Titre: Title:	Architecture multi-agents pour la recherche d'information à partir de sources hétérogènes reliées en réseaux
Auteur: Author:	Sophie-Julie Pelletier
Date:	1997
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Pelletier, S.-J. (1997). Architecture multi-agents pour la recherche d'information à partir de sources hétérogènes reliées en réseaux [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/6728/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/6728/
Directeurs de recherche: Advisors:	Hai-Hoc Hoang, & Samuel Pierre
Programme: Program:	Génie électrique

UNIVERSITÉ DE MONTRÉAL

ARCHITECTURE MULTI-AGENTS POUR LA RECHERCHE D'INFORMATION À PARTIR DE SOURCES HÉTÉROGÈNES RELIÉES EN RÉSEAUX

SOPHIE-JULIE PELLETIER

DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET DE GÉNIE INFORMATIQUE

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION

DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES

(GÉNIE ÉLECTRIQUE)

MAI 1997

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

ARCHITECTURE MULTI-AGENTS POUR LA RECHERCHE D'INFORMATION À PARTIR DE SOURCES HÉTÉROGÈNES RELIÉES EN RÉSEAUX

présenté par : PELLETIER Sophie-Julie

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de :

M. BERNARD Jean-Charles, Ph.D., président

M. HOANG Hai-Hoc., Ph.D., membre et directeur de recherche

M. PIERRE Samuel, Ph.D., membre et codirecteur de recherche

M. LEFÈBVRE Bernard, Ph.D., membre

À Daniel. Sans ton humour au quotidien et ta volonté à me voir atteindre le but, j'en serais encore sans doute à soupirer sur la première ébauche ...

REMERCIEMENTS

Le dépôt de ce mémoire n'aurait sans doute jamais vu le jour n'eut été l'appui reçu de Samuel tout au long de la recherche. La confiance et la détermination dont il a fait preuve furent infléchissables à travers le temps, les changements de priorités personnelles, et la fluctuation de ma motivation. Le contenu de la recherche a aussi grandement bénéficié des conseils judicieux et des relectures, de Hoc et de Samuel. Merci à vous deux.

Merci à tous ceux de ma famille et de mes amis qui ont su patienter au cours de ces deux années que je sois enfin disponible pour les activités et le quotidien ...

Finalement, je remercie le Fonds FCAR dont la contribution financière a permis la réalisation de ces travaux de recherche.

RÉSUMÉ

L'avancement des technologies de l'information a provoqué au cours des dernières années une explosion phénoménale des sources d'information : les CD-ROM, les bases de données corporatives et les millions de sites internet en sont quelques exemples. Par conséquent, la complexité de la tâche de recherche d'information s'est grandement accrue. De plus, le rôle du bibliothécaire traditionnel s'est graduellement transféré vers les utilisateurs d'information, ces derniers se chargeant de plus en plus eux-mêmes de leur recherche d'information. Étant donné l'accroissement de la complexité de cette tâche, dû au nombre et à l'hétérogénéité des sources disponibles, il devient nécessaire de mettre à la disposition des utilisateurs des mécanismes pouvant sinon automatiser du moins faciliter la recherche d'information parmi la multitude des sources disponibles. Puisque les sources d'information sont maintenant majoritairement disponibles sur support numérique (CD-ROM, réseaux locaux et mondiaux d'ordinateurs), un assistant de recherche électronique, possédant les connaissances sur l'utilisateur et sur l'ensemble des sources, semble une solution appropriée pour faciliter le travail de l'utilisateur.

L'élaboration d'un tel assistant électronique n'est cependant pas simple. En effet, afin de remplacer adéquatement un assistant humain, l'assistant électronique élaboré doit, entre autres choses, posséder la capacité d'identifier, d'explorer et de connaître les sources d'information disponibles ainsi que leur contenu, de connaître les utilisateurs pour lesquels il travaille, d'utiliser les connaissances acquises sur les sources et les

utilisateurs afin d'accroître la qualité des résultats de recherche, et faire preuve d'autonomie et de pro-activité dans le raffinement du processus utilisé pour la recherche d'information, y compris dans l'identification de nouvelles sources d'information. Les tâches à accomplir sont donc nombreuses et font appel à des aptitudes très variées. Force est de constater que l'intelligence d'un tel assistant ne peut être conçue de façon monolithique.

Ce mémoire propose donc une approche plus modulaire à travers laquelle l'intelligence de l'assistant peut être distribuée. Un tel système prescrit que chaque module composant l'assistant accomplisse de façon intelligente la tâche à laquelle il est dédié. Il nécessite également que des mécanismes de collaboration et de communication soient mis à la disposition de ces modules afin que ces derniers orientent leurs activités individuelles vers le comportement global visé. La théorie des systèmes multi-agents répond à ces besoins. Un agent intelligent est décrit comme une entité abstraite, spécialisée dans la mise en oeuvre d'une activité précise et capable d'agir sur son environnement. Cet agent possède une représentation partielle de son environnement et, dans un univers multi-agents, est capable de communiquer avec d'autres agents. Le comportement de chacun des agents est donc la conséquence de ses observations, de ses connaissances et de ses interactions avec d'autres agents. Dans l'élaboration d'un assistant intelligent de recherche d'information, il s'agit donc d'intégrer plusieurs agents, spécialisés dans différents domaines, pour bâtir un système complexe mettant à contribution l'expertise des agents individuels.

