyen terme

Une fois la tâche exécutée, une partie de son contenu est stockée dans la mémoire à moyen terme, située aussi dans le module « soi-même » de l'agent. Nous gardons pour chaque tâche le champ « *script* » qui contient toutes les informations formatées et envoyées à l'agent de service, ainsi que le champ « *description* » utile pour l'affichage d'informations à l'utilisateur (fenêtre de mémoires).

Nous stockons ces informations, dans un but particulier : d'abord, c'est un moyen de garder les informations déjà fournies par l'utilisateur. Ces informations pourront être utilisées par le mécanisme d'apprentissage de l'agent assistant pour prévoir des actions en fonction des futures requêtes de l'utilisateur. Enembreck a rajouté une méthode d'apprentissage automatique, capable de construire des généralisations à partir d'un ensemble d'informations. Il donne l'exemple dans lequel cette méthode peut être utilisée pour découvrir à qui l'agent doit demander un certain service ou encore prédire quel sera le résultat d'un service donné.

2.6.3 La mémoire à long terme

La mémoire à long terme stocke tous les énoncés de l'utilisateur au cours des sessions de dialogue. Notre but est de pouvoir les explorer pour identifier des clusters d'informations manquantes dans l'ontologie du domaine et dans le lexique du système. En appliquant sur cette base de données, des algorithmes du type TF/IDF (Salton, 1989) par exemple, l'agent assistant pourra trouver les mots qui sont fréquents dans les énoncés d'utilisateur, mais que le lexique ne connaît pas. Les énoncés sont physiquement stockés dans un fichier séquentiel.

2.7. Les fenêtres de contrôle

Les contenus des mémoires de l'agent assistant peuvent être examinés par l'utilisateur à travers quelques fenêtres. Il y a les structures de contrôle du système de dialogue et les structures de gestion des messages externes et des tâches en exécution par les agents de service. La Figure 74 présente les structures de contrôle du système de dialogue :

- la pile de tâches en exécution et ses paramètres respectifs;
- la liste des tâches exécutées ;
- le registre des conversations passées (log contenant tous les énoncés).

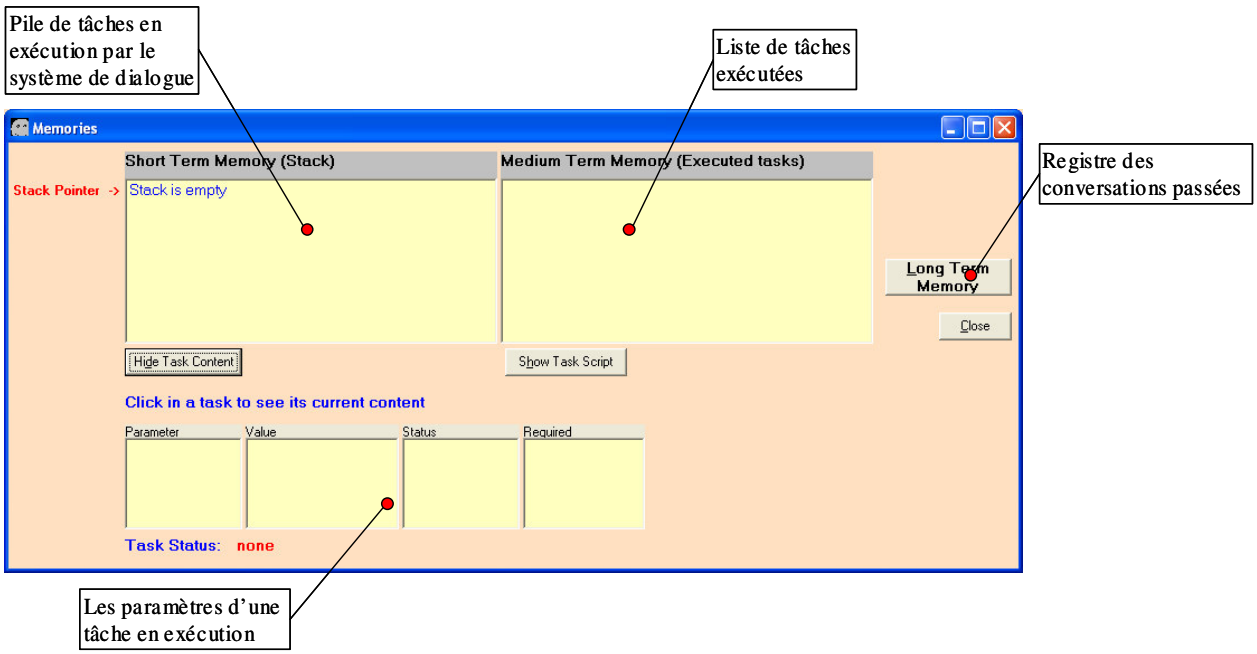


Figure 74 : La fenêtre des mémoires de l'agent assistant.

La fenêtre de la Figure 75 contient les structures de gestion des messages externes et des tâches en exécution par les agents de service. Cette fenêtre n'est pas systématiquement affichée par l'agent assistant. C'est l'utilisateur qui décide s'il le veut ou pas. Elle montre les messages en attente (de haute et de basse priorité), la liste de questions provenant des agents de service qui n'ont pas encore été posées à l'utilisateur et la liste des tâches en cours d'exécution par les agents de service (ne pas confondre avec les tâches en cours d'exécution dans la pile de tâches du système de dialogue).

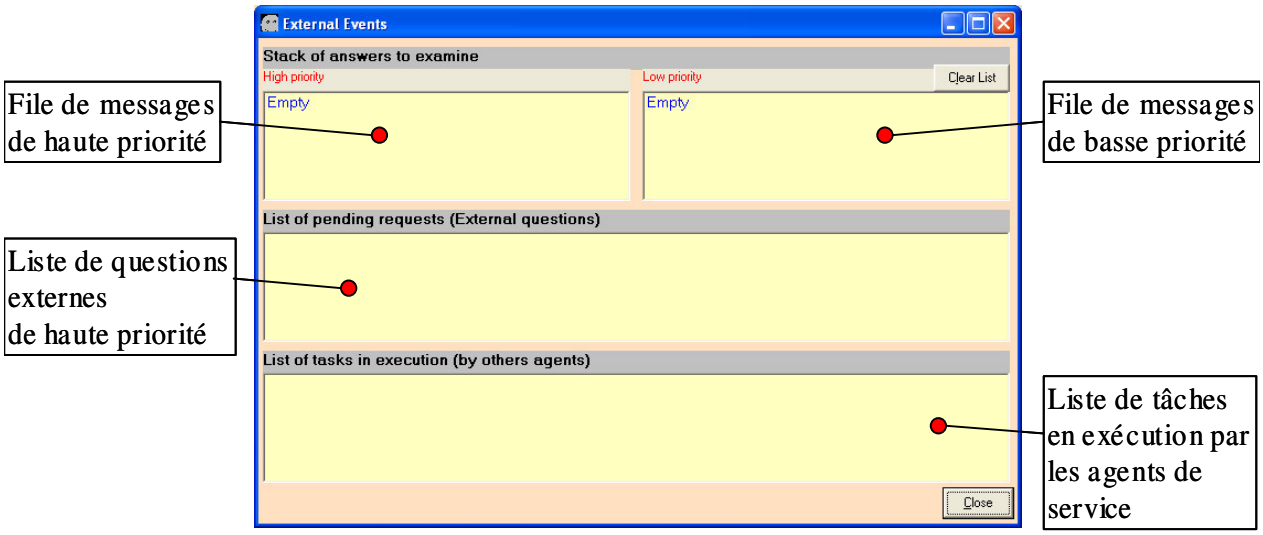


Figure 75 : La fenêtre d'affichage des messages externes.

Au fur et à mesure que les messages externes arrivent, l'agent assistant les stocke, en attendant que le mécanisme de traitement des messages les ramasse, en respectant les principes de la politique d'affichage des informations.

Au cas où l'agent assistant a supprimé un énoncé entré par l'utilisateur (parce qu'il a été jugé inutile), l'utilisateur pourra le visualiser à travers la corbeille d'entrées montrée sur la Figure 76.

Figure 76 : La corbeille d'entrées.

Enfin, la fenêtre de la Figure 77 liste les instances des concepts en mémoire. À partir d'une entrée comme : « USR(1) : When is the meeting in my office with Paul? », le gestionnaire de dialogue maintiendra un concept *Meeting* accessible à tout moment.

Figure 77 : Liste des instances des concepts en mémoire.

Après avoir introduit l'agent assistant que nous avons réalisé, nous nous déplaçons au niveau SMA, en présentant dans la section suivante la plate-forme OMAS.

3. La plate-forme OMAS et l'organisation en coteries

Du point de vue agent, toutes nos réalisations sont faites en utilisant la plate-forme multi-agents développée à l'UTC appelée OMAS. Dans cette plate-forme, les agents sont organisés en groupes relativement compacts appelés *coteries*. L'organisation en *coteries*, inspirée du travail de Nonaka (2000), considère que les agents sont homogènes et que la communication est publique. Chaque agent reçoit tous les messages, lui permettant de mettre à jour et de maintenir sa représentation du monde. Ainsi, des agents sont immergés dans un champ d'information semblable à un « réseau radio ». Une coterie est un endroit où les agents agissent les uns sur les autres et apprennent les uns des autres. Une coterie reproduit les caractéristiques d'un système ouvert puisqu'il n'y a pas de hiérarchie et le nombre d'éléments n'est ni fixe ni déterminé.

La communication dans une coterie est asynchrone en mode *broadcast*, les messages peuvent être envoyés point à point mais le mode de communication est en *broadcast*. Ceci pourrait être interprété comme un gaspillage de bande passante mais, en pratique, les agents

d'une coterie partagent un même réseau local, et chaque agent reçoit tous les messages au niveau le plus bas (Ethernet) (Tacla, 2003). Une communication en *broadcast* ne fait l'objet que d'un seul message utilisant le protocole UDP.

Dans OMAS, la charge d'interpréter le message est placée sur l'agent de réception qui doit utiliser sa propre connaissance et ses ontologies. Ainsi, un agent qui ne comprend pas un message tout simplement l'ignore. Un agent qui comprend partiellement un message essaiera de l'interpréter de son mieux pour agir en conséquence. Normalement, les agents utilisent un langage de contenu, connu par les participants du système, pour leur permettre d'exprimer leurs besoins. Dans le cadre d'un de nos prototypes, nous avons développé un langage de contenu présenté dans l'Annexe II.

OMAS permet la création de deux types différents d'agents : les agents de service (qui fournissent un type spécifique de service correspondant à des compétences qui leur ont été attribuées) et les agents assistants personnels. L'ensemble des agents de services consacrés à un agent assistant forme ce que Tacla (2003) a appelé le *staff de l'agent assistant*. Les agents de service d'un staff ne peuvent pas servir d'autres agents que l'agent assistant. Ils réalisent des tâches dédiées à l'agent assistant auquel ils sont attachés. Un staff d'agents s'exécute en général sur le même ordinateur (celui du maître). Les autres agents de service sont d'ordre plus généreux et peuvent s'exécuter sur différentes machines.

Le concepteur d'un SMA basé sur OMAS a la liberté de choisir comment coordonner les agents qui composent la *coterie*. Nous avons recensé deux modèles de coordination (la Figure 78 les schématisent). Un premier modèle, que nous appelons *Assistant facilitateur*, est caractérisé par la centralisation, où l'agent assistant connaît a priori les tâches accomplies par chaque agent de service. Le deuxième modèle, que nous appelons de *Centre de services*, est caractérisé par une plus forte distribution, où l'agent assistant ne connaît pas les tâches ni les agents de service disponibles. Les systèmes mult-agents développés auparavant dans notre laboratoire ont privilégié le premier modèle ((Enembreck, 2003) et (Tacla, 2003)).

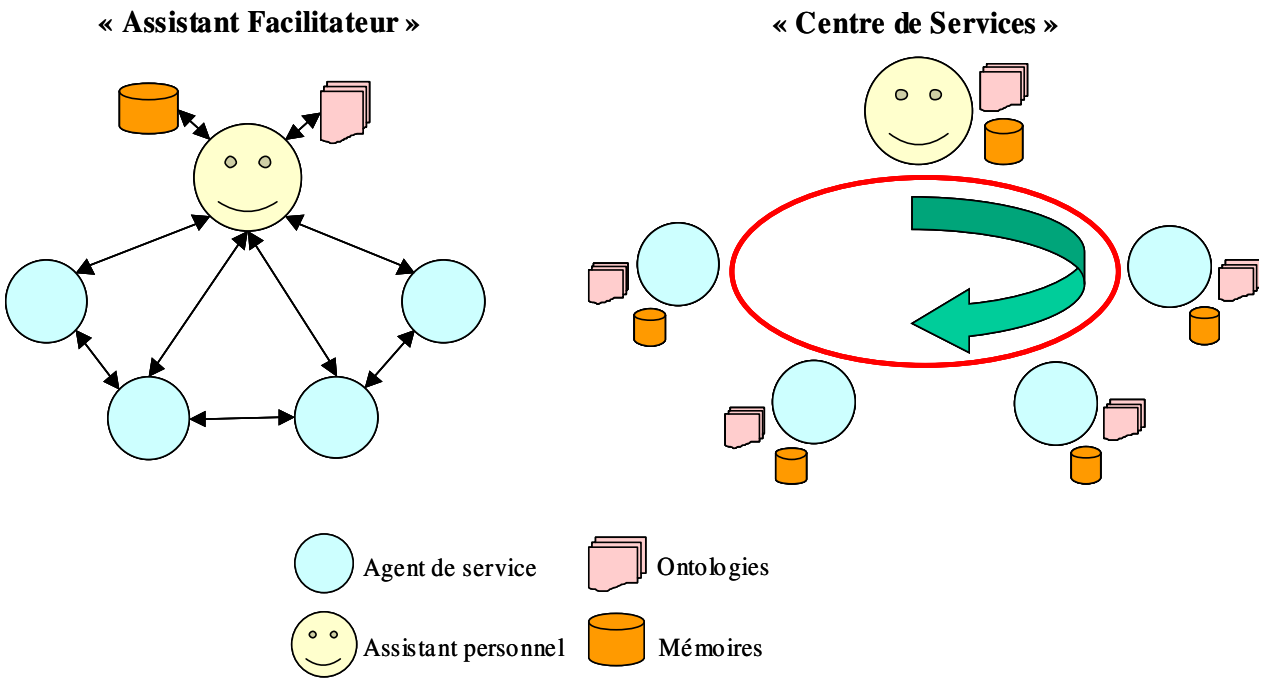


Figure 78 : Les deux architectures implémentées.

Le modèle plus centralisé (à gauche sur la Figure 78) est beaucoup plus simple à réaliser puisque l'agent assistant centralise les ontologies du domaine et de tâches. L'agent assistant possède toutes les ressources nécessaires pour interpréter les besoins de l'utilisateur et pour déléguer l'exécution des tâches aux agents de service. Il négocie les informations impératives pour l'exécution de la tâche avant de les envoyer à l'agent de service qui l'exécutera. Les mémoires sont toutes localisées dans l'agent assistant. Du point de vue de la politique de présentation, c'est un modèle plus souple, car les interruptions seront moins nombreuses. En contrepartie, l'ajout de nouveaux agents (par conséquent de nouveaux services) est plus coûteux, car il doit être annoncé à l'agent assistant et ses ontologies doivent être mises à jour.

Le deuxième modèle (à droite sur la Figure 78) fait de l'agent assistant un « portail de services ». Services d'ailleurs qu'il ne connaît même pas. L'agent assistant reste la seule interface avec l'utilisateur, responsable donc de la gestion du dialogue. Par contre, à chaque demande de l'utilisateur, il doit rechercher un agent de service capable de l'exécuter. Cette consultation est ouverte, amenant l'agent assistant à envoyer une demande d'offre en *broadcast*[14]. Les modèles de tâches sont distribués parmi les agents. Avant de pouvoir négocier les informations nécessaires à l'exécution d'une tâche, l'agent assistant doit recevoir de la part de

[14] La communication directe n'est pas exclue cependant, car l'agent assistant peut directement demander l'exécution d'une tâche à un agent de staff qu'il sait capable de l'exécuter.

l'agent de service une description de tout qui doit être récolté auprès de l'utilisateur. Comme point fort de ce modèle, l'entrée et la sortie des agents sont plus faciles ainsi que les modifications des agent de service (rajout de nouvelles fonctionnalités, par exemple) qui sont faites sans interruption de l'agent assistant. Comme point faible de ce modèle, l'augmentation d'échanges de messages entraîne à une plus grande complexité de la politique de présentation.

Nous n'avions d'autre choix que de tester les deux situations pour savoir si l'architecture de l'interface était suffisamment adaptée à ces deux modèles. Nous avons réalisé deux prototypes de SMA distincts pour tester les deux modèles : un premier prototype dans le cadre du développement d'une ontologie expérimentale pour le projet européen TerreGov et un autre prototype dans le cadre d'un système de gestion d'activités quotidiennes d'un chercheur.

4. Les systèmes multi-agents réalisés pour valider l'interface conversationnelle

Cette section présente les deux prototypes de SMA que nous avons réalisés pour valider nos recherches.

4.1. Un prototype expérimental pour le projet TerreGov

Nous allons démarrer par le prototype centralisé, développé en s'appuyant sur le projet européen TerreGov (Moulin et al., 2005). Le projet TerreGov est un projet européen intégré qui rassemble seize partenaires (dont l'UTC) de huit pays différents autour de la problématique du eGovernment. Plus précisément, TerreGov a pour ambition d'améliorer l'efficacité des fonctionnaires territoriaux dans le secteur de l'aide sociale.

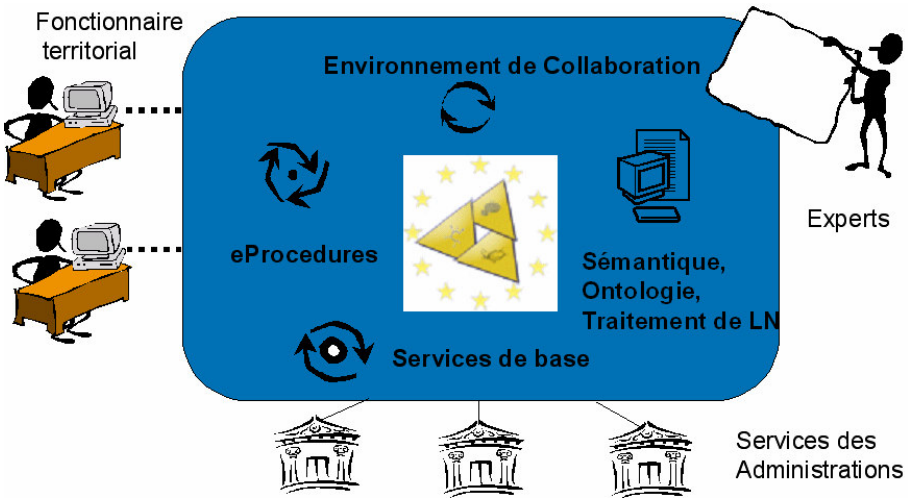


Figure 79 : Vision générale du projet TerreGov (extrait du site web du projet).

TerreGov propose la mise en place d'une plate-forme (voir vision générale sur la Figure 79) gérant des services web, avec l'addition d'une couche sémantique qui s'avère nécessaire pour :

- avoir une meilleure organisation de ces services (indexation conceptuelle) ;
- organiser des bases de connaissances construites progressivement qui seront mises à disposition des acteurs ;
- mettre en place un mécanisme de gestion de dialogue en langage naturel entre les utilisateurs (les fonctionnaires territoriaux) et le système.

Cette couche sémantique comporte un en ensemble d'ontologies (écrites en OWL).