Les travaux présentés dans ce mémoire portent sur l'élaboration et l'implantation partielle d'une architecture multi-agents, nommée ISAME, pour la recherche intelligente d'information parmi des sources d'information hétérogènes et distribuées. Cette architecture constitue en fait une bibliothèque virtuelle offrant, à des agents représentant des utilisateurs, un accès simplifié à un ensemble dynamique de sources d'information disponibles sous format électronique ainsi qu'à des services visant à faciliter et à optimiser cette recherche d'information. L'architecture proposée apporte plusieurs contributions aux notions d'architecture, de communication et de langage propres à la théorie multi-agents. Au niveau architecture, ISAME propose l'utilisation d'une série d'agents de soutien servant d'intermédiaire entre les agents consommateurs et producteurs d'information, de façon à faciliter la rencontre de ces agents, ainsi qu'à optimiser leurs possibilités de collaboration. Au niveau communication, ISAME tente de contourner les problèmes inhérents aux techniques de tableau noir et de messages en implantant une technique hybride basée sur l'utilisation de boîtes aux lettres et de commis. ISAME se charge d'aiguiller tous les messages échangés entre les agents inscrits auprès de l'environnement. Ceci évite à chacun des agents inscrits de tenir à jour des informations sur les agents enregistrés et sur l'adresse à laquelle les messages doivent leur être expédiés. De plus, le système de boîtes aux lettres, évite aux agents expéditeurs de messages de perdre du temps inutilement à expédier un message à un agent temporairement déconnecté et, d'autre part, ceci évite à un agent destinataire d'être à l'écoute de façon permanente afin d'éviter la perte de messages. ISAME propose aussi

une série de mécanismes de ramasse miettes permettant d'éviter, à l'intérieur de l'environnement, la conservation et la propagation d'information sur des agents ou message devenus inaccessibles ou désuets, ainsi que l'utilisation de ressources par des processus devenus indésirables. Finalement, puisqu'elle est essentiellement liée au problème de recherche d'information, l'architecture ISAME contribue à ce domaine par la mise en place d'éléments permettant aux utilisateurs et aux responsables de sources d'information de participer à l'échange électronique d'information par l'entremise d'agents spécialisés en la matière. Ces éléments comprennent l'utilisation d'un sous-ensemble minimal précis d'énoncés performatifs du langage multi-agents KQML, du langage ISAME-L et de 2 ontologies prédéfinies, soit Biblio-virtuelle et Agent.

La grande force de l'architecture ISAME est de simplifier la recherche d'information dans des sources d'information hétérogènes et distribuées en les rendant transparentes à l'utilisateur. Bien que seulement une partie des composantes de ISAME ait été mise en oeuvre dans le cadre de ce mémoire, il est permis de croire que, grâce au travail des agents intelligents, les utilisateurs n'auront pas à se soucier de la méthode d'implantation, de la localisation, de la disponibilité et du langage d'interaction propre à chacune des sources. ISAME offre aussi les bénéfices d'une architecture multi-agents ouverte, puisqu'elle permet en tout temps à de nouveaux agents représentant des utilisateurs et des sources d'information de s'inscrire et de quitter l'environnement, ce qui en fait un environnement où les ressources sont modifiées dynamiquement selon la disponibilité de chacun.

Les travaux d'implantation présentés dans ce mémoire mettent en relief la simplicité et l'efficacité des mécanismes de communication de l'architecture ISAME. Cependant, plusieurs travaux demeurent à être complétés afin de faire de ISAME un environnement complet pour la recherche intelligente d'information. Dans un premier temps, certains des objets implantés jusqu'à ce jour devront être complétés et testés en profondeur. Dans un deuxième temps, les objets prévus à l'architecture originale mais toujours absents dans l'environnement actuel devront être mis en place.

ABSTRACT

Due to the past decade advances in computer technologies, the task of searching for information has undertaken a phenomenal transformation. New information sources are created daily and techniques for querying these sources are now more diversified then ever. Users thus need efficient help in performing such a task. This thesis proposes a multi-agent architecture, named ISAME, which fulfills this purpose.

ISAME serves as a facilitating environment for four categories of agents, each one being specialized in a series of activities needed to complete a digital version of a user assistant dedicated to intelligent information retrieval. User Agents (UA) represent users in ISAME. Query Agents (QA) handle the life cycle of each query posted by the UAs. Information Agents (IA) represent specific information sources in the ISAME environment and specialize in querying these sources for any given query. Finally, several Support Agents (SA) are created within the environment to facilitate the communication and resource distribution process among the other agents. Communication among the agents takes place through KQML based messages. The ISAME-L language and two specific ontologies, VirtualLibrary and Agent, provide the other tools required by the agents to express information queries and results. These messages are exchanged between the agents by a central router, the Communiation Broker Agent, which then uses a series of dedicated mailboxes and message clerks which, together, form a communication mechanism lying in between the blackboard architecture and the peer-to-peer message mechanism. Finally, ISAME makes extensive

use of garbage collection mechanisms which prevent lock of resources by dead or unwanted agents or processes.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABBRÉVIATIONS

Sigle ou Abréviation	Signification
CA	Communication Area
CBA	Communication Broker Agent
CORBA	Common Object Request Broker Architecture
IA	Information Agent
ISAME	Information - Système d'aide multi-experts
ISAME-L	Langage agent défini spécifiquement pour ISAME
KQML	Knowledge Query Manipulation Language
ORB	Object Request Broker
QA	Query Agent
RA	Registry Agent
RA_{IA} ou RAIA	Registry Agent pour les agents de type IA
RA_{QA} ou RAQA	Registry Agent pour les agents de type QA
RA_{UA} ou RAUA	Registry Agent pour les agents de types UA
RFA	Result Filtering Agent
SFA	Source Filtering Agent
SRII	Système de recherche intelligente d'information
UA	User Agent
UMDL	University of Michigan Digital Library