Une première version de l'ontologie du domaine a été conçue à l'UTC, dont un extrait est montré sur la Figure 80. Nous avons récupéré un extrait de cette ontologie et nous l'avons utilisée comme source de connaissances du domaine.

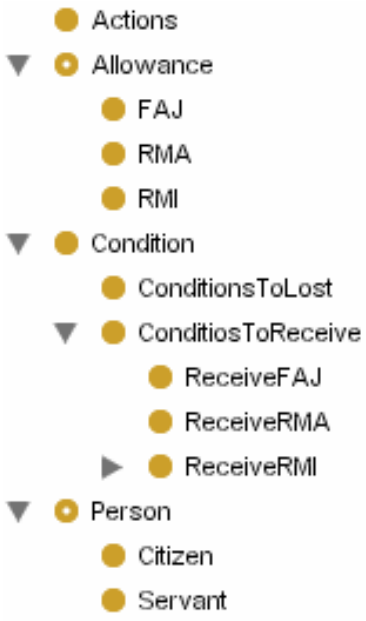


Figure 80 : L'extrait de l'ontologie du domaine en TerreGov.

L'ontologie du domaine contient des informations concernant le processus d'allocation de revenu, comme le RMI (Revenu minimum d'insertion) et le RMA (Revenu minimum d'activité) et permet de développer des dialogues simples comme celui montré du Tableau 11.

AP (1):	Welcome to the Voice Personal Assistant.
USR (2):	May you list all conditions to receive an RMI?
AP (3):	Yes. The candidate must: live in France, be older than 25, have income not higher than 417,88 € per month.
USR (4):	Start an RMI application.
AP (5):	What is you age?
USR (6):	26.
AP (7):	What is your monthly income?
USR (8):	300 euros.
AP (9):	Where do you live?
USR (10):	…

Tableau 11 : Un exemple de dialogue.

Les tâches du système, déclenchées à partir des énoncés, sont décrites comme le montrent les figures 60 et 61 suivant le schéma général présenté dans le Chapitre 4.

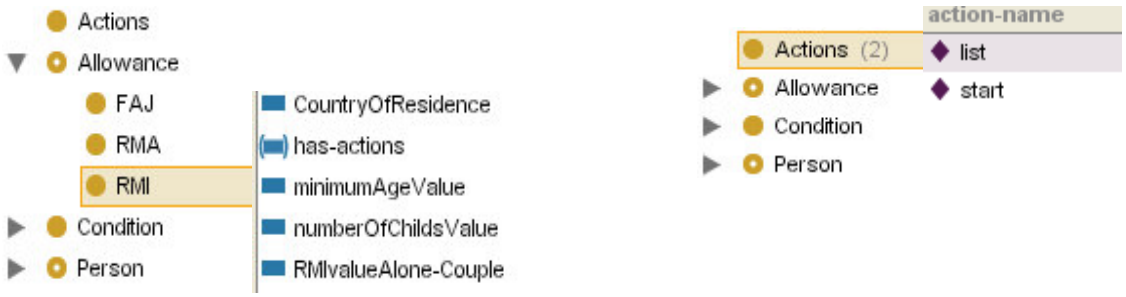


Figure 81 : Les attributs du concept RMI (à gauche) et quelques actions (à droite).

Le gestionnaire de dialogue, dans ce modèle centralisé, n'a pas besoin de chercher la description d'une tâche auprès d'un agent de service capable de l'exécuter, puisque toutes les tâches lui sont connues. Par exemple, à partir de l'entrée :

USR (4): Start an RMI application, la requête formelle:

`(start (RMI))`, est construite. S'il s'agissait du modèle distribué, l'agent assistant enverrait en *broadcast* aux agents de la coterie une demande pour d'obtenir, soit le résultat d'exécution de la tâche (si un des agents de service est capable de lui fournir à partir des informations reçues), soit la spécification des informations devrant être négociées auprès de l'utilisateur pour la future exécution de la tâche. Dans le modèle centralisé, chaque concept peut avoir une liste d'actions qui le relie à une tâche. Dans notre exemple, l'action *start* montrée sur la Figure 81 (à droite) est liée à la tâche montrée sur la Figure 83.

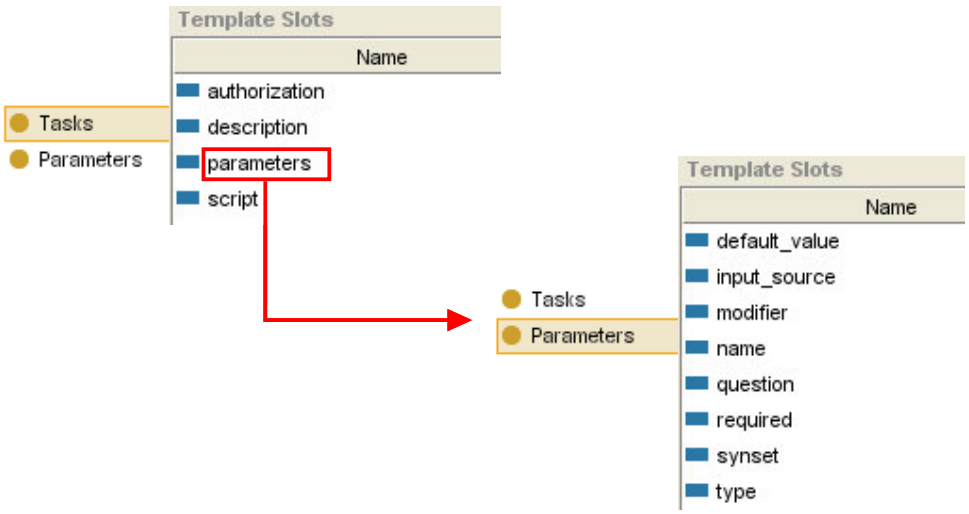


Figure 82 : Le modèle de tâches et ses paramètres.

Le gestionnaire de dialogue charge la tâche et démarre le processus de remplissage de paramètres jusqu'au point où ils sont tous remplis.

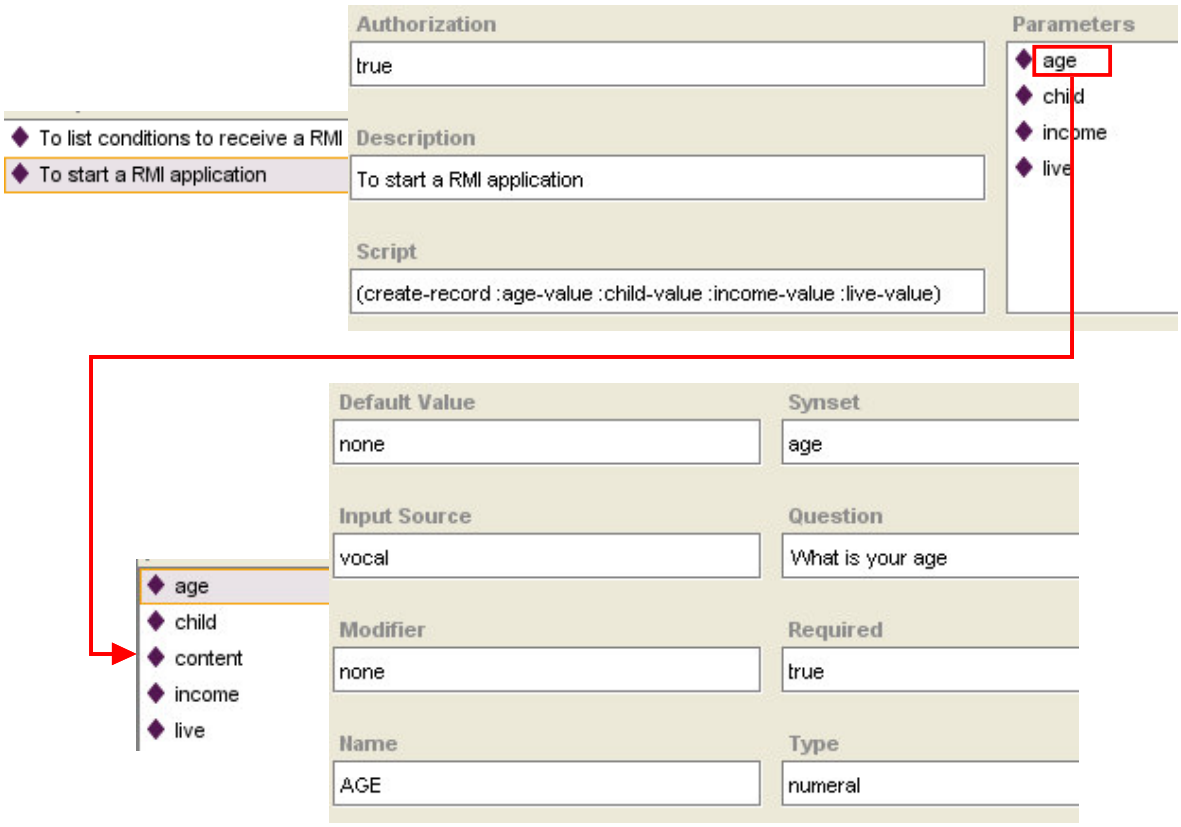


Figure 83 : La tâche *Start an RMI application* (en haut) et l'attribut *age* (en bas).

Une fois tous les paramètres remplis, le champ *script* est construit et envoyé à l'agent de service pour exécution.

Le diagramme de séquence de la Figure 84 illustre clairement le rôle de l'agent assistant dans le modèle centralisé. L'agent assistant négocie toutes les informations (paramètres de la tâche *Start an RMI application*) avant de les envoyer à l'agent de service concerné. À l'agent de service reste l'exécution de la tâche et l'envoi du résultat.

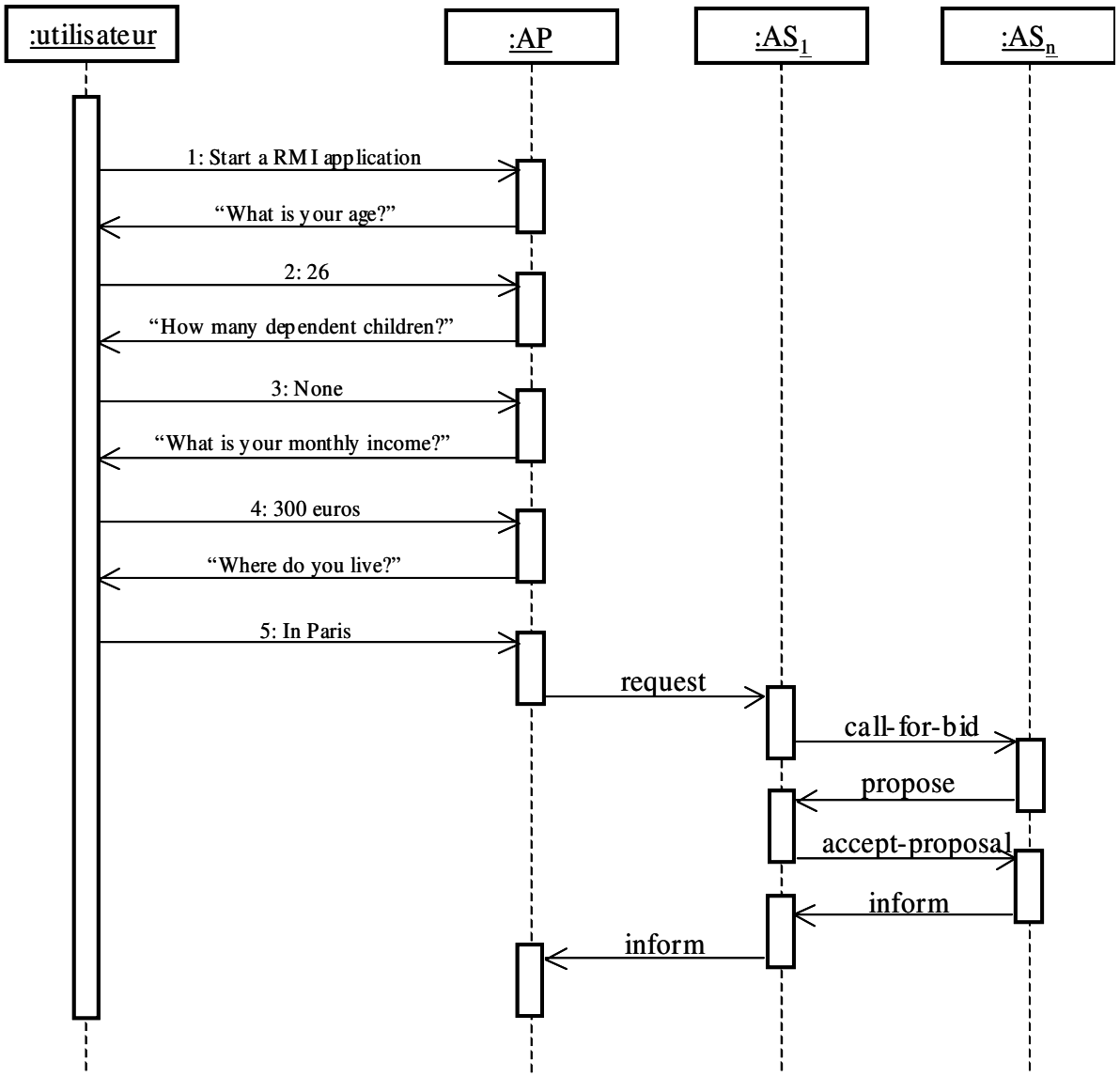


Figure 84 : Diagramme de séquence pour le processus de remplissage de la tâche.

Nous présentons dans la section suivante le prototype qui réalise le modèle distribué.

4.2. Un SMA de gestion d'activités quotidiennes

Ce deuxième prototype met en place le modèle distribué, où les tâches sont décrites dans chaque agent de service. Le but de ce prototype est de modéliser une partie d'un environnement de travail coopératif au sein d'un groupe de recherche. Cet environnement offre des services personnalisés réalisés par des agents aux participants d'un projet. La coordination est faite par un agent assistant. La Figure 85 présente les agents du prototype. Nous notons la présence de deux catégories d'agents de service : les agents de staff (*Agenda Agent* et *Addressbook Agent*) et les agents de service généraux (*Email Agent* et *Search-Doc Agent*).

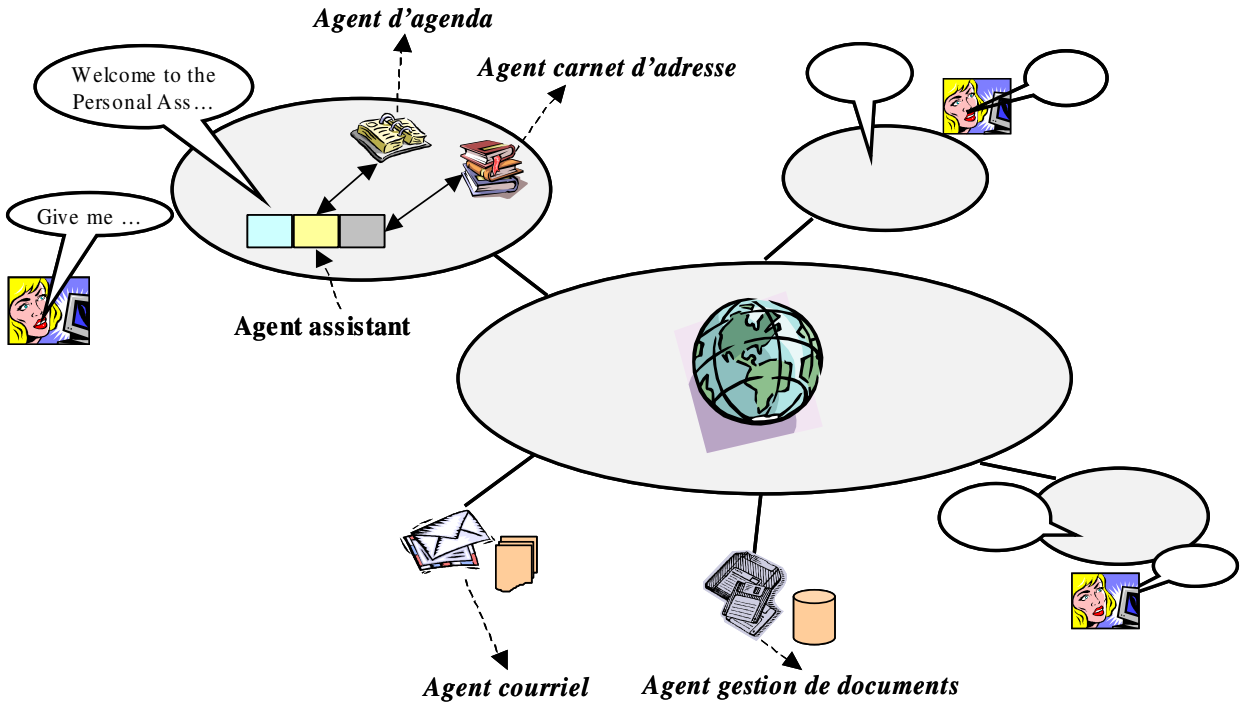


Figure 85 : Les agents du prototype.

Ce prototype suit le modèle distribué, où la description des tâches que chaque agent de service est capable d'accomplir reste « collée » à celui-ci : l'agent assistant ne connaît pas les tâches accomplies par chaque agent.

4.2.1 Les rôles des agents du système multi-agents

Nous décrivons, de façon succincte, les rôles des agents par rapport à l'ensemble du prototype.

Agent assistant personnel

Bien évidement, l'agent assistant a comme principal rôle de réaliser l'interface entre l'utilisateur et le système. Il est censé dialoguer avec l'utilisateur et apprendre à partir des interactions pour adapter son comportement aux préférences de l'utilisateur. Toutefois, l'adaptation de l'agent assistant à l'utilisateur n'est pas le sujet central de cette thèse.

En plus du dialogue, l'agent assistant est chargé de déléguer les services demandés par l'utilisateur à d'autres agents. Par principe, à part le dialogue avec l'utilisateur, l'agent assistant ne fournit aucun autre service. Par exemple, si l'utilisateur souhaite envoyer un e-mail à un collègue, l'agent assistant fait une requête à un agent de service qui est capable de l'envoyer.

Agent agenda

Le rôle d'un agent d'agenda est d'organiser l'agenda de l'utilisateur, ce qui le place comme un agent de staff. Cet agent emmagasine les événements de l'agenda de son maître et offre des services de consultation par événement ou par catégorie d'événement. Les registres des événements sont stockés sous forme de listes dans un fichier textuel séquentiel.

Agent carnet d'adresses

L'agent carnet d'adresses est un autre agent de staff chargé de la gestion du carnet d'adresses de l'utilisateur. Comme l'agent d'agenda, il emmagasine les coordonnées sous forme de listes, exploitables sur consultation par valeur d'un champ (nom de quelqu'un, numéro de téléphone, etc.) ou groupe de champs.

Agent courriel

Le rôle de l'agent courriel est d'ouvrir ou d'envoyer un courriel. Son travail étant plus général, il a été défini comme un agent de service, offrant des services à plusieurs agents assistants différents. Si nécessaire, il peut prendre contact avec les agents de gestion de carnet d'adresses pour demander l'adresse électronique du destinataire d'un message.

Agent gestion de documents

Cet agent est directement lié au processus de gestion de documents dans un projet développé de façon coopérative, ce qui fait de lui un agent de service d'ordre général. Ses principales compétences sont l'enregistrement et la recherche documentaire sur une base de documents de projet.

Tous les agents du SMA ont la capacité d'identifier une requête d'exécution de tâche qui les concerne, à partir d'un message lancé par un autre agent. Ils savent aussi formater un message de description de tâche à envoyer à l'agent assistant pour que celui-ci négocie les informations manquantes auprès de l'utilisateur.