LISTE DES ANNEXES

CHAPITRE 1

INTRODUCTION

L'un des grands avantages de la technologie informatique est de permettre le stockage, à peu de frais et sur très peu d'espace, de grandes quantités d'information. De plus, la rapidité des ordinateurs et la disponibilité des réseaux de télécommunications à haut débit nous permettent d'avoir accès quasi instantanément à toute information stockée sur support magnétique du domaine public, indépendamment du lieu où se trouve ce support dans le monde. Bien que cet accès à l'information soit efficace lorsqu'on sait où trouver l'information recherchée, il peut devenir rapidement complexe lorsque la source n'est pas connue. Cette complexité augmente considérablement lorsque l'on ne sait même pas quel type d'information peut s'avérer utile. Et, cela va de soi, cette complexité augmente encore davantage lorsque l'utilisateur n'a qu'une connaissance sommaire des sources d'information mises à sa disposition - ce qui est habituellement le cas - sans mentionner que l'organisation et l'utilisation de ces sources peuvent grandement varier de l'une à l'autre.

Il devient donc nécessaire de mettre à la disposition des utilisateurs des mécanismes pouvant automatiser ou du moins faciliter cette tâche de recherche d'information. De plus, le nombre et la durée des interactions avec l'utilisateur lui-même devront être minimisés, la recherche d'information étant souvent une tâche connexe à la tâche principale. Ces mécanismes de recherche intelligente d'information, aussi appelés

« *knowbots* » (Savetz, 1996), sont comparables à des secrétaires particuliers spécialisés, suffisamment intelligents pour s'adapter au profil de l'utilisateur. La mise en place d'un tel système de recherche intelligente d'information, capable de consulter plusieurs sources hétérogènes, constitue le coeur du mémoire présenté ici. Dans ce chapitre, les problèmes inhérents à la démocratisation de l'accès à l'information ainsi que les éléments à considérer dans l'élaboration d'une solution sont abordés. On y présente également les objectifs de recherche, l'originalité des travaux réalisés et les contributions du présent mémoire.

1.1 Motivations et contexte de recherche

Deux grands buts sont poursuivis dans l'élaboration d'outils de recherche d'information. Il s'agit d'abord de mettre à la disposition des utilisateurs un vaste choix de sources d'information, puis de leurs fournir les services habituellement offerts par un bibliothécaire compétent. L'intelligence de tels outils se caractérise donc par une habileté à répondre adéquatement aux besoins particuliers de chaque utilisateur. Les sections qui suivent dressent un tableau des caractéristiques à considérer lors de l'élaboration de ces outils de recherche d'information.

1.1.1 Démocratisation de l'accès à l'information

L'informatique et les réseaux télématiques ont eu pour effet de démocratiser l'accès à l'information. En effet, jusqu'à tout récemment, la recherche d'information s'était résumée à consulter des index imprimés ou dactylographiés de notices bibliographiques

au format uniforme dans les bibliothèques publiques ou privées, à consulter des microfiches indexées manuellement, ou encore à faire appel à des organismes comme Statistiques Canada. Bien que ces façons d'accéder à l'information fussent relativement simples à appliquer, les sources d'information étaient souvent inaccessibles ou dispersées géographiquement, restreignant ainsi les utilisateurs dans leurs recherches ou, du moins, imposant des délais considérables avant l'obtention de résultats concrets.

Depuis quelques années, la recherche d'information a subi des transformations considérables. Les sources d'information sont maintenant plus variées, indexées sur support numérique et, bien souvent, le contenu même des sources d'information est accessible sous format numérique. Il est maintenant possible d'avoir accès à de vastes sources d'information, que ce soit par des CD-ROMS (dictionnaires, résumés d'articles de périodiques, actes de conférences, encyclopédies, ...), par des réseaux institutionnels (données d'entreprise, bibliothèques universitaires, ...), ou encore par Internet (sites *web*, sites *ftp*, ...). Autre transformation d'importance, les distinctions entre les formats de stockage électronique et de visualisation des différents types de médias d'information ont pratiquement disparu. On peut maintenant visualiser textes, photographies et vidéos sur un même poste de travail informatisé grâce à l'avènement des logiciels multimédias.

Bien que cette démocratisation comporte de nombreux avantages, elle entraîne une plus grande complexité dans la recherche d'information. Le nombre de sources disponibles est sans cesse en croissance. Chaque source supporte son propre outil de recherche d'information et fournit les résultats dans un format précis qui lui est souvent

unique. L'utilisateur se retrouve donc confronté à de nombreuses questions lors d'une recherche donnée. En voici un échantillon :

- quelles sources sont les plus pertinentes en début de recherche?
- comment exprimer le thème de la recherche pour une source donnée? Comment raffiner cette recherche sur cette même source?
- quelles sont la fiabilité et la qualité des informations accessibles par l'intermédiaire d'une source donnée?
- si deux sources fournissent des informations contradictoires, comment déterminer la source la plus crédible?
- à quel rythme le contenu d'une source d'information change-t-il?

Cette démocratisation de la recherche d'information entraîne un autre changement majeur, soit la presque disparition du rôle des bibliothécaires. En effet, puisque l'utilisateur a maintenant accès, directement de son poste de travail, à la plus grande partie des sources d'information qui lui sont nécessaires, il tentera d'effectuer lui-même ses recherches d'information avant de faire appel aux services d'un bibliothécaire.

Les utilisateurs tentent ainsi d'accomplir une tâche complexe pour laquelle ils n'ont souvent pas de formation spécifique. Ils risquent donc de passer à côté de nombreuses sources pertinentes au sujet de recherche, ou encore de prendre un temps considérable à effectuer la recherche sans obtenir de résultats satisfaisants. Il importe aussi de souligner que cette tâche complexe de recherche d'information est souvent complémentaire à

d'autres tâches plus prioritaires : formation, planification d'entreprise, rédaction d'articles, recherche scientifique, analyse financière, etc.