4.2.2 Le langage de contenu

Pour faciliter les échanges entres les agents, nous avons créé un langage de contenu, succinctement présenté dans le tableau ci-dessous :

Message et arguments	**Description**
INFO content	Message utilisé par un agent de service pour envoyer un contenu informatif à l'agent assistant

REQUEST-INFO question-description msg-id	Message qui décrit une requête à l'utilisateur
INFO-REQUESTED msg-id content	Message de retour à une requête d'information
REQUEST-INFO-HELP msg-id	L'agent assistant envoie ce message aux agents pour leur demander une description de leurs compétences (services)
INFO-HELP-REQUESTED msg-id content	Message de retour à un REQUEST-HELP-INFO
TASK-SEARCH script msg-id	Message envoyé par l'agent assistant pour rechercher un agent de service capable d'accomplir une tâche
TASK-FORMAT task-description msg-id	Message envoyé par un agent de service à l'agent assistant lui spécifiant les informations à négocier auprès de l'utilisateur
TASK-REQUEST script msg-id	Message envoyé par un agent en direction d'un autre demandant l'exécution d'une tâche
TASK-RESULT msg-id response content	Ce message contient le résultat de l'exécution d'une tâche
TASK-CONFIRMATION msg-id content	Message de confirmation de l'exécution d'une tâche

Tableau 12 : Les messages du langage de contenu.

L'Annexe II présente ce langage de contenu en détails.

Ces messages permettent aux agents de coordonner la recherche d'un agent de service et la requête d'exécution de tâches, ainsi comme l'envoi d'informations à l'agent assistant pour être affichées à l'utilisateur. Ces messages sont encapsulés dans des messages du protocole de communication de la plate-forme OMAS.

4.2.3 L'ontologie du domaine

Dans le contexte du prototype, nous avons conçu une ontologie du domaine, suivant les principes de construction d'ontologies que nous avons listés dans la section 5.3.1 du Chapitre 4. Un extrait de l'ontologie est montré sur la Figure 86 (la même ontologie est représentée graphiquement sur la Figure 47).

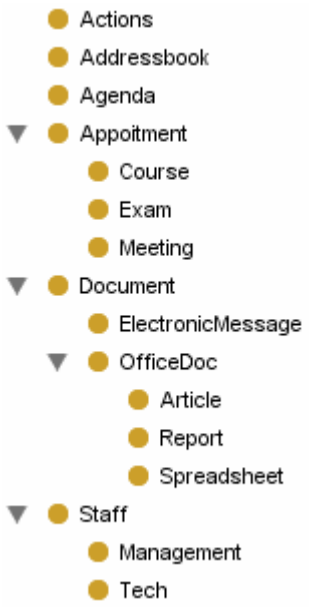


Figure 86 : L'ontologie du domaine (représentée selon le format Protégé).

Nous attirons l'attention sur l'absence d'une ontologie de tâches dans l'agent assistant. Les tâches sont dans chaque agent de service, bien entendu.

4.2.4 Déroulement de la recherche d'un fournisseur de service

Tout le processus d'analyse lexicale, syntaxique et sémantique a été montré dans le chapitre précédent. Nous montrons ici le principe de recherche d'un agent pour exécuter les tâches décryptées par l'agent assistant. Pour cela, nous allons étudier un exemple simple d'envoi de courriel. Tout le processus est présenté dans le diagramme de séquence montré sur la Figure 87.

À partir de l'entrée :

USR (1): Send an email to Mike, la requête formelle:

`(send (ElectronicMessage (:receiver "Mike")))`, est construite.

S'il s'agissait du modèle centralisé, l'agent assistant pourrait trouver, dans l'ontologie du domaine, l'indication de quelle tâche déclencher, puis aller chercher toutes les informations nécessaires dans l'ontologie de tâches. Comme ce n'est pas le cas, l'agent assistant va formater un message du type TASK-SEARCH et l'envoyer en *broadcast* aux agents de la coterie :

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

Le but poursuivi par l'agent assistant est d'obtenir, soit le résultat d'exécution de la tâche (si un des agents de service est capable de le lui fournir à partir des informations reçues), soit un

message du type TASK-FORMAT, spécifiant les informations à être négociées auprès de l'utilisateur. Ce processus d'envoi et de collecte des messages est contrôlé par OMAS. Dans la version actuelle, en cas de plusieurs réponses, l'agent assistant va prendre la première qui lui arrive.

Les agents de service reçoivent le message TASK-SEARCH et analysent le contenu de l'argument *script*. Pour pouvoir l'analyser, chaque agent utilise sa propre ontologie (nous rappelons que chaque agent possède ses propres connaissances).

L'agent de service courriel est directement concerné, ce qui l'amène à formater un message TASK-FORMAT et l'envoyer à l'agent assistant :

```
(TASK-FORMAT (<Task>
      <prm prm_id="1">
            <name>RECEIVER</name>
            <synset>receiver to</synset>
            <question>Who is the receiver</question>
            <type>propern</type>
            <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>
            <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)
```

Une fois le message arrivé, l'agent assistant le rajoute à la file de messages de haute priorité. Lors du traitement du message, le gestionnaire de dialogue respectera la spécification reçue, interrogeant l'utilisateur pour obtenir chaque paramètre manquant, suivant l'ordre dans lequel ils apparaissent. Au fur et à mesure des entrées, l'instance de concept en mémoire est mise à jour par le gestionnaire de dialogue. Le processus s'achève par le remplissage du champ *script*. L'agent assistant peut alors dépiler la tâche de la pile de tâches du gestionnaire de dialogue et l'envoyer à l'exécution, formatant un message TASK-REQUEST :

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

Après l'envoi, l'agent assistant met à jour la liste de tâches en exécution par d'autres agents.

En exécutant la tâche demandée, l'agent courriel doit trouver l'adresse électronique de *Mike*. Il envoie un message TASK-REQUEST à l'agent carnet d'adresse :

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

Après l'arrivée de la réponse provenant de l'agent carnet d'adresse et de l'effective exécution de la tâche, l'agent courriel peut retourner un message TASK-CONFIRMATION à l'agent assistant. L'agent courriel donc envoie le message :

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

Aussitôt reçu, le message est classé en basse priorité et empilé dans la file de messages de basse priorité.

Le diagramme de séquence montré sur la Figure 87 illustre en détails tout le processus de négociation entre l'utilisateur et l'agent assistant et entre celui-ci et les autres agents.

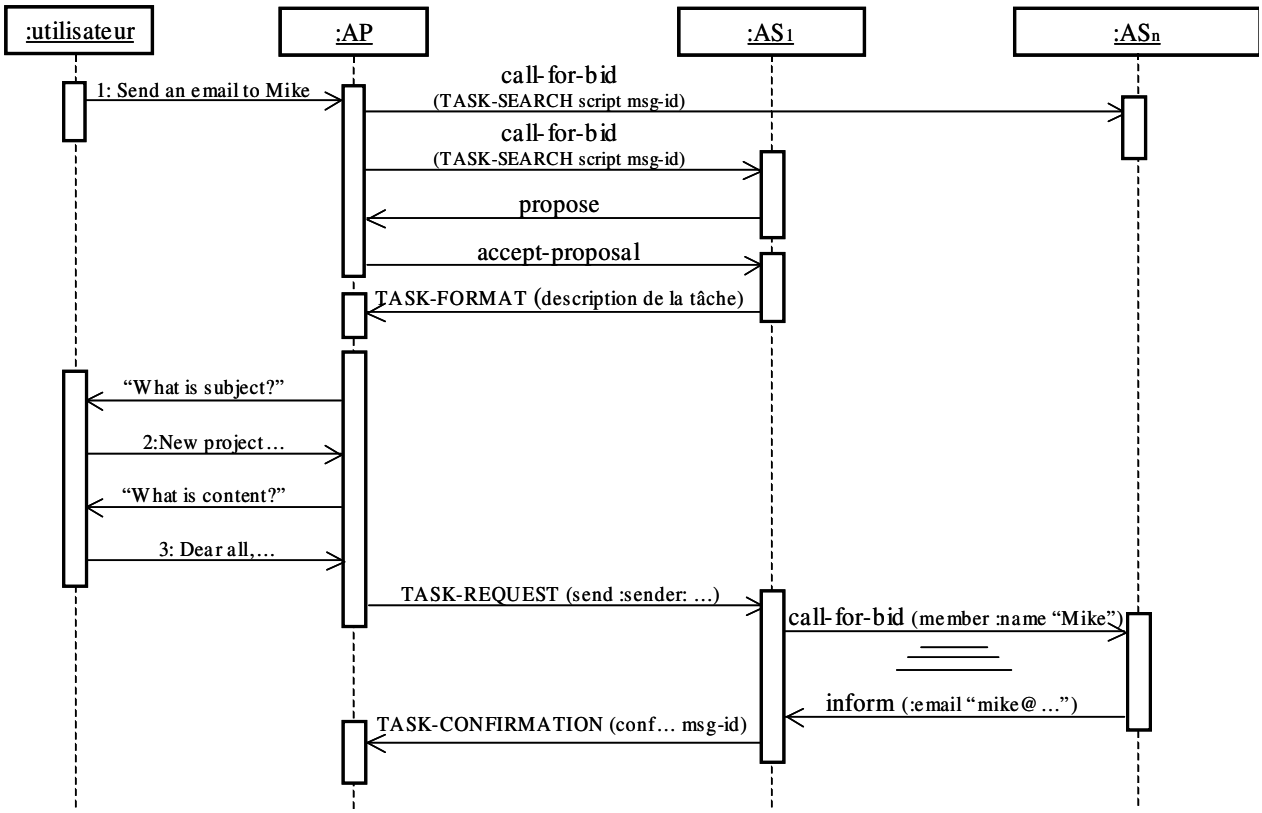


Figure 87 : Diagramme de séquence pour le processus de remplissage de la tâche.

L'agent de service qui reçoit un message TASK-REQUEST peut, en fonction de la demande, ou renvoyer un message du type TASK-CONFIRMATION ou renvoyer un message du type TASK-RESULT, comme l'illustre le diagramme de la Figure 88.

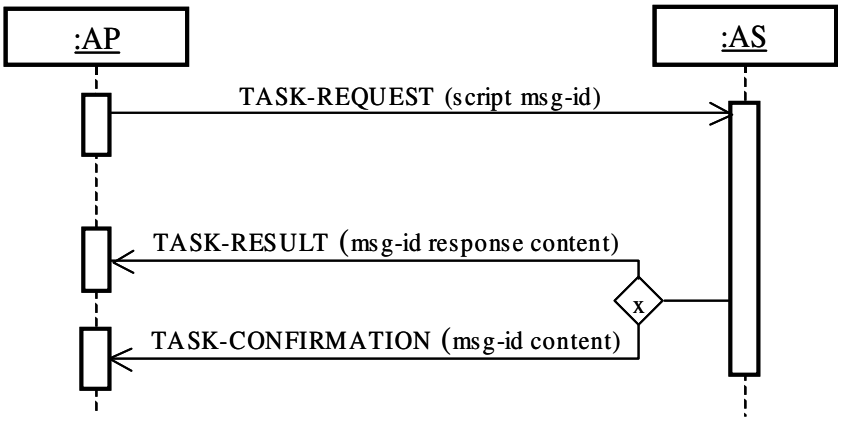


Figure 88 : Diagramme de séquence du résultat d'une tâche.

4.2.5 L'identification de la tâche par l'agent de service

En recevant un message du type TASK-SEARCH ou TASK-REQUEST, l'agent de service est capable de déterminer si ce message le concerne ou pas, en confrontant la description de la demande (champ *script*) et sa propre ontologie du domaine. Selon le modèle général d'agent

dont nous disposons, chaque agent de la coterie possède sa propre ontologie du domaine et de tâches. Il sait s'en servir pour interpréter et donner suite aux messages qui lui sont adressés.

4.3. La politique d'affichage d'informations et la gestion des mémoires en action

Dans ce contexte distribué, la politique d'affichage d'informations et les mémoires sont mises à l'épreuve. Nous allons donner un exemple à travers une petite conversation du travail conjoint de la politique d'affichage d'informations et de la gestion des mémoires au cours du déroulement du dialogue. Les scénarios (Figure 89, Figure 90 et Figure 91) montrent la situation courante à trois moments différents d'une même conversation. Les figures montrent les énoncés de la conversation (fenêtre du haut), les messages provenant d'autres agents (fenêtre au centre) et les mémoires (fenêtre du bas).

La Figure 89 capture la situation où deux tâches ont été exécutées et une troisième est en état de remplissage (voir fenêtre du bas). La confirmation de l'exécution de la deuxième tâche a été reçue, classée comme un message de basse priorité et affichée dans la bannette appropriée (fenêtre au centre). Au même moment est arrivée une requête d'un agent de service. Elle est empilée et attendra jusqu'au moment où la pile de tâches sera vide. Cette procédure évite le changement de contexte lors des questions.

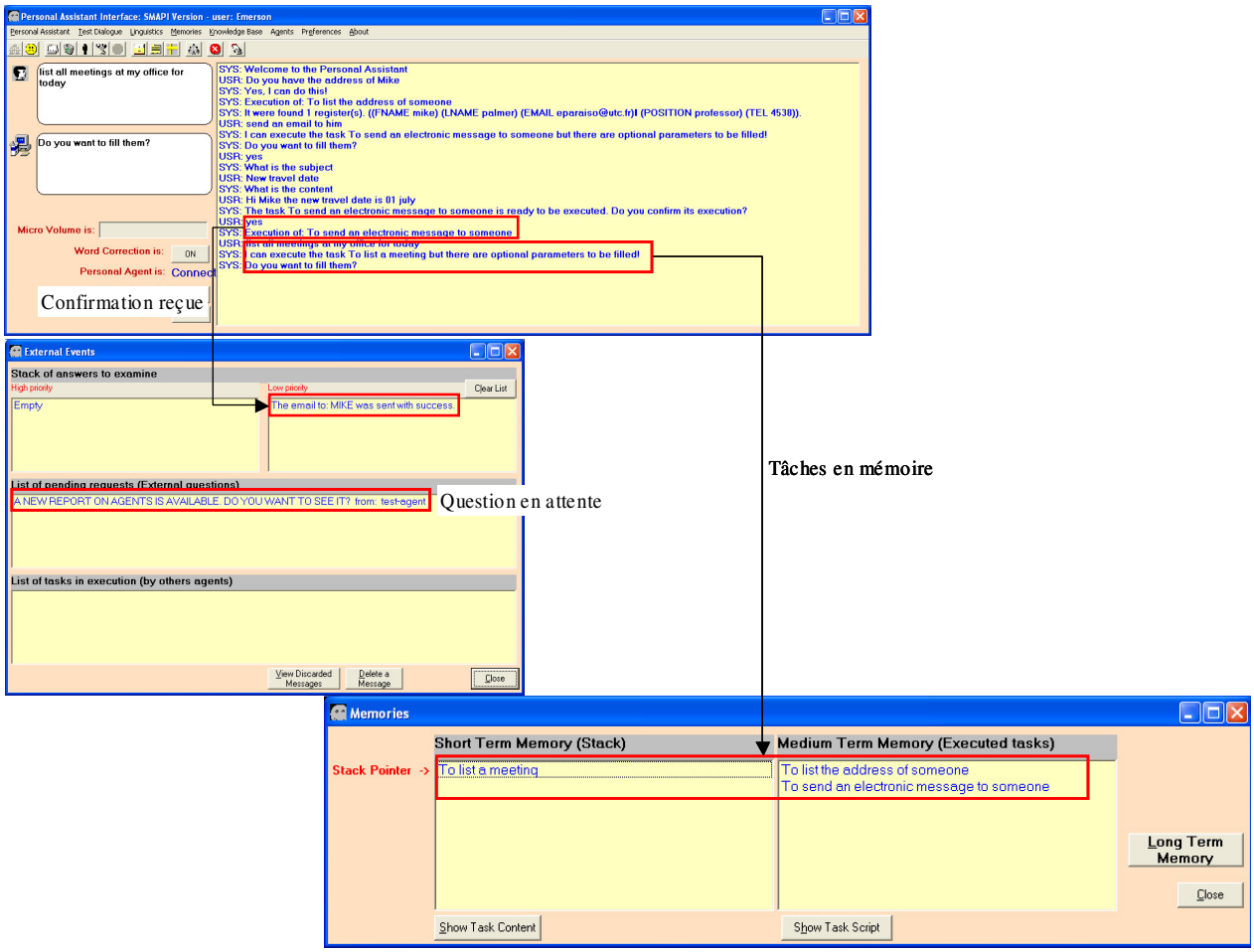


Figure 89 : Le premier scénario.

La conversation suit et arrive au scénario de la Figure 90, où la troisième tâche est envoyée à l'exécution. Comme la pile est vidée (fenêtre du bas), l'agent assistant peut poser la requête provenant de l'agent de service. En arrivant, le résultat de l'exécution de la troisième tâche est aussitôt classé en priorité haute et empilé dans la file de messages de priorité haute (fenêtre au centre).

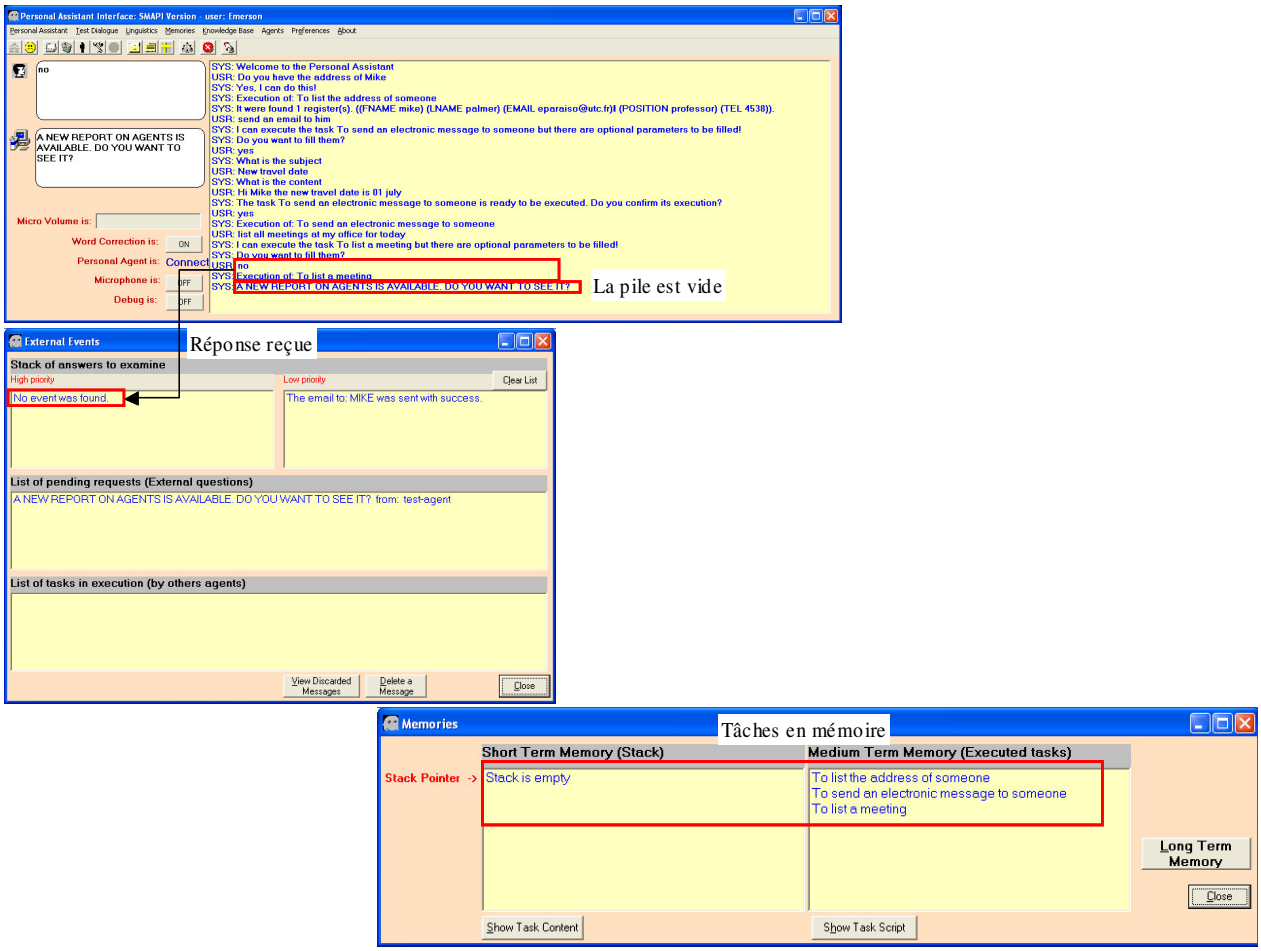


Figure 90 : La continuation du dialogue.

La conversation est finie dans le troisième scénario, lorsque l'utilisateur a répondu à la requête provenant de l'agent de service et l'agent assistant a imprimé le résultat de la dernière tâche en exécution. On note, sur la Figure 91, le statut final des mémoires et des piles.

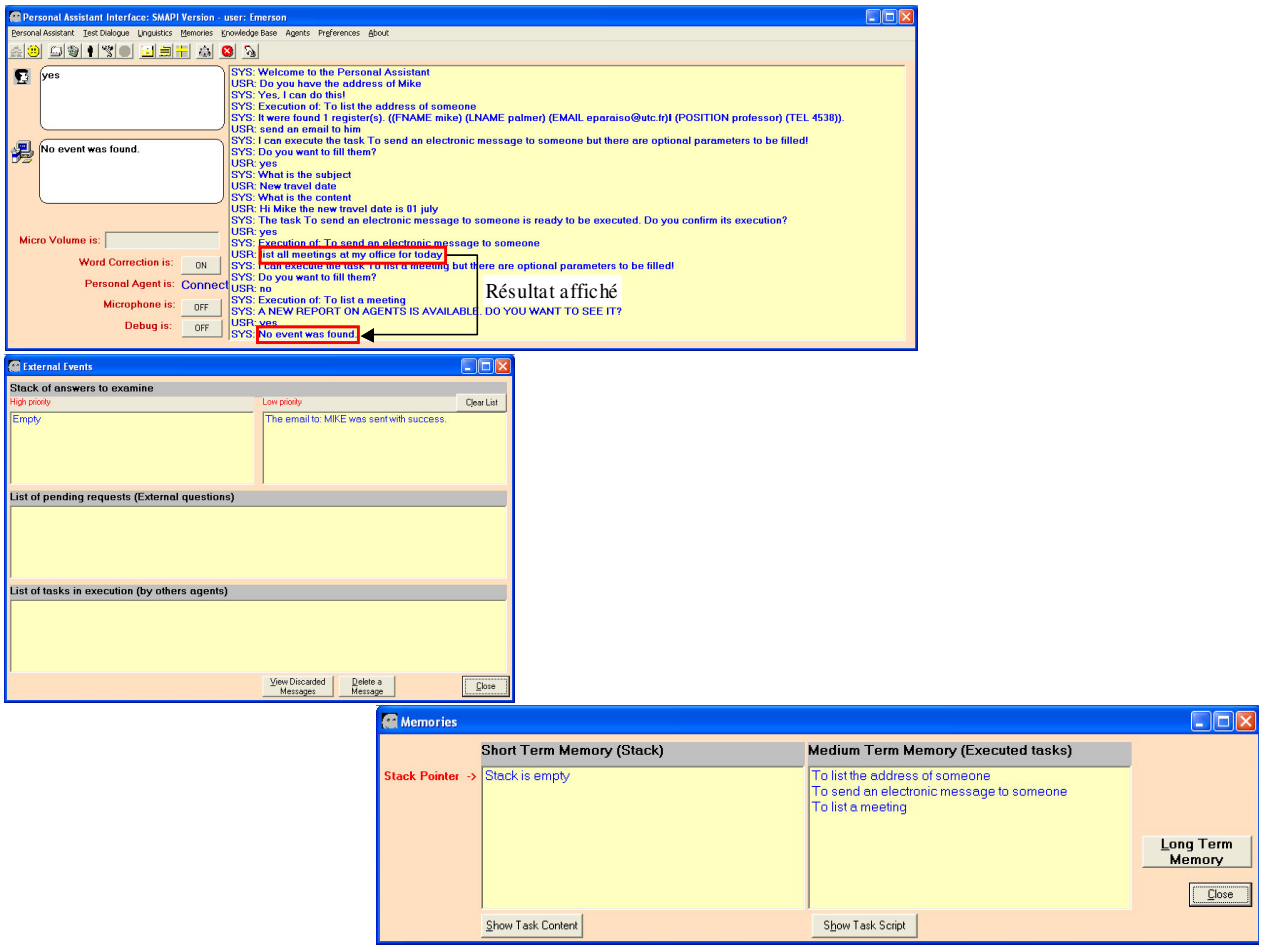


Figure 91 : La fin de la conversation.

5. Expérimentations

La réalisation des deux prototypes que nous venons de présenter nous a permis de mener quelques expérimentations pour :

- évaluer l'interface conversationnelle en termes quantitatifs et qualitatifs ;
- évaluer la présence d'un avatar pour animer l'interface de l'agent assistant ;
- mesurer l'amélioration de la capacité et de la qualité de l'aide de l'agent assistant.

Ces expérimentations ont été réalisées au sein de notre laboratoire à l'UTC avec la participation de quelques étudiants volontaires qui ont assumés le rôle d'utilisateur. Les utilisateurs ont été brièvement formés pour utiliser l'agent assistant.

5.1. Évaluation de l'interface conversationnelle

Dans cette expérimentation nous établissons un tableau évaluatif de l'interface conversationnelle. Au cours de l'expérimentation, nous avons recueilli des données statistiques afin de dresser un tableau de mesures qualitatives et quantitatives (voir Tableau 13), quelques-unes équivalentes aux mesures définies dans le test d'efficacité PARADISE (Walker et al., 1997).

Données relevées
Nombre d'entrées hors sujets acceptées
Temps entre l'identification et le déclenchement d'une tâche
Nombre d'offres d'aide au cours d'une conversation
Nombre de mots dites/nombre de mots non reconnus
Nombre d'échanges (turn-takings)
Nombre moyen de mots par entrée
Nombre d'entrées refusées

Tableau 13 : Les mesures quantitatives et qualitatives.

Les testeurs ont passé une phase préliminaire d'apprentissage d'environ 30 minutes (lire un texte) pour calibrer le dispositif de reconnaissance de la parole[15].

Un groupe de 4 utilisateurs a participé à cette expérimentation. Ils ont été invités à tester l'agent assistant en lui demandant d'accomplir quelques tâches préalablement établies dans le cadre du prototype de gestion d'activités quotidiennes. Une fois familiarisé avec l'agent, les utilisateurs ont été invités à démarrer les mêmes tâches, les unes après les autres et dans une seule session. Chaque testeur devrait demander à l'agent assistant l'exécution des quatre tâches :

1. une demande d'aide ;
2. la recherche et l'ouverture d'un article sur un sujet de l'intérêt de l'utilisateur ;
3. l'envoi d'un courriel à deux autres utilisateurs (préalablement enregistrées dans le carnet d'adresse) ;
4. l'affichage des rendez-vous fixés comptant comme participant un autre utilisateur et pour un jour donné.

[15] Il faut préciser que cette étape initiale n'est pas suffisante pour permettre au moteur de reconnaissance attendre ses meilleurs taux de reconnaissance. En fait, plus on l'utilise, plus précis seront ses résultats.

Bien évidement, les résultats de l'exécution des tâches ont été différents pour chaque testeur. Cependant, tous les résultats ont été correctement trouvés en fonction des données présentées par les utilisateurs.

5.1.1 Les données relevées

Le Tableau 14 présente les données relevées selon la procédure suivante : chaque utilisateur a été invité à tester l'agent assistant, installée dans un PC doté d'un processeur Pentium IV 3 GHz avec 512 Mo, placée dans une salle vidée pour l'occasion. Les données sont enregistrées dans un fichier au fur et à mesure que la saisie des entrées se déroule (un extrait est montré dans l'Annexe III).

Données relevées	**Testeur 1**	**Testeur 2**	**Testeur 3**	**Testeur 4**
Durée de la session en minutes	9	24	13	16
Nombre d'échanges (turn-takings)	46	78	64	82
Nombre d'entrées hors sujets acceptées	0	0	0	1
Nombre de tâches déclenchées	7	8	8	9
Temps moyen pour le déclenchement d'une tâche (en secondes)	13.86	19.86	15.14	20.67
Nombre d'offres d'aide spontanée[16]	0	1	1	2
Nombre de mots non reconnues	4	5	4	7
Nombre moyen de mots dits par entrée	4.63	3.83	4.05	3.45
Nombre d'entrées refusées	4	9	5	11

Tableau 14 : Les données relevées au cours de l'expérimentation.

La durée d'une session est le temps passé entre le démarrage et la fin de la session (en appuyant sur le bouton approprié). Le nombre d'échanges est obtenu en sommant le nombre d'entrées de chaque interlocuteur (l'agent assistant et le maître). Le nombre d'entrées hors sujets acceptées indique le nombre d'énoncés de l'utilisateur acceptés par l'agent comme étant du domaine, alors qu'ils ne l'étaient pas. Le nombre de tâches déclenchées est obtenu par le nombre de demandes de service auprès des agents de service. Le temps moyen pour le déclenchement d'une tâche est calculé par la formule suivante :

$$Tm = \frac{\sum_{k=1}^{tt} (D_k)}{(tt)}$$

où :

[16] Une offre d'aide est présentée lors que le gestionnaire de dialogue identifie au moins 5 entrées refusées de suite.

tt est le nombre de tâches exécutées ;

D$_k$ est la différence, en secondes, calculée entre l'identification de la tâche par le gestionnaire de dialogue et la construction du script envoyé à l'agent de service concerné.

Le nombre moyen de mots dits par entrée est obtenu par la division entre le nombre total de mots dits et le nombre d'entrées fournies par l'utilisateur.

5.1.2 Analyse des résultats

Les résultats obtenus nous montrent que l'ontologie du domaine joue bien son rôle de source des connaissances, permettant au gestionnaire de dialogue d'écarter les énoncés hors sujets et donner suite aux énoncés dans le domaine. Le seul énoncé hors sujet accepté a été obtenu lors que le testeur 4 dit : « *Please find me all Chinese papers* ». Par erreur, l'analyseur syntaxique a classé le mot *Chinese* comme un nom au lieu de classer comme un adjectif (en fait, le mot *Chinese* n'était pas présent dans le fichier contenant les adjectifs). À son tour, l'analyseur sémantique a interprété l'énoncé comme une demande de recherche de tous les articles sur le sujet *Chinese*. Nous avons corrigé ce problème.

Les résultats montrent une différence significative entre le temps moyen de déclenchement d'une tâche du premier testeur par rapport aux autres. À l'exception du premier testeur, tous les autres testeurs n'avaient pas utilisé notre interface conversationnelle auparavant. Cela explique les différences de performance entre le testeur 1, qui connaissait l'agent et savait comment les tâches sont traitées, et les autres. Cependant, il faut préciser que même les utilisateurs moins expérimentés ont réussi le déclenchement des tâches et obtenu les mêmes résultats. On note aussi que le fait de ne pas connaître *a priori* les tâches « réalisables » par les agents de service, n'a pas directement influencé les résultats de l'agent assistant.

En analysant le nombre de tâches déclenchées, on note qu'il est plus important que le nombre de tâches demandées (4 tâches). Cela est expliqué par le fait qu'une demande de tâche peut exiger l'envoi de plusieurs requêtes aux agents de service. Par exemple : quand on pose la question « *Can you find and open the article on electric motors* » le gestionnaire de dialogue crée deux requêtes formelles distinctes. Une offre d'aide spontanée résout aussi le déclenchement d'une tâche.

Le nombre de mots non reconnus est dû surtout à l'utilisation de prénoms de consonance d'origine non anglo-saxonne.

Le nombre d'échanges entre les testeurs varie en fonction du nombre d'entrées non reconnues fournies pour chacun d'entre eux.

Le nombre moyen de mots dits par énoncé est relativement faible (environ 4 mots). Cette moyenne doit être relativisée puisque parmi les entrées d'utilisateurs, on trouve celles formées par un seul mot (par exemple, une autorisation d'exécution de tâche *yes/no*). On note que le testeur 1 est responsable de la moyenne la plus grande. Cela est expliqué par le fait que ce testeur connaît le système et sait quels sont les paramètres nécessaires pour déclencher une tâche. Dans ce cas, il les fournit dans une seule entrée ou dans un nombre réduit d'entrées.

Concernant la politique d'affichage d'informations, elle a bien fonctionné, triant de façon correcte les messages provenant des agents de service.

5.2. Évaluation d'un avatar pour animer l'interface

Dans le Chapitre 2 section 3, nous avons exclu les avatars comme forme d'interface pour l'agent assistant travaillant dans un environnement professionnel. Néanmoins, nous voulions les évaluer, même d'une façon minimale. Nous avons alors intégré dans notre interface, un avatar (*James*) montré sur la Figure 92, avec le but de tester la capacité d'interaction et l'acceptabilité de l'utilisateur. L'avatar était chargé de synthétiser toutes les sorties sonores et graphiques de l'agent assistant. Pour cette réalisation, nous avons utilisé le package Microsoft Agent (Microsoft, 1999) qui met à disposition du développeur une interface programmable pour un ensemble d'avatars[17].

[17] Les mouvements et les comportements de l'avatar sont limités.

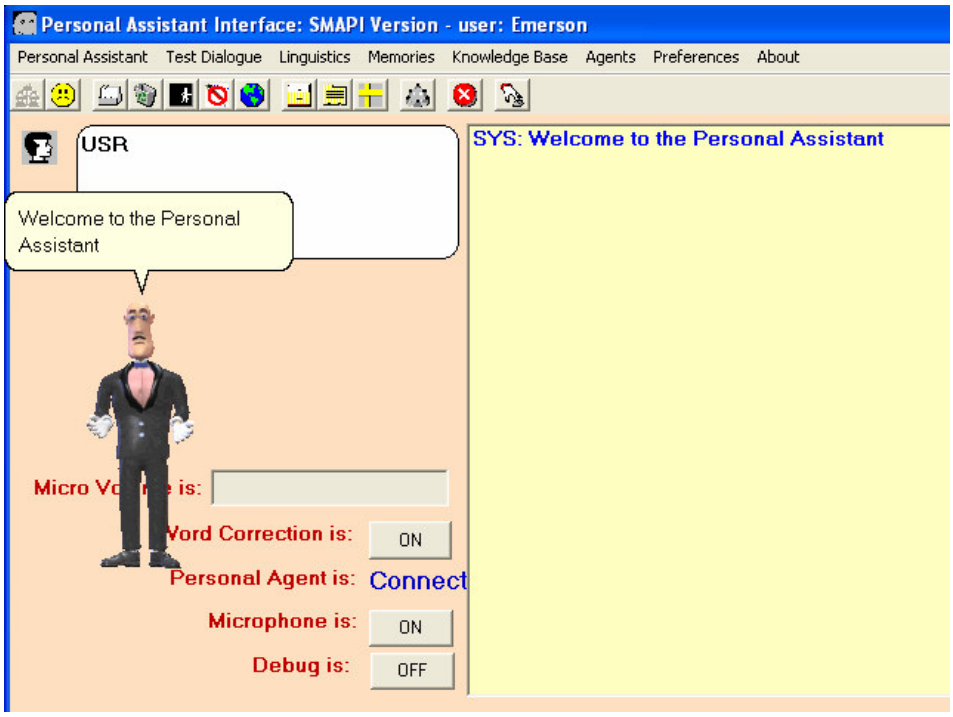


Figure 92 : L'avatar *James* de Microsoft.

Cette expérimentation a été menée conjointement à l'expérimentation précédente.

D'habitude, les avatars traduisent leur personnalité et leurs émotions par des gestes corporels et par un style linguistique particulier (Walker et al., 1997a). Dans notre agent assistant, le comportement de l'avatar est gouverné par la stratégie de dialogue et par le système de dialogue lui-même, puisque c'est lui qui produit les sorties du système. L'avatar doit donc suivre la stratégie générale de l'agent assistant. Comme la stratégie de l'agent assistant est de réduire les interruptions de l'utilisateur et l'espace de dialogue, le comportement de l'avatar reste figé aux énoncés du système, qui excluent toutes les expressions d'émotions et de politesse, limitant donc son action. Ce résultat était tout à fait attendu et confirme notre idée initiale que les agents animés ne pouvaient pas rajouter de la valeur à l'agent assistant dans les conditions d'utilisation que nous avons définies.

Malgré tout, les utilisateurs ont montré un certain degré d'attachement à l'avatar, tout en remarquant qu'il n'est pas indispensable pour le bon déroulement de la conversation.

Les agents animés peuvent être une bonne alternative pour des applications où l'utilisateur doit interagir avec l'application pendant une longue période de temps et où la stratégie de dialogue comporte une animation plus effective de l'avatar (comme les jeux ou les environnements interactifs d'apprentissage). Un travail dans cette direction est mené actuellement par Mi et Chen (2005), proposant un modèle basé sur les agents pour un avatar

immergé dans un environnement virtuel collaboratif. Toutefois, une étude plus approfondie, appliquant des agents animés plus interactifs, pourrait donner d'autres indices.

5.3. Évaluation de l'amélioration de la capacité et de la qualité de l'aide de l'agent assistant

L'une de nos hypothèses de départ était qu'une interface conversationnelle spécialement conçue pour l'agent assistant entraînerait une amélioration de la qualité de l'aide de ce dernier. Les expérimentations que nous avons menées nous ont permit de collecter plusieurs indices démontrant la validité de cette hypothèse, même si nous ne pouvons pas être catégorique. Le premier indice concerne la « traçabilité » des activités de l'utilisateur. En interprétant les énoncés de l'utilisateur et en les représentant de façon explicite nous contribuons à la réussite d'un certain nombre d'applications, on pense notamment à celles liées à la gestion de connaissances, dont l'efficacité dépend de l'extraction de nouvelles connaissances obtenues au fur et à mesure que l'utilisateur interagit avec le système. Cela est plus difficile quand on utilise une interface du type WIMP.

Le fait que l'agent assistant ait une interface plus conviviale, donne à l'utilisateur une motivation de plus pour utiliser son agent. Actuellement, l'agent assistant partage « l'attention » de l'utilisateur avec d'autres applications qui tournent sur sa machine. Si l'agent assistant est capable de rendre des services personnalisés à travers une interface appropriée, l'utilisateur va lui porter plus d'intérêt, en attendant le jour où l'agent assistant arrivera à offrir tous les services dont l'utilisateur a besoin.

Finalement, comme déjà indiqué par d'autres chercheurs ((Privat et al., 2000) ou encore (Allen et al., 2001a)), une interface conversationnelle permettra à des utilisateurs moins expérimentés ou atteints de limitations physiques, à être plus à l'aise avec l'agent assistant, en augmentant la perception de l'utilité du système.

6. Bilan des réalisations et des expérimentations

Dans cette section nous dressons un bilan sur la réalisation et les expérimentations de l'interface et de l'agent assistant.