1.1.2 Aide intelligente pour la recherche d'information

Face à ce transfert de tâche des spécialistes de l'information vers les utilisateurs, *il devient impératif de munir les utilisateurs d'outils pouvant leur permettre d'effectuer leurs recherches d'information de façon efficace.* Cette recherche s'effectue de plus en plus à l'aide d'ordinateurs, étant donné qu'un nombre croissant de sources d'information sont maintenant disponibles sur des supports électroniques. De plus, les informations trouvées (textes, photos, vidéos) sont souvent manipulées à l'intérieur d'un environnement informatisé. Il est donc nécessaire que les outils rendus disponibles au poste de l'utilisateur s'intègrent naturellement à la fois à l'environnement de travail de l'utilisateur et à ces sources d'information numérique. De plus, pour être efficaces, de tels outils doivent être en mesure de considérer le contexte de la requête (Qui est l'utilisateur? Pour quel motif la recherche est effectuée? etc.), et doivent être aussi capables d'évaluer la pertinence et la qualité des informations trouvées (King, 1995) en fonction de la requête. La métaphore bibliothécaire en recherche d'information permet, en se basant sur une analyse heuristique des tâches effectuées traditionnellement par ces assistants (Digital Library Project Research Proposal, 1995), de dresser une liste des caractéristiques que devrait rencontrer un outil « idéal ». En effet, un bibliothécaire, ou un assistant humain spécialisé en recherche d'information, possède, entre autres, les aptitudes suivantes :

- connaissance des sources d'information : Il connaît les sources d'information auxquelles il a accès : contenu, méthode d'accès, format de l'information, fréquence de mises à jour. Il explore sans cesse les nouvelles possibilités de sources d'information.
- connaissance du client : Il connaît le profil de son client (ou utilisateur), soit les préférences et besoins de ce dernier. Il sait ajuster ce profil au fur et à mesure des interactions avec son client.
- utilisation de ses connaissances : À partir du profil de son client et de la requête donnée, il peut diriger sa recherche vers les sources les plus adéquates.
- présentation adéquate des résultats : Il sait présenter les résultats de ses recherches selon un format pratique pour son client (sélection des documents les plus pertinents, notices bibliographiques seulement, résumés seulement, par ordre de pertinence, etc.), ce format pouvant varier selon le contexte de la requête. La quantité de résultats présentés est aussi ajustée au contexte.
- suggestions de raffinement de recherche : Il est en mesure de guider son client dans la définition de sa requête d'information en aidant ce dernier à préciser sa requête soit dès le départ, soit par des mécanismes de raffinements successifs.
- explication du cheminement de recherche : Il est capable de fournir des explications quant au cheminement utilisé pour compléter la recherche d'information, formant par le fait même son client à la recherche efficace d'information, évitant ainsi de rendre celui-ci complètement dépendant de ses services.

- autonomie et pro-activité : Il est à l'affût des nouvelles informations pouvant s'avérer intéressantes pour son client, cumule ces informations et les présente à celui-ci au moment opportun.

Force est de constater que les outils de recherche associés aujourd'hui aux sources d'information électroniques, tels que les CD-ROM, les outils de requêtes des systèmes de gestion de base de données (SQL ou autre), ou encore les outils de recherche disponibles sur le *web*, ne couvrent, chacun, que certaines de ces caractéristiques. Entre autres, chacun de ces outils ne peut habituellement manipuler efficacement qu'une ou des sources d'information données, peu de ces outils peuvent intégrer un profil utilisateur dans leurs techniques de recherche, et encore moins d'outils font preuve d'autonomie dans leurs activités. À la lumière des caractéristiques associées à l'assistant de recherche d'information humain, le présent mémoire soulève la question suivante : est-il possible de créer une version partiellement ou complètement automatisée d'un tel assistant?

1.2 Éléments de problématique

L'implantation de chacune des aptitudes identifiées chez l'assistant de recherche humain constitue un problème complexe en soi dans le cadre d'une implantation informatisée. Il s'ensuit que l'intégration de l'ensemble de ces fonctionnalités à l'intérieur d'un seul outil, soit l'utilisation d'un modèle monolithique, s'avère inefficace parce que trop complexe à modéliser et à maintenir face aux changements rapides

survenants dans le domaine de la recherche automatisée d'information. Il devient donc nécessaire d'envisager une approche plus modulaire dans le traitement de ce problème.

La création d'un outil intelligent de recherche d'information soulève plusieurs problèmes de taille. Ces problèmes sont principalement reliés à la notion d'intelligence que l'on désire associer à l'assistant. Si l'humain arrive à accomplir toutes ces tâches à l'aide de son seul système cognitif, il semble qu'aucune des techniques d'intelligence artificielle présentement connues n'englobe l'ensemble de ces aptitudes. Si l'on pense seulement à chacune des sources d'information qui peuvent être utilisées par l'assistant de recherche, il est évident qu'il n'existe pas d'outil de recherche universel et optimal pour toutes les sources. Mentionnons comme exemple que l'outil de recherche dans une base de données relationnelle, soit le langage SQL, ne sera pas le même que celui utilisé pour une recherche de type plein texte, soit une combinaison d'indexeur lexical et sémantique. Chacun de ces outils obéit à des règles très précises, bien que le but final soit le même pour tous, soit de trouver l'information la plus pertinente pour une requête donnée.