6.1. Discussion du bilan des réalisations

La réalisation de deux prototypes nous a permis de mettre au point l'architecture générale de l'agent assistant et d'envisager quelques évolutions.

6.1.1 Analyse des prototypes dans une perspective de réalisation

Les prototypes, mis en service dans notre laboratoire, nous ont permis d'évaluer la validité de notre approche. L'architecture proposée se montre adéquate : les entrées sont correctement traitées à l'aide de l'ontologie du domaine, le gestionnaire de dialogue est capable d'identifier et de déclencher des tâches (avec l'appui des agents de service), la politique d'affichage d'information permet une organisation des informations affichées. Les deux formes d'organisation des agents (centralisée et distribuée) ont été testées. Le prototype distribué nous a permis d'affiner la politique d'affichage d'informations, fortement sollicitée dans cette configuration.

Une difficulté rencontrée fut le besoin d'intégrer plusieurs composants différentes pour constituer les prototypes. Cette difficulté est due substantiellement à la combinaison de différentes technologies, systèmes et langage de programmation. La liaison entre l'interface (codée en Visual Studio) et le noyau de l'agent (codé en Allegro-Common Lisp) représentait un point sensible. Le prototype nous a permis de vérifier la faisabilité de cette intégration, bien que l'utilisation d'une seule plate-forme de développement soit recommandée.

Enfin, l'architecture obtenue est faiblement connectée au restent de l'agent, ce qui favorise les modifications.

6.1.2 Les améliorations et les évolutions

De cette réalisation, nous avons dégagé quelques éléments de repère pour de futures améliorations et évolutions.

Vers une politique de présentation

Nous avons réalisé une politique d'affichage d'information, étape intermédiaire vers une politique de présentation. Une politique de présentation spécialement conçue pour l'agent assistant lui permettra d'être plus facilement accepté et adopté par l'utilisateur puisque celle-ci lui donnera une autonomie plus grande. Cette politique certainement donnera à l'agent la capacité de :

- analyser les informations qui arrivent, en prenant compte leur contexte et leur relation avec l'historique de travail de l'utilisateur ;
- regrouper les informations qui doivent être affichées selon leur domaine, facilitant leur interprétation par l'utilisateur ;
- différer la présentation d'un message en fonction de son contenu.

La politique de présentation est en réalité une voie de recherche intéressante pour l'évolution de l'agent.

Le traitement de la référence

Nous avons réalisé une partie du traitement de la référence. Le traitement de la référence doit être amélioré pour couvrir toutes les variantes possibles, comme celles produites par des pronoms réflexifs (*myself, yourself,* etc.).

Un mécanisme de synthèse de l'information

Le processus actuel de synthèse de l'information est primaire. Même si l'agent assistant est capable de synthétiser ses énoncés, ces réponses restent limitées. Nous sommes conscients que la génération du langage naturel est importante pour augmenter l'efficacité de l'agent assistant. Une étude plus approfondie doit être engagée, en particulier pour améliorer les messages d'éclaircissements. La synthèse des résultats doit aussi être élaborée avec plus de soin, pour éviter l'impression de données inutiles.

6.2. Discussion du bilan des expérimentations

Les expérimentations nous ont permis d'analyser la faisabilité de notre approche ainsi que de tirer plusieurs leçons utiles pour définir quelques expérimentations futures.

6.2.1 Analyse des résultats des expérimentations

Nous concluons, basés sur les expérimentations menées, que l'architecture de l'interface conversationnelle est faisable. Cela toutefois, peut ne pas être définitif si on arrive à développer une application réelle, à grange échelle, avec un processus d'évaluation préalablement établi.

Les expérimentations ont permis de démontrer aussi la prévisibilité de l'interface. 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 repérées à l'aide des ontologies.

Au cours des expérimentations, nous avons observé que les utilisateurs se sentent plus à l'aise lorsque le système leur offre des informations précises sur le déroulement des activités de l'assistant. Ainsi, les messages classés comme basse priorité par l'agent ne sont pas forcément inutiles (du point de vue des utilisateurs). Cela renforce le besoin de faire évoluer la politique d'affichage d'informations vers une politique de présentation.

6.2.2 Les expérimentations futures

Les expérimentations et les analyses doivent être poursuivies afin de nous aider à faire évoluer l'interface conversationnelle et l'agent assistant.

Évaluation à grande échelle

Bien évidement une évaluation à grande échelle, avec un nombre plus important d'utilisateurs, est la prochaine étape de nos expérimentations. Elle permettra de confirmer ou pas les résultats préliminaires obtenus et de nous guider vers des améliorations et évolutions.

Une expérimentation réelle nous permettra aussi de valider l'hypothèse selon laquelle l'interface conversationnelle améliore la qualité de l'aide de l'agent assistant.

Pour l'instant, il est prévu d'utiliser notre agent assistant en interne dans notre laboratoire, dans le cadre de quelques thèses de doctorat.

L'ontologie du domaine

Les ontologies du domaine, dans les deux réalisations, ont bien joué leur rôle de source de connaissances au cours de la gestion de dialogue. En revanche, nos expérimentations n'ont pas été suffisamment diversifiées pour permettre d'affirmer que les composants de base de l'ontologie du domaine listés dans la section 5.3.1 du Chapitre 4 sont suffisants pour exprimer les besoins de toutes les applications potentielles. Ce n'est qu'après plusieurs expérimentations, que nous aurons les indices nécessaires pour définitivement valider cette liste.

La gestion de la base de connaissances

Nous avons réalisé un mécanisme ad hoc de gestion de l'ontologie du domaine. Des expérimentations avec MOSS (Barthès, 1994) sont envisageables, car en sa toute nouvelle version, MOSS est capable de raisonner sur des ontologies écrites en SOL (*Simple Ontology Language*), ainsi que de produire ces ontologies en format OWL.

La mobilité de l'utilisateur

Nous ne pouvons pas négliger la capacité grandissante de déplacement des applications grâces aux dispositifs portables comme les PDAs (Personal Digital Assistant), favorisés par un environnement de plus en plus connecté. Ainsi, des recherches futures devraient être menées pour faire évoluer l'agent assistant vers les conditions où l'écran est limité ou quand il n'existe même pas.

Mesure d'efficacité du système de dialogue

Nous envisageons aussi d'évaluer constamment notre système de dialogue un utilisant un test de mesure d'efficacité, comme PARADISE (Walker et al, 1997). Ce test nous permettra d'évaluer des scénarios différents, comme le changement de stratégie de dialogue, et l'efficacité des divers modules qui composent le système de dialogue.

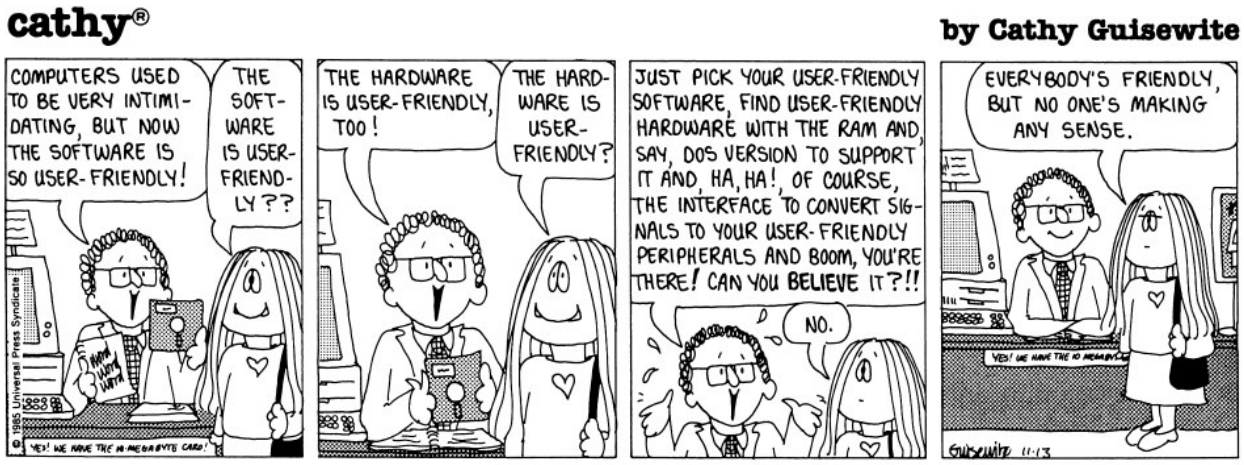
7. Synthèse et conclusions de ce chapitre

Ce chapitre a présenté les détails de la réalisation de l'interface conversationnelle ainsi que les résultats de quelques expérimentations. Nous avons décrit deux prototypes de système multi-agents développés pour valider l'interface de l'agent assistant.

Les prototypes, mis en service dans notre laboratoire, nous ont permis d'évaluer la validité de notre approche. Les expérimentations ont permis de démontrer la prévisibilité de l'interface. L'architecture proposée se montre adéquate.

Grâce à la réalisation de ces prototypes, nous avons identifié quelques voies de recherches futures pour poursuivre nos travaux, dont la conception et l'implémentation d'une politique de présentation et l'étude de l'impact de la mobilité de l'utilisateur.

CHAPITRE 6



Cathy © 1985 Cathy Guisewite. Reprinted by permission of Universal Press Syndicate.

Conclusions, contributions et perspectives

1. Rappel de la problématique

Dans cette thèse, nous avons abordé la conception d'une interface conversationnelle pour les agents assistants personnels. Nous avons examiné les divers types d'interfaces utilisées par des agents et nous avons choisi les interfaces conversationnelles comme interfaces idéales pour nos besoins. Notre conclusion est qu'une interface conversationnelle spécialement conçue pour l'agent assistant lui permettra d'être plus facilement piloté par l'utilisateur, améliorant ainsi la qualité de l'aide par rapport aux interfaces purement graphiques. Nous faisons l'hypothèse que l'agent assistant doit être l'interface privilégiée avec l'utilisateur, car il peut connaître ses préférences et il a la capacité de communiquer avec lui en langage naturel parlé.

La conception et la réalisation d'une interface conversationnelle pour l'agent assistant ne sont pas évidentes. Nous avons mis en relief les principaux défis au développement d'une telle interface, comme le besoin d'un mécanisme d'analyse syntaxique robuste, capable de traiter le langage parlé, pas forcément bien reconnu, la capacité d'ancrer sémantiquement l'agent sur les ontologies ou encore le besoin d'un gestionnaire de dialogue capable de gérer des conversations pour des domaines spécifiques.

Nous avons conçu une interface conversationnelle respectant les particularités de l'agent assistant et de l'environnement dans lequel il évolue : un système multi-agents. Pour valider cette approche, deux prototypes ont été mis en service dans notre laboratoire. À travers ces prototypes nous concluons que l'architecture proposée se montre adéquate : les entrées sont correctement traitées à l'aide de l'ontologie du domaine, le gestionnaire de dialogue est capable d'identifier et de déclencher des tâches (avec l'appui des agents de service) et la politique d'affichage d'information permet d'organiser les informations qui seront postées pour l'utilisateur. L'interface est prévisible, car les demandes impossibles, du type hors contexte, sont facilement repérées à l'aide des ontologies.

Nous sommes partis d'un modèle d'agent assistant générique proposé par Ramos (2000) et amélioré par Enembreck (2003), mais qui en pratique, se manquait d'une véritable interface avec l'utilisateur et méritait une étude plus approfondie. Nous estimons que ce n'est qu'après la mise en place d'une interface appropriée que le modèle générique de l'agent assistant peut effectivement être validé.

2. Nos principales contributions

Les recherches entreprises nous ont permis de dégager le concept d'interface conversationnelle pour une aide intelligente - ICAI. *Une ICAI est le résultat de la conjonction d'un mécanisme conversationnel en langage naturel parlé et de la gestion intelligente de ce mécanisme, permettant le déroulement d'un dialogue coopératif et capable de gérer le déclenchement de plusieurs tâches à la demande de l'utilisateur, avec un minimum d'effort de la part de ce dernier*. Cette interface conversationnelle peut être vue comme une architecture générique pour ceux qui veulent concevoir des agents assistants capables de dialoguer avec leur maître. D'une part, l'architecture est modulaire et faiblement couplée à l'agent assistant, ce qui donne au concepteur une flexibilité plus grande. D'autre part, nous montrons comment concevoir une ontologie du domaine capable d'assurer le traitement sémantique.

Notre approche s'appuie sur les ontologies, qui nous permettent d'interpréter les messages, provenant de l'utilisateur ou d'ailleurs. Les ontologies sont la seule forme de représentation des connaissances dont nous avons besoin. Nous avons montré pourquoi et comment il est possible d'utiliser les ontologies au cours des analyses syntaxique et sémantique dans le cadre d'un système de dialogue spécialement conçu pour être intégré à l'interface conversationnelle de l'agent assistant.

Le gestionnaire de dialogue dont dispose l'interface conversationnelle s'appuie sur les actes de langage directifs (ordre, question et réponse), dans le cadre d'une stratégie coopérative, capable de gérer plusieurs tâches simultanément et capable de traiter l'arrive de messages provenant de sources distinctes.

L'architecture de l'interface propose la séparation physique des connaissances du domaine et des tâches, ce qui favorise les évolutions et les mises à jour ainsi que l'utilisation des connaissances du domaine au niveau du traitement sémantique.

L'expérience obtenue dans cette recherche nous a conduit à proposer une évolution du modèle générique de l'agent assistant, avec un couplage plus léger entre l'interface utilisateur et le reste de l'agent assistant. Dans ce modèle, les modules de gestion du dialogue sont tous insérés dans l'interface et le module de contrôle de l'agent assistant n'a pas de tâches supplémentaires à surveiller. Le modèle comprend maintenant une politique d'affichage d'informations, capable de traiter les interruptions externes comme le traitement de questions (qui seront posées à l'utilisateur) provenant des agents de service. Il faut reconnaître cependant que cette politique d'affichage d'informations ne peut remplacer une politique de présentation.

3. Les perspectives des travaux futurs

Les prolongements à donner à ces travaux sont divers, car des nombreuses voies de recherches et de développements se sont ouvertes.

Du point de vue des réalisations et des expérimentations, nous aimerions évaluer l'agent assistant dans une application réelle, comptant un nombre plus important d'utilisateurs. Pour cela, il faudra finaliser le traitement du contexte (référence), penser à un système de synthèse de l'information plus performant, en particulier pour améliorer les messages d'éclaircissements et définir les critères spécifiques d'efficacité qui devront être atteints. Par ailleurs, nous

envisageons de faire évaluer constamment notre système de dialogue en utilisant un test de mesure d'efficacité comme PARADISE.

Concernant les voies de recherches futures, nous identifions plusieurs axes possibles, dont la conception et la mise en place d'une politique de présentation, l'étude de l'impact de la mobilité de l'utilisateur et l'approfondissement de l'étude du rôle des ontologies dans l'interprétation et dans la personnalisation de l'interaction entre l'utilisateur et son agent assistant.

Une politique de présentation spécialement conçue pour un agent assistant lui permettra d'être plus facilement accepté et adopté par l'utilisateur, puisque elle lui donnera une autonomie plus grande. La mise en place de cette politique donnera à l'agent la capacité de :

- analyser les informations qui arrivent, en prenant compte leur contexte et leur relation avec l'historique de travail de l'utilisateur ;
- regrouper les informations qui doivent être affichées selon leur domaine, facilitant leur interprétation par l'utilisateur ;
- différer la présentation d'un message en fonction de son contenu.

Nous aimerions également développer des recherches pour que l'agent assistant soit accessible par d'autres moyens, comme des dispositifs portables. Ces dispositifs peuvent imposer des contraintes intéressantes à étudier comme l'utilisation d'un écran de taille limitée ou une forme d'interaction exclusivement vocale (comme un téléphone).

Finalement, une autre perspective de recherche concerne les formes de représentations des connaissances, plus particulièrement les ontologies. Dans notre recherche, les ontologies ont montré leur utilité lors des traitements sémantiques. Cette recherche devrait être poursuivie pour mieux connaître la structure et le contenu idéal d'une telle ontologie.

4. Bilan des travaux publiés

Les travaux développés pendant la thèse ont donné lieu à plusieurs publications, dans les domaines des agents intelligents, des systèmes multi-agents et de l'interaction homme-machine (voir Tableau 15 récapitulatif).

Publications en chiffres	**Francophones**	**Internationales**	**Total**
Revues		1+1[18]	2
Actes de colloques	1	5	6
Chapitre de projet européen		1	1
Rapports Internes	2		2
TOTAL	**3**	**8**	**11**

Tableau 15 : Bilan chiffré des publications dérivées de la thèse.

[18] Publication en phase finale de révision.

Bibliographie

ALLEN, J., FERGUSON, G. et STENT, A. (2001a). *An architecture for more realistic conversational systems*. In: Proceedings of Intelligent User Interfaces (IUI-01), Santa Fe, NM, 2001.

ALLEN, J. F., BYRON, D. K., DZIKOVSKA, M., FERGUSON, G., GALESCU, L. et STENT, A. (2001b). *Towards Conversational Human-Computer Interaction*. AI Magazine 2001.

ANTOINE, J.-Y., GOULIAN, J. et VILLANEAU, J. (2003). *Quand le TAL robuste s'attaque au langage parle: Analyse incrementale pour la comprehension de la parole spontanee*. Traitement Automatique des Langues Naturelles, Batz-sur-Mer.

AUBRY, S., LENNE, D., THOUVENIN, I., et GUENAND, A. (2005). *VR Annotations for collaborative design*. In Proceedings HCII 2005 proceedings, 22-27 July 2005, Las Vegas, USA

AUSSENAC-GILLES, N. et MOTHE, J. (2004). *Ontologies as Background Knowledge to Explore Document Collections*. Dans : RIAO 2004, Avignon, 26 à 28 avril 2004. p. 129-142.

AUSTIN, J. L. (1962). *How to do things with words*. Harvard University Press, Cambridge, MA, 1962.

BAPTIST, L. (2000). GENESIS-II: A Language Generation Module for Conversational Systems, Thèse de doctorat présentée au MIT, USA.

BARBUCEANU, M., FOX, M., HONF, L., LALLEMENT, Y. et ZHANG, Z. (2003). *Building Agents for the Customer Service Front*. In Proceedings of the Fifteenth Innovative Applications on Artificial Intelligence Conference, Acapulco, Mexico.

BARTHÈS, J-P. A. (1994). *MOSS 3.2*. Memo UTC/GI/DI/N 111, Université de Technologie de Compiègne, Mars 1994.

BARTHÈS, J-P. A. (2002). *OMAS v 1.0 Technical reference*, Memo UTC/GI/DI/N 151, Janvier 2002.

BARTHÈS, J-P. A. (2005). *MOSS 5 Dialogs*, Memo UTC/GI/DI/N 189, Avril 2005.

BATEMAN, J. A. (1997). *The Theoretical Status of Ontologies in Natural Language Processing*. Proceedings of Workshops on Text Representation and Domain Modeling – Ideas from Linguistics and AI. 1997.

BERINGER, N., KARTAL U., LOUKA, K., SCHIEL, F. et TURK, U. (2002). *PROMISE - A Procedure for Multimodal Interactive System Evaluation*. Proceedings of Workshop on Multimodal Resources and Multimodal Systems Evaluation.

BEUN, R. J., VOS, E. de et WITTEMAN, C. (2003). *Embodied Conversational Agents: Effects on Memory Performance and Anthropomorphisation*. Springer Lecture Notes in AI, IVA, 315-319.

BEUST, P. (2000). Pour *une modélisation de la référenciation dans le dialogue homme-machine*. Actes de la Journée ATALA, 2000.

BISOGNIN, L. et PESTY, S. (2004). *Agents, Langage et Emotions : un prototype d'agent émotionnel*. Actes de la Journée d'étude AGENTAL : Agents et Langue, 2004.