Concevoir un assistant intelligent ayant la capacité d'explorer lui-même l'ensemble des sources d'information exigerait que cet assistant ait une connaissance absolue des outils de recherche possibles actuels et futurs, ainsi que la connaissance sur la façon de sélectionner le bon outil de recherche pour une source donnée. Pour N types de sources différentes, l'assistant devra connaître N outils de recherche différents. Sans oublier que l'assistant doit être en mesure de transformer les éléments d'information d'un standard

d'outil de recherche à l'autre s'il veut être en mesure d'utiliser les résultats de recherche fournis par une source comme éléments de requête pour une autre source. Comme l'illustre la Figure 1.1, la connaissance de *N* outils de recherche signifie donc *N(N-1)* protocoles de transformation.

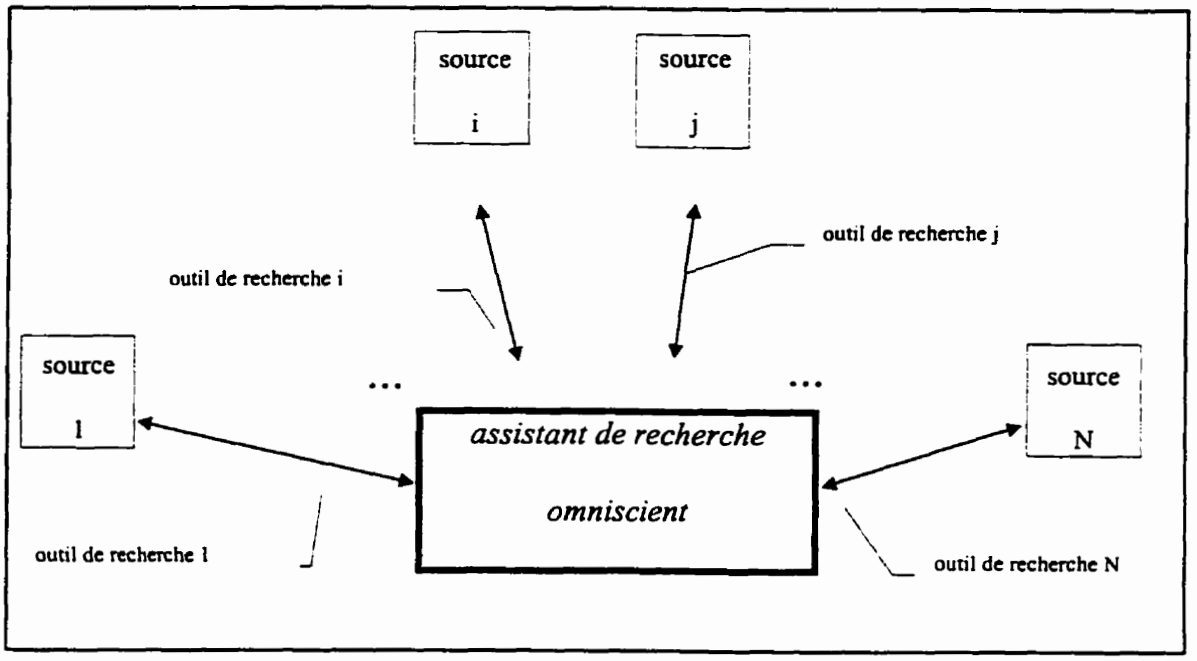


Figure 1.1 Protocoles de communication et de transformation associés à un assistant de recherche.

La diversité et l'hétérogénéité aussi bien des sources d'information que des techniques de recherche associées ne constituent qu'un aspect de la non-uniformité des connaissances nécessaires à un assistant intelligent pour la recherche d'information. D'autres tâches tout à fait distinctes mais complémentaires à la recherche dans les sources d'information nécessitent elles aussi des techniques d'implantation répondant à des caractéristiques très spécifiques. Ainsi, établir le profil d'un utilisateur requiert des connaissances précises et distinctes de celles requises pour l'exploration de nouvelles

sources d'information. Autre exemple, filtrer les résultats de recherche pour qu'ils répondent adéquatement aux besoins des utilisateurs est une tâche très différente de celle visant à fournir une explication sur le cheminement utilisé pour trouver l'information. Cette brève analyse nous amène donc au constat suivant : l'intelligence d'un tel assistant ne peut être conçue de façon monolithique (Birmingham, 1995; Knoblock, 1994).

1.3 Fondements de l'architecture

Une architecture plus modulaire à travers laquelle l'intelligence de l'assistant peut être distribuée semble plus appropriée au contexte décrit précédemment. Un tel système nécessite que chaque module composant l'assistant accomplisse de façon intelligente la tâche à laquelle il est dédié. Il nécessite également que des mécanismes de collaboration et de communication soient mis à la disposition de ces modules afin que ces derniers orientent leurs activités individuelles vers le comportement global visé. Il s'agit donc de suivre la philosophie communément désignée *Divide and Conquer*.

La théorie des systèmes multi-agents tente de répondre à ces besoins. Selon Benslimane (1993), un agent est une entité réelle ou abstraite qui est capable d'agir sur son environnement, qui dispose d'une représentation partielle de cet environnement, qui dans un univers multi-agents peut communiquer avec d'autres agents et dont le comportement est la conséquence de ses observations, de sa connaissance et des interactions avec d'autres agents. Il est évident que cette définition correspond à celle donnée plus haut pour les modules implantant chacune des tâches de l'assistant. Quant aux mécanismes de collaboration et de communication nécessaires à la cohérence des

activités de l'ensemble des modules, ceux-ci se retrouvent dans la définition même d'un système multi-agents : un ensemble d'entités appelées agents, coopérant entre eux aux fins d'exécution d'une activité dite "intelligente" ... l'intelligence du système se trouve distribuée à travers les différents agents du système » (Benslimane, 1993). Chacune des tâches identifiées dans le système est confiée à un agent, les agents devant collaborer afin d'exécuter la tâche globale à laquelle le système est dédié. Par cette collaboration, les agents arrivent à accomplir plus en tant que groupe qu'ils ne le pourraient individuellement (Singh, 1993). À cet effet, plusieurs mécanismes de collaboration et de coopération ont déjà été mis en place à travers le monde, dont plusieurs s'inspirant de principes d'économie (Mullen, 1995) et de la théorie des jeux (Bicchieri et al, 1995).

Il s'agit donc d'intégrer plusieurs agents, spécialisés dans différents domaines, pour bâtir un système plus complexe mettant à contribution l'expertise des agents individuels. L'idée qui sous-tend l'organisation des agents en une société est que la société peut accomplir plus que la somme des possibilités des agents individuels, d'où la mise en valeur du processus de travail collectif. Ceci implique une amélioration de la performance globale, puisque les agents ont la possibilité de développer une grande variété de solutions et de points de vue, découlant en une plus grande variété de solutions pour la collectivité (Chaib-draa, 1993). Voilà exactement ce que nous cherchons à obtenir pour notre assistant intelligent de recherche d'information. Nous tenterons donc de mettre en place et de démontrer partiellement une modélisation multi-agents ermettant

d'atteindre la majorité des objectifs reliés à l'élaboration d'un assistant tel que celui proposé ici.

1.4 Objectifs de recherche

L'objectif principal de ce mémoire est de mettre en place une architecture multi-agents pouvant servir de plate-forme pour la recherche intelligente d'information. Il s'agit donc de définir une architecture multi-agents permettant de créer un assistant répondant aux critères suivants :

- *l'indépendance de l'assistant intelligent par rapport aux modes d'implantation des sources* d'information utilisées ;
- *l'indépendance de l'assistant par rapport aux utilisateurs voulant faire appel à ces sources d'information* et *par rapport aux types de requête* portant sur ces sources ;
- *l'indépendance de ce même assistant vis-à-vis la disponibilité des sources connues.*

Nous désirons aussi planifier à l'intérieur de cette architecture tous les éléments et mécanismes qui permettront éventuellement à l'assistant de guider les utilisateurs dans leurs recherches d'information. L'idée est de permettre l'intégration d'outils de recherche déjà disponibles (outils de recherche disponibles sur le *web*, CD-ROM, etc.), ainsi que l'ajout de nouveaux outils de recherche et sources d'information au fur et à mesure de leur disponibilité. Il faut également rendre transparente à l'utilisateur une utilisation optimale de ces sources par l'entremise d'un seul guide, l'assistant de

recherche. Parmi les éléments et mécanismes à être supportés par l'architecture afin que l'assistant puisse jouer son rôle de guide on retrouve l'éventuelle considération du profil de l'utilisateur lors de la recherche d'information, l'utilisation de techniques intelligentes de sélection de sources d'information, ainsi que l'habileté à fournir une explication quant au raisonnement utilisé pour mener la recherche à terme. Notons ici qu'un utilisateur se définit soit comme un humain, soit comme un processus informatique.

Pour atteindre ces objectifs, il nous faut dans un premier temps élaborer l'architecture et les composantes intelligentes de l'assistant, fournir à ces composantes des mécanismes de communication leur permettant de collaborer efficacement et, dans un deuxième temps, implanter l'architecture choisie afin de vérifier sa faisabilité ainsi que sa validité.

1.5 Contribution et originalité des travaux

L'architecture présentée dans le cadre de ce mémoire repose essentiellement sur l'élaboration d'un système multi-agents hétérogène distribué sur des réseaux de télécommunication. Cette architecture, nommée ISAME, tente de tirer profit des nombreux avantages présentés par les systèmes multi-agents tels que mentionnés à la section 2.3.4.1, tout en proposant des solutions à certains des désavantages qui leur sont reliés. Ainsi, l'architecture proposée est largement inspirée de l'ensemble de travaux de recherche présentés au chapitre 2, plus particulièrement des notions d'architecture définies par Lander et Lesser (1994) et par l'équipe du projet UMDL (Digital Library Project Research Proposal, 1995) présenté à l'Annexe E, ainsi que du langage KQML

développé par Finin et son équipe (1993). L'architecture proposée apporte plusieurs contributions aux notions d'architecture, de communication et de langage propres à la théorie multi-agents. Au niveau architecture, ISAME propose l'utilisation d'une série d'agents de soutien servant d'intermédiaire entre les agents consommateurs et producteurs d'information, de façon à faciliter la rencontre de ces agents, ainsi qu'à optimiser leurs possibilités de collaboration. Au niveau communication, ISAME tente de contourner les problèmes inhérents aux techniques de tableau noir et de messages en implantant une technique hybride basée sur l'utilisation de boîtes aux lettres et de commis. L'architecture tire profit du protocole de communication réseau TCP/IP. ISAME propose aussi une série de mécanismes de ramasse miettes permettant d'éviter, à l'intérieur de l'environnement, la conservation et la propagation d'information sur des agents ou message devenus inaccessibles ou désuets, ainsi que l'utilisation de ressources par des processus devenus indésirables. Finalement, puisqu'elle est essentiellement liée au problème de recherche d'information, l'architecture ISAME contribue à ce domaine par la mise en place d'éléments permettant aux utilisateurs et aux responsables de sources d'information de participer à l'échange électronique d'information par l'entremise d'agents spécialisés en la matière. Ces éléments comprennent l'utilisation d'un sous-ensemble minimal précis d'énoncés performatifs du langage KQML, du langage ISAME-L et de 2 ontologies pré définies, soit Biblio-virtuelle et Agent.