BOISSIER, O. et GUESSOUM, Z. (2004). *Systèmes Multi-Agents : Défis Scientifiques et Nouveaux Usages*. Actes de la conférence JFSMA'04, Hermès/Lavoisier.

BRACHMAN, R. J. et SCHMOLZE, J. G. (1985). *An Overview of the KL-ONE Knowledge Representation System*. Cognitive Science, vol. 9, n°2, p. 171-216.

CAELEN, J. (2004). *Modèles de dialogue pour l'interaction homme-machine* In: Tutorial of the 16th Conférence Francophone sur l'Interaction Homme-Machine, Namur, Belgique, 2004.

CALLAWAY, C et LESTER, J. (2001). *Narrative Prose Generation*. Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, Seattle, WA, August 2001.

CHALON, R. et DAVID, B.T. (2004). *Modélisation de l'interaction collaborative dans les systèmes de Réalité Mixte*. Actes de la conférence 16e Conférence sur l'Interaction Homme-Machine (IHM 2004) Namur, Belgique.

CHAN-LAM, V. (1979). *Conception et réalisation d'une base de données ensembliste*. Thèse de Doctorat, Université de Technologie de Compiègne, Décembre 1979.

CHENG, J., KUMAR, B. et LAW, K. (2002). *A Question Answering System for Project Management Applications*. In: Advanced Engineering Informatics, No. 16, Elsevier, 2002, pp. 277-289.

CHOMSKY, N. (1980). *Studies on Semantics in Generative Grammar*. The Hague: Mouton, 1972. Reprint. Berlin and New York, 1980.

DAHLBÄCK, N. et JÖNSSON, A. (1997). *Integrating Domain Specific Focusing in Dialogue Models*. Proceedings of EuroSpeech-97, Rhodos, Greece.

DAVIS, R. et SMITH, R.G. (1983). *Negociation as a Metaphor for Distributed Problem Solving*. Artificial Intelligence, V.20(1), p. 63-100, 1983.

d'INVERNO, M et LUCK, M. (2004). *Understanding Agent Systems*. Springer, 2nd edition.

DUTOIT, T., COUVREUR, L., RIS, C., MALFRERE, F. et PAGEL, V. (2002). *Synthèse Vocale et Reconnaisance de la Parole : Droites Gauches et Mondes Parallèles*, Actes du 6è Congrès Français d'Acoustique, Lille, 8-11 avril 2002.

DZIKOVSKA, M. O., SWIFT, M. D. et ALLEN, J. F. (2003) *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.

ENEMBRECK, F. (2003). *Contribution à la conception d'agents assistants personnels adaptatifs*. Thèse de doctorat, Université de Technologie de Compiègne, France, décembre, 2003.

ENEMBRECK, F. et BARTHÈS, J-P A. (2003). *Improving CSCW with Personnel Assistant Agents*. Journal of Integrated Design & Process Science, B. Kramer, J.P. Tsai (eds.), IOS Press, ISSN: 1092-0617.

ERIKSSON, A. (2001). *Domain Knowledge Management in Information-providing Dialogue Systems*. Thèse de doctorat, School of Engineering à Linköping University, Suède, 2001.

ERIKSSON, A. (2003). *Design of Ontologies for Dialogue Interaction and Information Extraction*. Proceedings of International Joint Conference on Artificial Intelligence IJCAI 2003, Acapulco, Mexico.

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

FERRET, O. (2004). *Découvrir des sens de mots à partir d'un réseau de cooccurrences lexicales*. TALN 2004, Fès, Maroc.

FERRET, O., GRAU, B., HURRAULT-PLANTET, M., ILLOUZ, G. et JACQUEMIM, C. (2001). *Utilisation des entités nommées et des variantes terminologiques dans un système de question-réponse*. Actes de TALN 2001, Tours, France.

FININ, T., McKAY, D., FRITZON R. et McENTIRE R. (1994). *The KQML Knowledge Exchange Protocol*. Third International Conference on Information and Knowledge Management. National Institute of Standards and Technology, Gaithersburg, Maryland.

FIPA (2001). *FIPA Personal Assistant Specification*. Document XC00083B disponible en ligne http://www.fipa.org/specs/fipa00038/XC00038B.pdf.

FIPA (2003). *FIPA Abstract Architecture Specification*. Document SC00001L (Standard 3/12/2003), 2003 disponible en ligne http://www.fipa.org/specs/fipa00001/SC00001L.pdf.

FIPA (2003A). *FIPA Communicative Act Libray Specification*. Document SC00037J (Standard 3/12/2003), 2003, disponible en ligne http://www.fipa.org/specs/fipa00037/SC00037J.pdf.

FLAMMIA, G. (1998) *Discourse Segmentation of Spoken Dialogue: An Empirical Approach*. Thèse de MIT, 1998, 152p.

FRIED, D., WILKINS, D. et GROIS, E. (2003). *The Gerona Knowledge Representation Language and Its Support for Spoken Dialogue Tutoring of Crisis Decison Making Skills*. Proceedings of International Joint Conference on Artificial Intelligence IJCAI 2003, Acapulco, Mexico.

GAIFFE, B., LANDRAGIN, F. et GUIGNARD, M. (2004). *Le dialogue naturel comme un service dans contexte multi-applicatif*. Actes de la journée Agents et Langue, Paris, France, 2004, pp. 57-66.

GENNARI, J., MUSEN, M. A., FERGERSON, R. W. GROSSO, W. E., CRUBEZY, M., ERIKSSON, H., NOY, N. F. and TU, S. W. (2002). *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.

GIRAULT, F. (1999). *Footux team description : A hybrid recursive based agent architecture*. In Proceedings of the IJCAI Third Robocup Workshop.

GRAESSER, A., VENTURA, M., JACKSON, G. T., MUELLER, J., HU, X. et PERSON, N. (2003). *The Impact of Conversational Navigational Guides on the Learning, Use, and Perceptions of Users of a Web Site*. Agent-Mediated Knowledge Management (AMKM-03), Stanford, USA.

GRINBERG, D., LAFFERTY, J. et SLEATOR, D. (1995). *A robust parsing algorithm for link grammars*. Carnegie Mellon University Computer Science technical report CMU-CS-95-125, and Proceedings of the Fourth International Workshop on Parsing Technologies, Prague, September, 1995.

GRUBER, T. (1993). *A Translation Approach to Portable Ontology Specification*. Knowledge Acquisition 5.

GUARINO, N. (1998). *Formal Ontology in Information Systems*. Proceedings of FOIS'98, IOS Press, Italy, pp. 3-15.

HARABAGIU S, MILLER G.A., MOLDOVAN D (1999). *WordNet 2 - A Morphologically and Semantically Enhanced Resource*. Actes de SIGLEX'99, pp. 1-8.

HE, Y. et YOUNG, S. (2005). *Semantic processing using the Hidden Vector State model*. Computer Speech and Language, Vol. 19, No. 1, 2005, pp. 85-106.

HELLWIG, P. (1999). *Natural Language Parsers. A "Course in Cooking.* Accessible à partir de: http://www.cl.uni-heidelberg.de/~hellwig/tutorial.html.

HORVITZ, E. (1999) *Principles of Mixed-Initiative User Interfaces*. Proceedings of CHI '99, ACM SIGCHI Conference on Human Factors in Computing Systems, Pittsburgh, PA, May 1999. ACM Press. pp 159-166.

HOWDEN, N., RÖNNQUIST, R., HODGSON, A. et LUCAS, A. (2001). *JACK Intelligent Agents: Summary of an Agent Infrastructure*. Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS, 2001.

HYLTON, J. (1996). *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.

HYÖTYNIEMI, H., HEMANUS, J. et LANTZ, V. (2004). *Exchanging emotions – SOM approach*. Proceedings of ECAI - European Conference on Artificial Intelligence, Valence - Espagne.

JENNINGS, N.R., SYCARA K. et WOOLDRIDGE, M (1998). *A Roadmap of Agent Research and Development*. Journal of Autonomous Agents and Multi-Agent Systems, Kluwer Academic Publishers, vol. 1, no1, p. 7-18, 1998.

JOHNSON, W., SHAW, E., MARSHALL, A. et LABORE, C. (2003) *Evolution of User Interaction: the Case of Agent Adele*. Proceedings of Intelligent User Interfaces (IUI-03), ACM Press, 2003, pp.93-100.

JURAFSKI, D. et MARTIN, J. (2000). *Speech and Language processing: an introduction to natural language processing*. Computational Linguistics and Speech Recognition. Prentice Hall, 2000.

KIM, J., SPRARAGEN, M. et GIL, Y. (2004). *An Intelligent Assistant for Interactive Workflow Composition*. Proceedings of Intelligent User Interfaces 2004 (IUI-04), Madera, Portugal.

KOLZER, A. (1999). *Universal Dialogue Specification for Conversational Systems.* Proceedings of IJCAI 99 – Workshop on Knowledge and Reasoning in Practical Dialogue Systems, 1999.

KRAMER, N. C., TIETZ, B. et BENTE, G. (2003). *Effects of Embodied Interface Agents and their Gestural Activity*. In R. Aylett, D. Ballin, T. Rist & J. Rickel (eds.), 4th International Working Conference on Intelligent Virtual Agents. Hamburg.

KUMAR A. et ROMARY, L. (2002). *A Comprehensive Framework for Multimodal Meaning Representation*. In Proceedings of the International Workshop on computational Semantics (IWCS-5), Tilburg, Netherlands.

LANGLEY, C., LAVIE, A., LEVIN, L., WALLACE, D., GATES, D. et PETERSON, K. (2002).

Spoken Language Parsing Using Phrase-Level Grammars and Trainable Classifiers. Proceedings of the Workshop on Speech-to-Speech Translation: Algorithms and Systems, Philadelphia, July 2002, pp. 15-22.

LARSON, J. A. (2003). *Commonsense Guidelines for Developing Multimodal User Interfaces*. Disponible sur http://www.larson-tech.com.

LEBARBE, T. (2002). *Hiérarchie Inclusive des Unités Linguistiques en Analyse Syntaxique Coopérative*. Thèse de doctorat, Université de Caen, 2002.

LEMEUNIER, T. (2000). *L'intentionnalité communicative dans le dialogue homme-machine en langue naturelle*. Thèse l'Université du Maine (235 pages).

LENAT, D. B. (1995). *Cyc: A Large-Scale Investment in Knowledge Infrastructure*. Communications of the ACM 38, no. 11, November, 1995.

MAES, P. (1994). *Agents that Reduce Work and Information Overload*. Communications of ACM, Vol. 37, No.7, ACM Press, pp. 30-40.

MANDIAU, R., GRISLIN-LE STRUGEON, E., PÉNINOU A. (2002). Organisation et applications des SMA. Hermes, ISBN : 2-7462-0439-8.

MARTIN, J.-C., RÉTY, J.-H. et BENSIMON, N. (2002). *Multimodal and Adaptative Pedagogical Resources*. Electronic proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC'2002), Las Palmas, Spain.

MARTIN, J.-C., BUISINE, S. et ABRILIAN, S. (2004). *2D Gestural and Multimodal Behavior of Users Interacting with Embodied Agents*. Proceedings of the workshop Embodied Conversational Agents: Balanced Perception and Action, Edited by Catherine Pelachaud, Zsófia Ruttkay & Kris Thorisson. Held during the Third International Joint Conference on Autonomous Agents & Multi Agent Systems, New York, USA, 19-23 July 2004, 34-41.

MEYER, J. L. (2004). *Reasoning about emotional agents*. Proceedings of ECAI - European Conference on Artificial Intelligence, Valence - Espagne.

MI, X. et CHEN, X. (2005). *Agent-¨Based Interaction Model for Collaborative Virtual Environments*. Proceedings of CSCWD in Design. (pp. 401-404). Coventry.

MICROSOFT. (1999). *Microsoft Agent: Software Development Kit.* Microsoft Press, Redmond Washington.

MIDDLETON, S. E., De ROURE, D. C. et SHADBOLT, N. R. (2001). *Capturing Knowledge of User Preferences: Ontologies in Recommender Systems*. Proceedings of the International Conference on Knowledge Capture (KCAP'01), ACM Press, pp. 100-107.

MILWARD, D. et BEVERIDE, M. (2003). *Ontology-based dialogue systems*. Proceedings of International Joint Conference on Artificial Intelligence IJCAI 2003, Acapulco, Mexico.

MINSKY, M. (1975). *The Psychology of Computer Vision*. P. Winston (Ed.), McGraw-Hill.

MOULIN, C., SBODIO, M., BARTHES, J-P. A. (2005). *eGovernment Application Interoperability*. Proceedings of the Joint INTEROP-ESA/eGov Interop 2005 Conference, Genève, Suisse.

NICHOLS, J et MYERS, B. A. (2005). *Generating Consistent Interfaces for Appliances*. Second Workshop on Multi-User and Ubiquitous User Interfaces (MU3I) at Intelligent User Interfaces 2005. January 9, 2005.

NIEKRASZ, J., PURVER, M., DOWDING, J. et PETERS, S. (2005). *Ontology-based discourse understanding for a persistent meeting assistant*. In Proceedings of the AAAI Spring Symposium on Persistent Assistants: Living and Working with AI, Stanford, California.

NGUYEN, A. et WOBCKE, W. (2005). *An Agent-Based Approach to Dialogue Management in Personal Assistants*. Proceedings of IUI 2005, San Diego, USA, pp. 137-144.

NONAKA, I., TOYAMA, R. et KONNO, N. (2000). *SECI, Ba and Leadership: a Unified Model of Dynamic Knowledge Creation*. In Long Range Planning, Elselvier Science, n. 33, p. 5-34, 2000.

NOY, N. F. et HAFNER, C. D. (1997). *The state of the art in ontology design, a survey and comparative review*. AI Magazine, 18(3):53–74.

OTTENS, K., AUSSENAC-GILLES, N., GLEIZES, M.-P. et GLIZE, P. (2004). *Système Multi-Agents pour l'extraction d'ontologies à partir de textes: revue de questions.* Actes de AGENTAL, Paris, 2004.

OZONE (2004). *Offering an Open and Optimal Roadmap Towards Consumer Oriented Ambient Intelligence*. http://www.extra.research.philips.com/euprojects/ozone, IST-2000-29487 European Project, 2001-2004.

PANAGET, F. (1998) *Le générateur de langage naturel de l'agent dialoguant ARTIMIS*. Traitement Automatique des Langues 39(2), pp. 107-126.

PARAISO, E. C. et BARTHES, J.-P. A. (2004). *Une interface conversationnelle pour les agents assistants appliqués à des activités professionnelles*. In: Proceedings of 16th Conférence Francophone sur l'Interaction Homme-Machine, Namur, Belgique, 2004, pp. 243-246.

PARAISO, E. C., BARTHES, J.-P. A et TACLA, C. A. (2004). *A Speech Architecture for Personal Assistant in a Knowledge Management Context.* Proceedings of ECAI - European Conference on Artificial Intelligence, Valence - Spain, 2004, pp. 971-972.

PARAISO, E. C. et BARTHES, J.-P. A. (2005). *SpeechPA: An Ontology-Based Speech Interface for Personal Assistants*. Proceedings of the 2005 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, Compiègne, France.

PARAISO, E. C. et BARTHES, J.-P. A. (2005a). *A Voice-Enabled Assistant in a Multi-Agent System for e-Government Services.* Proceedings of F.F. Ramos et al. (Eds.): ISSADS 2005. (Guadalajara, Mexico, 2005). Lecture Notes in Computer Science - 3563, pp. 495 - 503.

POPESCU, A. M., ETZIONI, O. et KAUTZ, H. (2003). *Towards a theory of natural language interfaces to databases*. In: Proceedings of Intelligent User Interfaces (IUI-03), ACM Press, 2003, pp. 149-157.

PRIVAT, R., VIGOUROUX, N. et TRUILLET, P. (2000). *Utilisabilité de l'interaction vocale pour l'accès aux systèmes interactifs par les personnes âgées*. Actes de la deuxième conférence pour l'essor des technologies d'assistance – HANDICAP 2000, Juin, 2000.

PURVER, M., NIEKRASZ, J. et PETERS, S. (2005). *Ontology-based multi-party meeting understanding*. Abstract accepted for the CHI 2005 workshop The Virtuality Continuum Revisited, April 2005.

QUESADA, J. F., GARCIA, F., SENA, E., BERNAL, J. et AMORES, G. (2001). *Dialogue Managements in a Home Machine Environment: Linguistic Components Over an Agent Architecture*. SEPLN, 2001, pp. 89-98.

QUILLIAN, M.R. (1968). *Semantic Memory in Semantic Information Processing*. M.I.T. Press.

RAMOS, M. (2000). *Structuration et évolution conceptuelles d'un agent assistant personnel dans les domaines techniques*. Thèse de l'UTC, Compiègne, 215p.

RAO A. S. et GEORGEFF M. P. (1995). *BDI Agents: From Theory to Practice*. Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, EUA, June, 1995.

REITER, E. et DALE, R. (2000). *Building Natural-Language Generation Systems*. Cambridge University Press.

RICKEL, J., LESH, N., RICH, C., SIDNER, C., et GERTNER, A. (2001). *Building a Bridge between Intelligent Tutoring and Collaborative Dialogue Systems*. Proceedings of the 10th Int. Conf. on Artificial Intelligence in Education, San Antonio, TX, pp. 592-594.

ROUILLARD, J. (2002). *L'hyperdialogue*. Interaction home-machine et recherché d'information. Paris, pp.229-259.

RUSSEL, S. et NORVIG, P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice Hall.

SADEK, D. (1996). *Le dialogue homme-machine : de l'ergonomie des interfaces à l'agent intelligent dialoguant*. Nouvelles interfaces Homme-machine, Observatoire Français des Techniques Avancées (ed.), Série Arago n° 18, Lavoisier, Paris, pp. 277-321.

SALTON, G. (1989). *Automatic Text Processing: The Transformations, Analysis and Retrieval of Information by Computer*. Addison-Wesley.

SANSONNET, J.-P. (2004). *Présentation du groupe de travail sur les Agents Conversationnels Animés*. Paris, 3 march 2004.

SCALABRIN, E. et BARTHÈS, J-P. (1993). *OSACA: une architecture ouverte d'agents cognitifs independents*. Actes de la 2ème Journée Nationale du PRC-IA sur les systèmes multi-agents, Montpellier, France.

SCALABRIN, E. E. (1996). *Conception et Réalisation d'Environnement de Développement de Systèmes d'Agents Cognitifs*. Thèse de l'UTC, Compiègne, 181p, 1996.

SCHANK, R. C. et ABELSON, R. P. (1977). *Scripts, Plans, Goals and Understanding*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1977.

SADEK, D. (1999). *Design considerations on dialogue systems: from theory to technology. The case of Artimis*. Proceedings of IDS'99, Kloster Irsee, Germany, pp.173-187.

SADEK, D., BRETIER, P. et PANAGET, F. (1997). *Artimis: Natural dialogue meets rational agency*. Proceedings of IJCAI'97 (International Joint Conference on Artificial Intelligence), Nagoya, Japan, 1030-1035, 1997.

SHARMA, C. et KUNINS, J. (2001). *Voice XML.* Wiley.

SEARLE, J. R. (1975). *A Taxonomy of Illocutionary Acts*, In Proceedings of Language, Mind and Knowledge, Vol. 7, University of Minnesota Press, 1975, pp. 344-369.

SENEFF, S, HURLEY, E., LAU, R., PAO, C., SCHIMID P. et ZUE, V. (1998). Galaxy-II: A Reference Architecture for Conversational System Development. Proc. ICSLP 98, Sydney, Australia, November 1998.

SENEFF, S. et POLIFRONI, J. (2000). Dialogue Management in the Mercury Flight Reservation System, Satellite Dialog Workshop ANLP-NAACL, Seattle, April 2000.