1.6 Organisation du mémoire

Le prochain chapitre propose une revue sélective de littérature servant de fondements aux travaux de ce mémoire. Le chapitre 3 expose le modèle conceptuel proposé pour la mise en place de l'assistant intelligent de recherche d'information selon un modèle multi-agents. Le chapitre 4 expose les choix et considérations d'implantation. Le chapitre 5 détaille les résultats d'une expérimentation conduite avec une partie de l'architecture proposée, les résultats d'analyse de cette expérimentation. Le chapitre 6 esquisse des orientations de recherches futures.

CHAPITRE 2

RECHERCHE INTELLIGENTE D'INFORMATION

ET SYSTÈMES MULTI-AGENTS

Si l'on pouvait reproduire tel quel le modèle des bibliothèques classiques dans le cadre d'un réseau informatique, on se retrouverait probablement avec un groupe de bibliothécaires virtuels, appelés assistant de recherche d'information, travaillant pour chacun d'entre nous sur notre poste de travail. Ces assistants sauraient utiliser efficacement les sources d'information directement accessibles, ou encore faire appel aux services de « prêts entre bibliothèques » pour acquérir des documents disponibles dans d'autres bibliothèques. Ces assistants sauraient sans doute nous reconnaître au fil des visites et améliorer la qualité de leur service au cours des interventions. La bibliothèque virtuelle intégrerait donc à la fois contenus, soit les informations proprement dites, et services, soit ceux offerts par ses bibliothécaires. Bien que plusieurs sources d'information soient déjà accessibles aux utilisateurs par le biais d'outils de recherche, que ce soit sur l'Internet ou sur des réseaux privés, le concept de bibliothèque virtuelle intégrée, c'est à dire comme lieu unique d'accès à l'ensemble des contenus et à des services complémentaires, n'en n'est encore qu'à ses premiers balbutiements. Ce mémoire propose un modèle pouvant servir de solution de base à l'implantation de la bibliothèque virtuelle intégrée.

En premier lieu, ce chapitre présente une critique des bibliothèques virtuelles telles qu'elles se présentent à ce jour. Il explique ensuite comment les structures englobantes distribuées peuvent être mises à contribution dans la création de bibliothèques virtuelles évoluées. Par la suite, une revue de littérature partielle dresse les principes de base d'un type de structure englobante distribuée, la modélisation par système multi-agents.

2.1 Bibliothèques virtuelles

L'avènement des réseaux de télécommunication combiné à la prolifération des sources d'information disponibles sur supports électroniques a donné naissance à un tout nouveau modèle de bibliothèque : les *bibliothèques virtuelles*. Ces bibliothèques nous ouvrent certes l'accès à un nombre sans cesse croissant de sources d'information des plus variées. Cependant, telles qu'elles existent à ce jour, ces mêmes bibliothèques sont difficilement exploitables efficacement. En effet, puisqu'elles n'intègrent pas toujours les services offerts par le personnel des centres de documentation traditionnels et qu'elles ne comportent pas de systèmes de classification tels celui développé par Dewey, la plupart de ces bibliothèques ne sont présentement que de vastes entrepôts virtuels de documents. En d'autres mots, est bien chanceux (ou patient, ou très expérimenté !) celui qui y trouve quelqu'information pertinente à un sujet donné !

L'information est donc déjà disponible sous format électronique et, avouons-le, si l'on sait dans quelle source d'information se trouve le document qui nous intéresse, accéder à ce document à partir de notre poste de travail facilite grandement notre vie. Mais qu'arrive-t-il lorsque l'on ne sait pas où trouver le document cherché, comment

effectuer la recherche dans une source donnée ou, encore pire, comment peut-on identifier les documents pertinents à un sujet lorsque l'on ne sait même pas quel genre d'information est mise à notre disposition? Pour l'instant, il demeure sans doute encore plus simple de s'adresser au service de documentation de notre bibliothèque que de passer des nuits blanches à explorer l'ensemble des sources mises à notre disposition, y compris l'Internet ! Ne perdons cependant pas espoir, de nombreux projets de recherche présentement en cours tentent de trouver une solution visant l'implantation des bibliothèques virtuelles.

2.2 Définition de structures englobantes distribuées

Afin de tirer profit d'un ensemble de documents organisés en collections, il faut plus qu'un outil de recherche pour l'ensemble des collections. En effet, de façon à automatiser la tâche du bibliothécaire traditionnel, des mécanismes de sélection de sources d'information, d'intégration des résultats fournis par chacune des sources, ainsi que des capacités de filtrage des résultats doivent être intégrés au **SRII** (Système de recherche intelligente d'information). Bien entendu, ces mécanismes ne seraient pas nécessaires si toutes les sources d'information étaient fondues en une seule collection générale. Cependant, le volume d'information produit à chaque jour dans le monde étant considérable, il serait difficile d'implanter un outil de recherche unique suffisamment performants pour pouvoir utiliser une telle collection. De plus, la maintenance d'une telle collection tient de l'irréel. De toute façon, nous l'avons mentionné en introduction, il n'existe pas de format de représentation de connaissances pouvant répondre aux

besoins de l'ensemble des sources. Finalement, les sources d'information étant habituellement mises à jour par leurs auteurs, il est préférable qu'elles soient situées sur un serveur facilement accessible à ces auteurs.