SHEN, W. et BARTHÈS, J-P. A. (1996). *An experimental multi-agent environment for engineering design*. International Journal of Cooperative Information Systems, vol. 2&3(5) (World Scientific Publishing Company), p. 131-151.

SINGH, M. P. et HUHNS, M. N. (2005). *Service-Oriented Computing: Semantics, Processes, Agents*. John Wiley & Sons, Ltd., 2005.

SLEATOR, D. et TEMPERLEY, D. (1993). *Parsing English with a Link Grammar*. Third International Workshop on Parsing Technologies.

SOWA, J.F. (1984). *Conceptual Structures Information Processing in Mind and Machine*. Addison-Wesley.

SYCARA, K. (1998). *Multi-Agent Systems*. AI Magazine, vol. 19, no. 2, pp. 79-92.

TACLA, C. A. (2003). *De l'utilité des systèmes multi-agents pour l'acquisition des connaissances au fil de l'eau*. Thèse de l'UTC, Compiègne, 234p.

TADDEI, L., CONSTANTINI, E. et LAVIE, A. (2002). *The NESPOLE! Multimodal Interface for Cross-lingual Communication - Experience and Lessons Learned*. Proceedings of ICMI 2002 International Conference on MULTIMODAL INTERFACES, Pittsburgh, USA, 14-16 October 2002.

TSAI, M.-J. (2005). *The VoiceXML Dialog System for the E-Commerce Ordering Service*. The 9th International Conference on CSCW in Design, IEEE, 2005, pp. 95-100.

TUTIN, A. (2000). *INTEX pour l'annotation semi-automatique d'un corpus d'anaphores*. Linguisticae Investigationes, Volume 22, Number 2, octobre 2000, pp. 173-189(17).

USCHOLD, M. (1998). *Knowledge Level Modelling: Concepts and Terminology*. Knowledge Engineering Review, 13(1).

VOSSEN, P. (1998). *EuroWordNet: A Multilingual Database with Lexical Semantic Networks*. Kluwer Academic Publishers, Dordrecht.

VU-MINH, Q, BESACIER, L., BLANCHON, H. et BIGI, B. (2004). *Modèle de langage sémantique pour la reconnaissance automatique de parole dans un contexte de traduction*. Actes de la conférence TALN.

WALKER, M., CAHN, J., et WHITTAKER, S.J. (1997a). *Improving Linguistic Style: Social and Affective Bases for Agent Personality*. Proceedings of Autonomous Agents '97. (pp. 96-105). Marina del Rey.

WALKER, M. A., LITMAN, D. J., KAMM, C. A. et ABELLA, A. (1997). *PARADISE: A Framework for Evaluating Spoken Dialogue Agents*. Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics.

WARD, W. et PELLOM, B. (1999). *The CU Communicator System*. IEEE Workshop on Automatic Speech Recognition and Understanding, Keystone Colorado, December, 1999.

YANKELOVICH, N. (2001). *Using Natural Dialogs as the Basis for Speech Interface Design.* Chapitre du livre, Automated Spoken Dialog Systems, edited by Susann Luperfoy.

YATES, A, ETZIONI, O. et WELD, D. (2003). *A Reliable Natural Language Interface to Household Appliances*. Proceedings of Intelligent User Interfaces (IUI-03), ACM Press, 2003, pp. 189-196.

ZOE, V. W. et GLASS, J. R. (2000). *Conversational Interfaces : Advances and Challenges*. Proceedings of The IEEE, Vol. 88, pp. 1166-1180.

Annexe I

Réalisation d'applications vocales avec *Voice Tools*

Dans cette annexe, nous montrons à travers quelques exemples simples, la réalisation d'applications vocales avec l'outil Voice Tools. Nous nous limitons à des applications « autonomes » (ou *stand-alone*) pour le système d'exploitation Windows. Nous présentons à la fin de cette annexe, un serveur vocal pour permettre l'intégration de la parole à des applications du type commande et contrôle codées en Allegro Common Lisp.

1. Développement d'une application vocale sur le système d'exploitation Windows

La réalisation d'une application vocale passe impérativement par l'utilisation d'un moteur de reconnaissance et de synthèse de la parole. Plusieurs de ces moteurs sont commercialement disponibles : Via Voice d'IBM et Dragon NaturallySpeaking de Scansoft en sont deux exemples. Dans notre travail, nous avons utilisé initialement le moteur Microsoft English Engine 5.1. Tous les moteurs sont contrôlés directement par le système d'exploitation. Ces moteurs fournissent plusieurs fonctionnalités liées à leur usage : la reconnaissance et la synthèse en sont, bien entendu, les principales. Une API (*Application Programming Interface*) a été développée par Microsoft pour permettre l'accès à ces fonctionnalités : la SAPI (*Speech Application Programming Interface*). La Figure 93 donne une vision générale de comment une application accède à ces fonctionnalités.

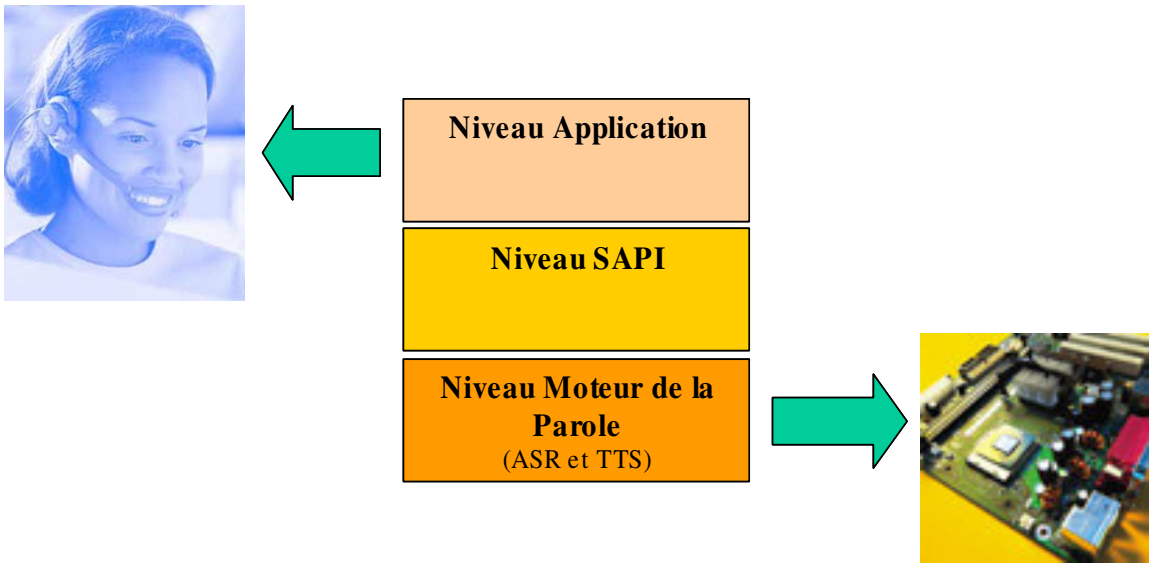


Figure 93 : Le niveau Application par rapport aux moteurs.

Tandis que l'usage de toutes ces fonctionnalités est relativement complexe, des contrôles ActiveX/COM ont été développés par différentes sociétés, encapsulant la complexité, et permettant d'interfacer les applications à la SAPI. Du point de vue de l'application, la manipulation des moteurs de reconnaissance et de synthèse est beaucoup plus simple (voir Figure 94).

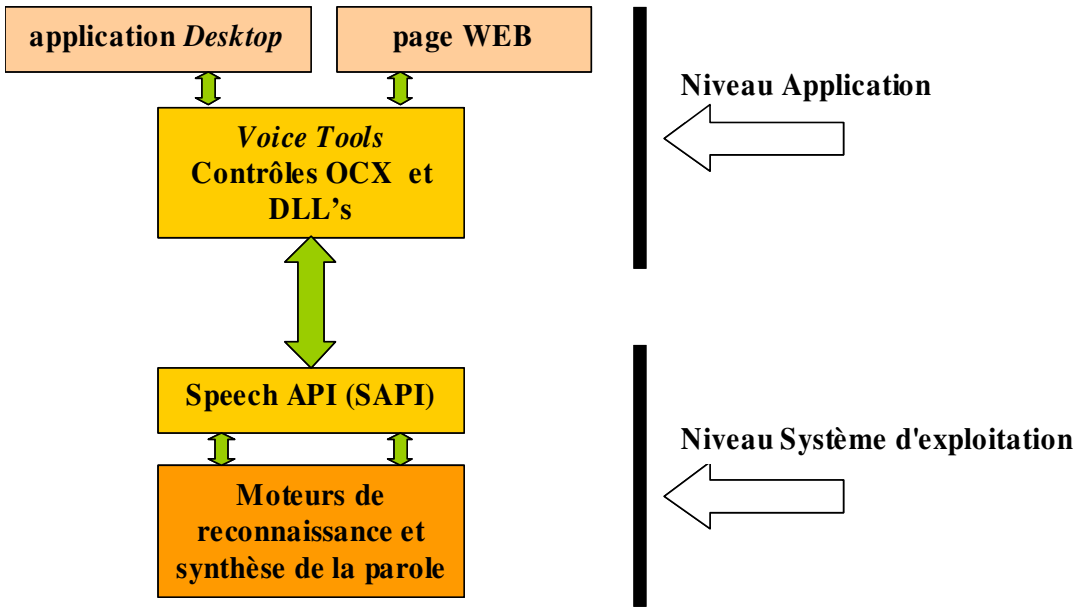


Figure 94 : Les niveaux Application et Système d'exploitation.

Ces ensembles de contrôles, comme l'outil Voice Tools de la société Wizzard Software ou Dragon NaturallySpeaking SDK de la société Scansoft, peuvent être utilisés pour le

développement d'applications locales ou pour le WEB, à travers les langages de programmation qui acceptent les contrôles COM.

Nous avons développé quelques applications du type commande et contrôle pour démontrer la façon de les utiliser.

1.1. Quelques applications Commande et Contrôle

Une application commande et contrôle est caractérisée par la « vocalisation » des commandes. Ci-après, nous avons deux exemples simples : une application Météo et une application de *Home banking*.

L'application Météo (voir fenêtre sur la Figure 95) a quatre options : *Select a City* (sélection d'une ville), *Satellite Photos* (photos de satellite), *Conversion* (conversion) et *Exit* (sortie).

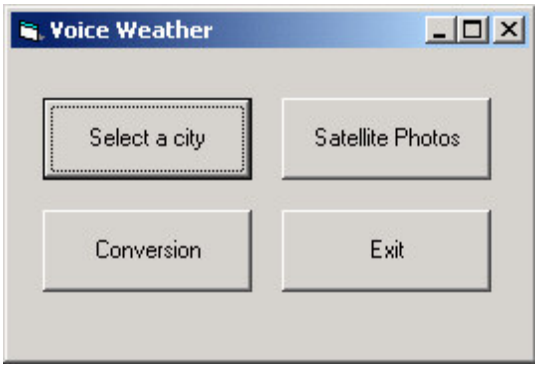


Figure 95 : Fenêtre de l'application Météo.

Les quatre options sont des boutons qui déclanchent les commandes désirées. Dans ce cas, un bouton sera actionné si l'utilisateur appuie dessus ou si l'utilisateur dit le titre du bouton. Dès que le moteur de reconnaissance reconnaît l'énoncé, un contrôle COM donnera suite au traitement. Tout ce qui est saisi par le micro est transféré au moteur de reconnaissance qui fait la conversion et remonte l'information à l'application à travers un contrôle COM. À la charge de l'application reste donc l'analyse du contenu du texte reçu.

Nous listons ci-dessous, les lignes de commandes en Visual Basic capables de traiter ces quatre boutons. Notons que la variable `TextRecognizedByTheSpeechEngine`, contient le texte reconnu et traduit par le moteur de reconnaissance. Il faut rappeler que cette application commande et contrôle, n'utilise pas les mêmes algorithmes de traitement syntaxique et sémantique de l'interface conversationnelle, car les commandes sont figées.

```
Select Case TextRecognizedByTheSpeechEngine
        Case "Select a city"
            Call Exec_SelectCity()
        Case "Satellite Photos"
            Call Exec_Satellite()
        Case "Conversion"
            Call Exec_Conversion()
        Case "Exit"
            Call Exit()
        Case Else
            TextToSpeech = "Invalid Command"
            Call Say_It(TextToSpeech)
End Select
```

Pour l'utilisateur de l'application, il n'y a pas de différences entre appuyer sur un bouton ou dire son titre.

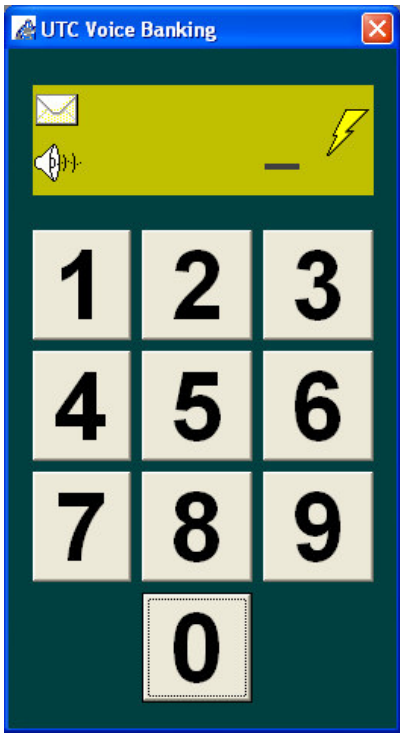


Figure 96 : Le simulateur d'un *Home Banking*.

La deuxième application est un simulateur d'un Home banking accessible par téléphone (voir fenêtre sur la Figure 96). Puisqu'il n'y a aucune fenêtre graphique disponible, le développeur doit mettre en place un mécanisme un peu plus complexe (un dialogue du type état

fini). La stratégie de dialogue choisie est directive, où l'initiative du dialogue part forcément du serveur, qui est responsable du but à atteindre.

Le dialogue suivant illustre le démarrage du système :

SYS (1): Welcome to Voice Bank. Please enter or say your account number.

USR (2): Five, six, one, one, two, eight.

SYS (3): Please, enter your PIN number?

USR (4): « L'utilisateur va taper son mot de passe. »

SYS (5): Please say "Account balance" or "Transfer funds" or "Preferences" or "Exit" to quit the system.

…

On note clairement que le système s'occupe de guider l'utilisateur au cours du dialogue. Chaque entrée doit être traitée en suivant les états préalablement prévus. L'application utilise cette fois la synthèse de la parole à partir de textes associés à chaque état (énoncés 1, 3 et 5). Pour cela, un autre contrôle COM sera utilisé.

1.2. AllegroCL et le serveur vocal

La version Allegro Commom Lisp Release 6.1, disponible au sein de notre laboratoire et que nous avons utilisée pour l'implémentation des agents, ne manipule pas les contrôles AcitiveX/COM. Si nous voulons développer une application du type commande et contrôle en lisp, une solution est envisageable : l'utilisation d'un serveur vocal. Nous avons réalisé un serveur vocal (en Visual Basic) et un ensemble de fonctions lisp pour permettre le déploiement d'applications du type commande et contrôle.

1.2.1 Le serveur vocal

Le serveur vocal est une application qui doit tourner sur la même machine que l'application lisp cliente (fenêtre du serveur sur la Figure 97). Chaque application cliente doit enregistrer auprès du serveur la liste de commandes qui seront vocalisées. Le serveur de son côté, informera le client si l'utilisateur prononce une de commandes attendues. La version courante du serveur est capable de traiter les boutons, les menus et les listes. Une autre fonctionnalité du serveur est la synthèse des textes provenant du client.

Figure 97 : Le serveur vocal.

Le serveur doit être chargé avant le client. Il « veille » constamment sur la sortie du moteur de reconnaissance, travaillant indépendamment.

Un format de message simple a été défini pour permettre l'enregistrement de commandes :

```
<string> ::= ASCII
<message> ::= <component type>&<caption>
<component type> ::= "button" | "menu" | "list" | "tosay" | "text"
<caption> ::= <string>
```

Pour enregistrer un bouton intitulé *Open File*, le client doit envoyer le message :

"button&Open File"

Les communications se font par le port 9999 (entrée du serveur) et 9998 (entrée du client).

1.2.2 Le client lisp

Le client lisp doit incorporer un package de fonctions, codées en lisp, conçues pour gérer les communications entre les deux applications. Après cela, tout est prêt pour l'utilisation du serveur.

D'abord, le client envoie une requête d'enregistrement, appelant la fonction *init-voice*. Ensuite, le client peut enregistrer tous les composants qu'il veut « vocaliser », appelant l'une des fonctions : *add-voice-to-button*, *add-voice-to-menu* ou *add-voice-to-list*.

```
(add-voice-to-button caption function-to-execute list-of-args)
(add-voice-to-menu caption function-to-execute list-of-args)
(add-voice-to-list caption function-to-execute list-of-args)
```

où:

caption est le titre du composant ;

function-to-execute est la fonction lisp qui va être déclanchée quand le titre du composant est reconnu par le serveur;

list-of-args est une liste d'arguments qui sera envoyée à la fonction *function-to-execute*.

La fonction *say-it* est utilisée par le client pour envoyer un texte qui sera lu à l'utilisateur.

Pour terminer la connexion, le client appelle la fonction *finish-voice*.

Le code ci-dessous, donne l'exemple du processus d'enregistrement des composants d'une application (voir Figure 98) :

```
(init-voice)
(add-voice-to-button "register" 'exec-register nil)
(add-voice-to-button "components" 'exec-components nil)
(add-voice-to-button "send color" 'exec-send nil)
(add-voice-to-button "new color" 'exec-new nil)
(add-voice-to-button "close" 'exec-close nil)
(add-voice-to-list "blue" 'exec-color 'blue)
(add-voice-to-list "yellow" 'exec-color 'yellow)
```

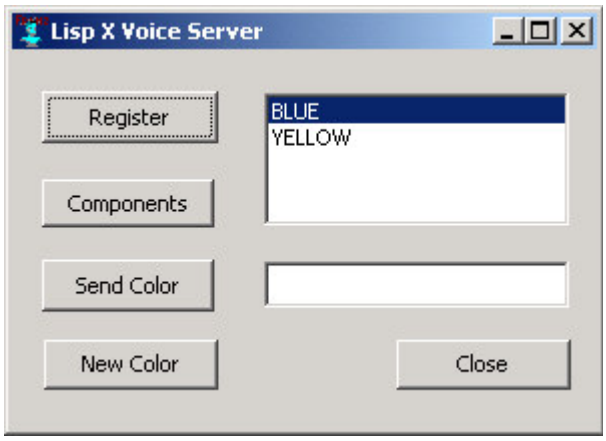


Figure 98 : La fenêtre du client lisp.

Le client peut demander la synthèse d'un texte :

```
(say-it "This is a test !")
```

Pour terminer la connexion :

```
(finish-voice)
```

Dans l'exemple précédent, si l'utilisateur prononce les mots « *new color* », la fonction lisp *exec-new* sera aussitôt appelée. La même chose se passerait, si l'utilisateur appuyait sur le bouton « *new color* ». On note que pour le développeur du client lisp, le travail se résume à enregistrer les composants.

Annexe II

Le langage de contenu de gestion des agents

Dans cette annexe, nous présentons le langage de contenu utilisé au cours des échanges entre les agents travaillant dans le modèle distribué d'identification et d'exécution de tâches.

1. Le format des messages

Le langage est composé de plusieurs messages de contrôle et de distribution de tâches. Ces messages sont échangés entre les agents soit en *broadcast* soit en connexion directe.

1.1. Le message INFO

Description : message utilisé par un agent de service pour envoyer un contenu informatif à l'agent assistant.

Format : (INFO *content*)

Paramètres :

Non	**Description**
Content	Message textuel informatif

Forme d'envoi : directe.

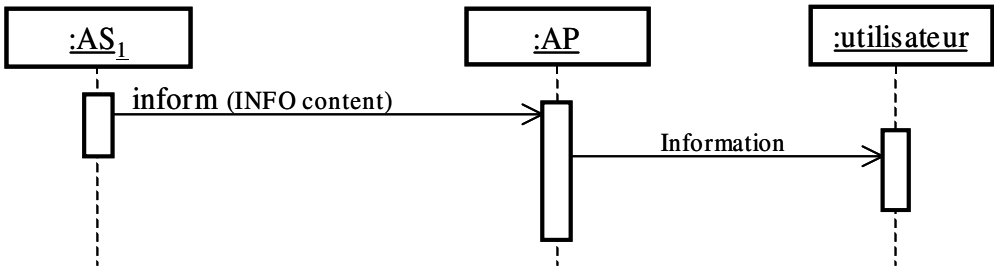
Priorité donnée par l'agent assistant lors de la réception : basse.

Retour attendu: aucun.

Exemple :

```
(INFO "You have a new message.")
```

Diagramme de séquence :



1.2. Le message REQUEST-INFO

Description : message qui décrit une requête posée à l'utilisateur.

Format : (REQUEST-*INFO question-description msg-id*)

Paramètres :

Non	Description
question-description	Une structure du type question-description
msg-id	Identificateur de contrôle

Forme d'envoi : directe.

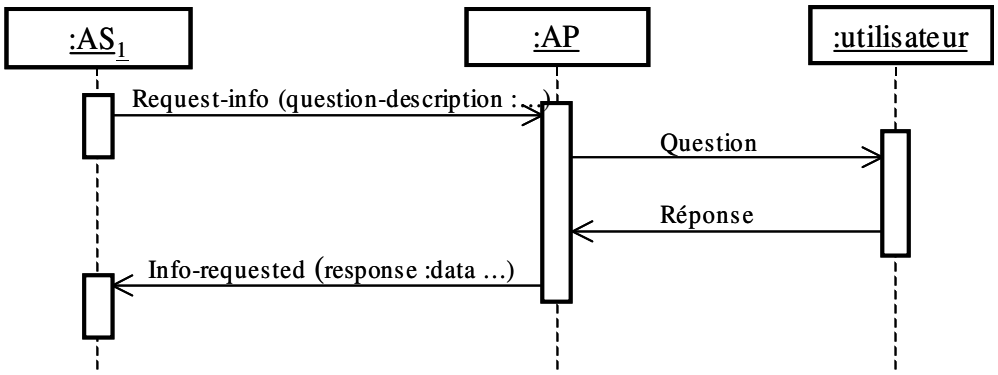
Priorité donnée par l'agent assistant lors de la réception : haute.

Retour attendu :INFO-REQUESTED

Exemple :

```
(REQUEST-INFO ("
<prm>
 <name>REPORT</name>
          <synset>nil</synset>
          <question>A new report on Agents is available. Do you
      want to read it?</question>
 <type>yn</type>
 <required>true</required>
 <input_source>vocal</input_source>
</prm>")
1)
```

Diagramme de séquence :



1.3. Le message INFO-REQUESTED

Description : message de retour à une requête d'information.

Format : (INFO-REQUESTED *msg-id content*)

Paramètres :

Non	Description
msg-id	Identificateur de contrôle reçu lors de l'envoi du message REQUEST-INFO
Content	Liste contenant la réponse

Forme d'envoi : directe.

Priorité donnée par l'agent assistant lors de la réception : non applicable.

Retour attendu : aucun.

Exemple :

```
(INFO-REQUESTED 1 (REPORT "yes"))
```

Diagramme de séquence : voir diagramme du message REQUEST-INFO.

1.4. Le message REQUEST-HELP-INFO

Description : l'agent assistant envoie ce message aux agents pour leur demander une description de leurs compétences (services).

Format : (REQUEST-HELP-INFO *msg-id*)

Paramètres :

Non	Description
msg-id	Identificateur de contrôle

Forme d'envoi : *broadcast*.

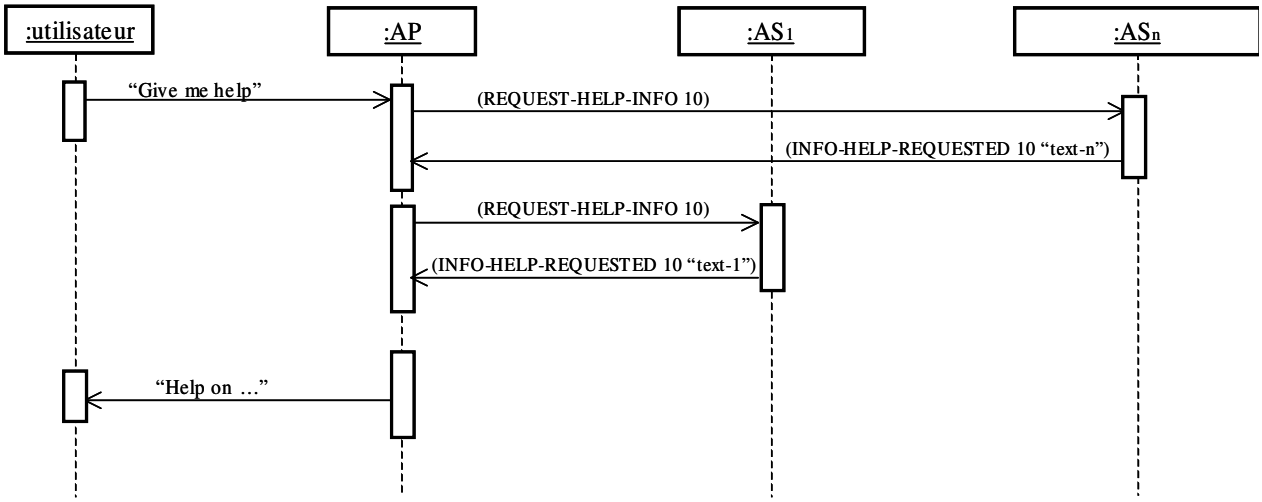
Priorité donnée par l'agent assistant lors de la réception : non applicable.

Retour attendu : INFO-HELP-REQUESTED.

Exemple :

```
(REQUEST-HELP-INFO 10)
```

Diagramme de séquence :



1.5. Le message INFO-HELP-REQUESTED

Description : message de retour à un REQUEST-HELP-INFO.

Format : (INFO-HELP-REQUESTED *msg-id content*)

Paramètres :

Non	Description
msg-id	Identificateur de contrôle reçu lors de l'envoi du message REQUEST-HELP-INFO
Content	Texte descriptif

Forme d'envoi : directe.

Priorité donnée par l'agent assistant lors de la réception : haute.

Retour attendu : aucun.

Exemple :

```
(INFO-HELP-REQUESTED 10 "To manage the email account")
```

Diagramme de séquence : voir diagramme du message REQUEST-HELP-INFO.

1.6. Le message TASK-SEARCH

Description : message envoyé par l'agent assistant pour chercher un agent de service capable d'accomplir une tâche. Comme retour un agent de service peut envoyer un message de spécification de tâche (remplissage de paramètre) ou le résultat d'exécution si l'agent de service n'a pas besoin d'informations complémentaires provenant de l'utilisateur. L'agent de service pourra encore répondre à travers un message du type TASK-CONFIRMATION s'il a bien exécuté la tâche, mais celle-ci n'a pas de résultat à afficher à l'utilisateur.

Format : (TASK-SEARCH *script msg-id*)

Paramètres :

Non	Description
script	Liste du format d'une requête formelle
msg-id	Identificateur de contrôle

Forme d'envoi : *broadcast*.

Priorité donnée par l'agent assistant lors de la réception : non applicable

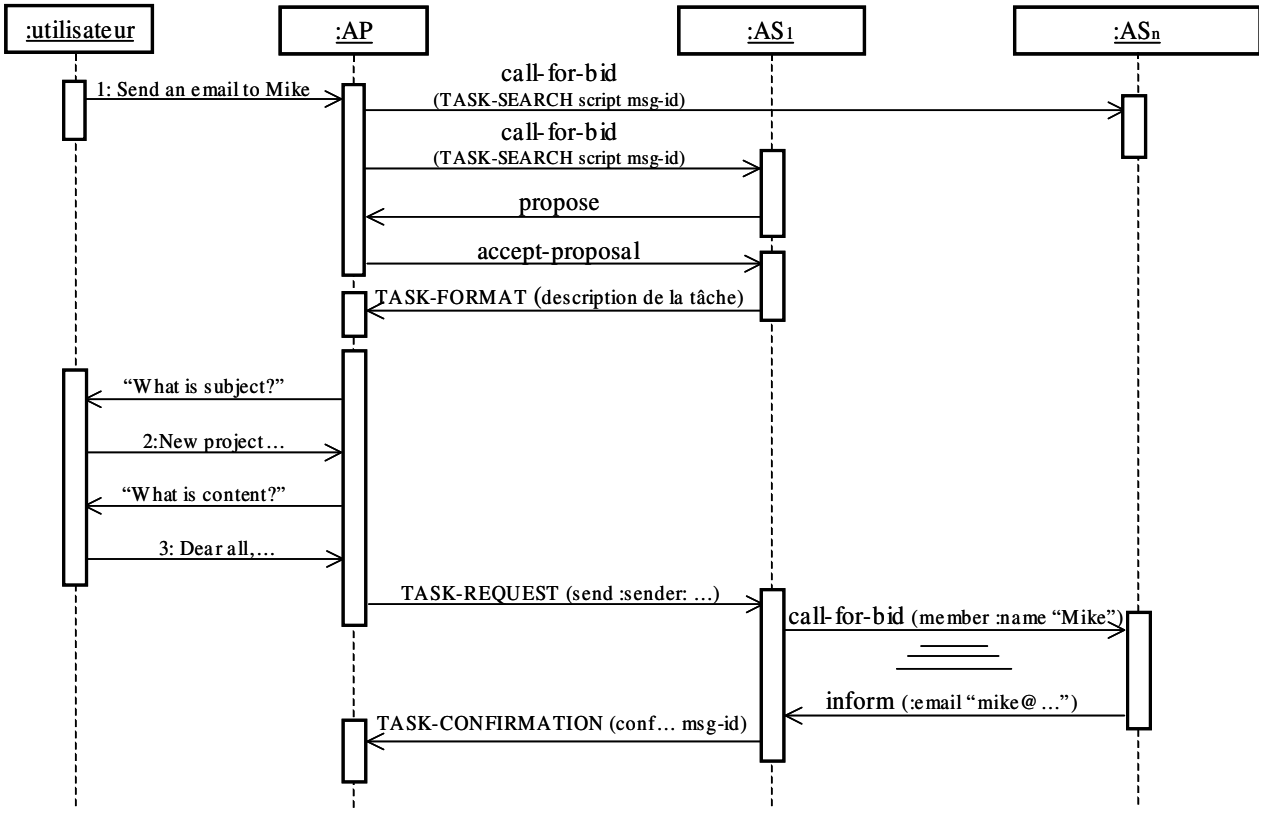
Retour attendu : TASK-FORMAT, TASK-RESULT ou TASK-CONFIRMATION

Exemple :

(TASK-SEARCH (list (address (AddressBook (:name "Mike")))) 8)[19]

(TASK-SEARCH (send (E-Message (:receiver "Marc"))) 25)[20]

Diagramme de séquence :



[19] Ce message est un exemple où l'agent de service exécute la demande et retourne directement le résultat, vu qu'il n'a pas besoin d'autres informations.

[20] Cet exemple, en revanche, amène l'agent de service à renvoyer à l'agent demandeur un message du type TASK-FORMAT, sollicitant des informations complémentaires.

1.7. Le message TASK-FORMAT

Description : message envoyé par un agent de service à l'agent assistant lui spécifiant les informations à négocier auprès de l'utilisateur.

Format : (TASK-FORMAT *task-description msg-id*)

Paramètres :

Non	Description
task-description	Une structure du type task-description
msg-id	Identificateur de contrôle

Forme d'envoi : directe.

Priorité donnée par l'agent assistant lors de la réception : haute.

Retour attendu : aucun.

Exemple :

```
(TASK-FORMAT (<Task>
        <prm prm_id="1">
              <name>RECEIVER</name>
              <synset>receiver to</synset>
              <question>Who is the receiver</question>
              <type>propern</type>
              <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>
              <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>)
    105)
```

Diagramme de séquence : voir diagramme du message TASK-SEARCH.

1.8. Le message TASK-REQUEST

Description : message envoyé par un agent en direction d'un autre demandant l'exécution d'une tâche. Celui qui reçoit s'engage à répondre dans le plus court délai sans pour autant le donner aucune garantie.

Format : (TASK-REQUEST *script msg-id*)

Paramètres :

Non	Description
Script	Liste du format d'une requête formelle
msg-id	Identificateur de contrôle

Forme d'envoi : directe.

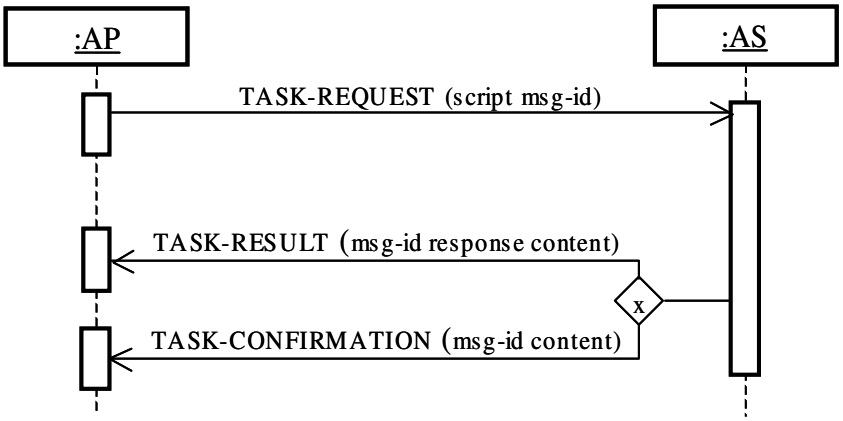
Priorité donnée par l'agent assistant lors de la réception : non applicable.

Retour attendu : TASK-RESULT ou TASK-CONFIRMATION

Exemple :

```
(TASK-REQUEST (@SENDEMAIL (:receiver "Marc")
                          (:subject "Setting a meeting…")
                          (:content "Dear Marc, Mrs. Mary Smith
is working …")) 2)
```

Diagramme de séquence :



1.9. Le message TASK-RESULT

Description : ce message contient le résultat de l'exécution d'une tâche.

Format : (TASK-RESULT *msg-id response content*)

Paramètres :

Non	Description
msg-id	Identificateur de contrôle reçu lors de l'envoi du message TASK-REQUEST
Response	Valeur booléenne correspondant à *true* pour une exécution réussie ou *false* pour une exécution non réussie
Content	Résultat de l'exécution pour y être affichée à l'utilisateur

Forme d'envoi : directe.

Priorité donnée par l'agent assistant lors de la réception : haute.

Retour attendu : aucun.

Exemple :

```
(TASK-RESULT 2 t "Here is the list of meetings for tomorrow…")
```

Diagramme de séquence : voir diagramme du message TASK-REQUEST.

1.10. Le message TASK-CONFIRMATION

Description : message de confirmation d'exécution d'une tâche. C'est à l'agent de service de décider l'envoi de la confirmation.

Format : (TASK-CONFIRMATION *msg-id content*)

Paramètres :

Non	Description
msg-id	Identificateur de contrôle reçu lors de l'envoi du message TASK-REQUEST
content	Message textuel informatif d'exécution de la tâche

Forme d'envoi : directe.

Priorité donnée par l'agent assistant lors de la réception : basse.

Retour attendu : aucun.

Exemple :

```
(TASK-CONFIRMATION 2 "The email was sent with success")
```

Diagramme de séquence : voir diagramme du message TASK-REQUEST.

2. Les structures de données

Les structures de données utilisées dans les messages sont présentées dans cette section.

2.1. La structure QUESTION-DESCRIPTION

Cette structure balisée contient les champs suivants :

```
<prm>
 <name></name>
          <synset></synset>
          <question></question>
 <type></type>
 <required></required>
 <input_source></input_source>
</prm>
```

Non	Description
name	Nom de la variable d'identification
synset	Liste de mots synonymes
question	Question à être présentée à l'utilisateur
type	Type d'information attendue (les mêmes types syntaxiques de l'analyseur syntaxique)
required	Booléen qui indique si la valeur doit impérativement être remplie (valeur *true*) ou pas (valeur *false*)
input_source	Indique si l'entrée doit être saisie par clavier (valeur *keyboard*) ou par le microphone (valeur *vocal*)

2.2. La structure TASK-DESCRIPTION

Cette structure balisée est, en effet, le modèle de tâches du système :

Non	Description
prm	Liste de paramètres du

	type QUESTION-DESCRIPTION ajoutée du champ *default_value*
script	Liste d'arguments pour déclancher un *skill* d'un des agents de service (celui responsable pour exécuter la tâche). Ce champ est rempli par le moteur du dialogue
authorization	Champ booléen signalant si le moteur de dialogue doit (valeur *true*) ou pas (valeur *false*) demander à l'utilisateur l'autorisation d'envoyer la tâche à l'exécution

Annexe III

Un extrait des données des expérimentations

Dans cette annexe, nous présentons un extrait des données recueillies au cours de nos expérimentations.

"Start new conversation - 9/14/2005 : 3:42:32 PM"

"SYS: Welcome to the Personal Assistant"

"USR: I need some help"

"Start new task - 3:42:36 PM"

"Exec new task - 3:42:36 PM"

"USR: please locate and open the report on agent simulation"

"SYS: I understood you want more than one thing"

"SYS: I will start by the first one:"

"SYS: please locate the report on agent simulation"

"Start new task - 3:43:02 PM"

"SYS: I can execute the task To locate a document but there are optional information to be filled!"

"SYS: Do you want to fill them?"

"USR: no"

"Exec new task - 3:43:37 PM"

"SYS: This old demand will be treated:"

"SYS: please open the report on agent simulation"

"Start new task - 3:43:38 PM"

"Exec new task - 3:43:38 PM"

"USR: can you by a ticket from Paris to London"

"Out of domain"

"SYS: Your request is out of my domain!"

"USR: when does the flights from London arrive"

"Out of domain"

"SYS: Your request is out of my domain!"

"USR: send an e-mail to Mike and to Paul"

"SYS: I understood you want more than one thing"

"SYS: I will start by the first one:"

"SYS: send an e-mail to Mike"

"Start new task - 3:45:12 PM"

"SYS: What is the subject?"

"USR: testing agent"

"Exec new task - 3:46:39 PM"

"SYS: This old demand will be treated:"

"SYS: send an e-mail to Paul"

"Start new task - 3:46:39 PM"

"SYS: What is the subject?"

"USR: testing agent"

"Exec new task - 3:47:58 PM"

"USR: least 0 meetings in my office tomorrow morning"

"Word errors: 4"

"Out of domain"

"USR: mist all our meetings, all fixed tomorrow morning"

"Unknown input"

"Word errors: 1"

"Out of domain"

"USR: Can you list all meetings in my office with Mike"

"Start new task - 3:49:27 PM"

"SYS: yes"

"SYS: I can execute the task To list a meeting but there are optional information to be filled!"

"SYS: Do you want to fill them?"

"SYS: Give the date"

"USR: tomorrow"

"Exec new task - 3:50:26 PM"

"USR: close the assistant"

"SYS: Closing the assistant!"

"Close application - 3:50:47 PM"