Birmingham (1995) soutient que l'élaboration des bibliothèques virtuelles doit être basée sur la définition de structures fédératives ou englobantes distribuées pouvant exploiter plusieurs sources d'information à la fois. Ces structures doivent être en mesure de réagir intelligemment tant à la volatilité des sources et de leur contenu, qu'à l'hétérogénéité des contenus et outils de recherche mis en place, ainsi qu'à l'expansion phénoménale du nombre de sources. De son côté, Vossos (1993) mentionne que les SRII de cette fin de siècle devront être à la fois distribués et hétérogènes. Knoblock (1994) souligne quant à lui que la construction d'un système unique intégrant toutes les sources d'information est impossible. Il suggère cependant une approche alternative : décentraliser cet accès à l'information en construisant des outils d'information spécialisés, appelés agents, responsables d'un sous-ensemble de l'information, et permettre à ces agents de communiquer et collaborer entre eux afin d'échanger requêtes et information. De cette façon, chaque agent peut encapsuler un modèle détaillé de son domaine d'expertise, tout en bénéficiant des connaissances des autres agents. En complément à ces agents information, une série d'agents utilitaires peuvent être mis en place : agents de contrôle information, agents possédant les méta-connaissances sur les agents information (quel agent est disponible, sa localisation, etc.), agents capables d'intégrer les résultats fournis par les différents agents information, agent interpréteur de

la requête de l'utilisateur, etc. (King, 1995). Ce genre d'implantation des SRII permettrait à la fois de répondre aux requêtes de l'utilisateur et d'optimiser l'utilisation des sources d'information. Cette description des SRII correspond tout à fait à la définition d'un système multi-agents tel que montré à la Figure 2.1. La section qui suit fait un survol de ce domaine.

2.3 Les systèmes multi-agents

Les systèmes multi-agents constituent l'un des domaines de recherche les plus prometteurs pour la recherche intelligente d'information. Ce domaine de recherche bénéficie de résultats de recherche de nombreux autres domaines dont les systèmes distribués, la représentation de connaissances, la modélisation de l'utilisateur et les réseaux de télécommunications. En quelques mots, la représentation d'un problème par un système multi-agents consiste à distribuer la tâche entre plusieurs entités autonomes pouvant collaborer. Cette section présente quelques définitions plus détaillées de ces systèmes et de leurs composantes, ainsi que différents aspects théoriques qui y sont reliés.

2.3.1 Définitions d'agent

Une grande partie de la recherche visant à rendre les ordinateurs plus "évolués" se situe dans les domaines des agents et systèmes multi-agents. Bien que les définitions des concepts d'agent et de multi-agents varient avec les auteurs et les écoles de pensée, Benslimane (1993) en donne une définition générale :

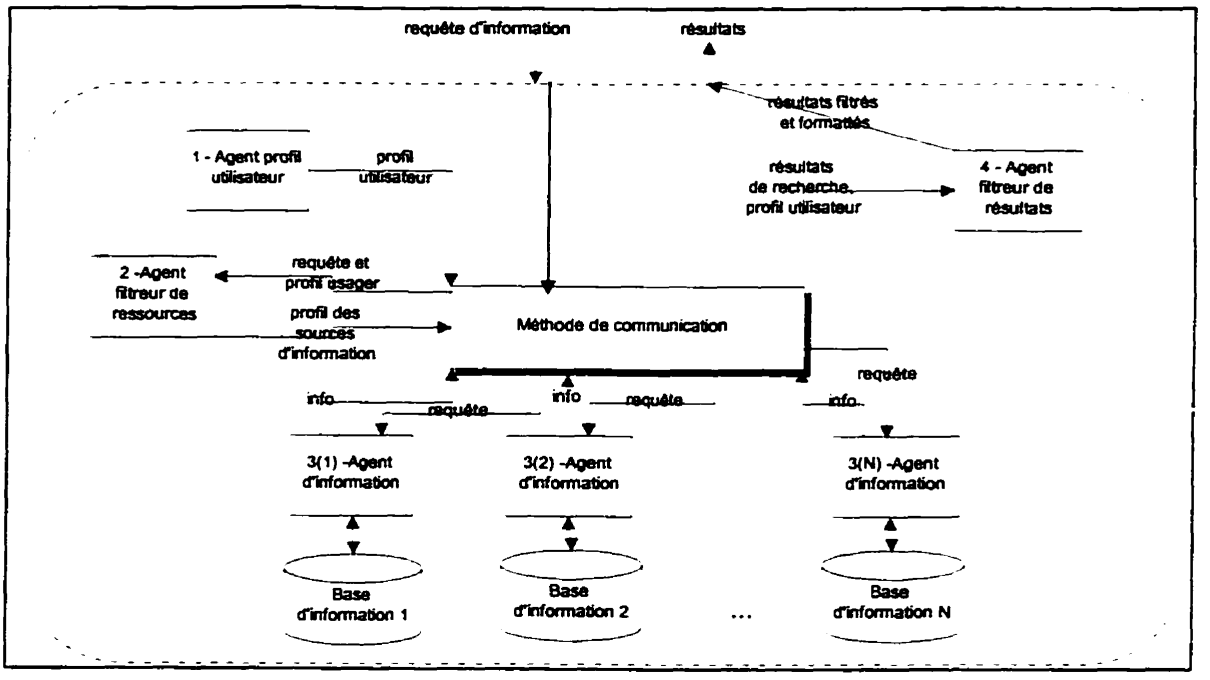


Figure 2.1. Implantation d'un SRII par système multi-agents.

"Un agent est une entité réelle ou abstraite qui est capable d'agir sur son environnement, qui dispose d'une représentation partielle de cet environnement, qui dans un univers multi-agents peut communiquer avec d'autres agents et dont le comportement est la conséquence de ses observations, de sa connaissance et des interactions avec d'autres agents."

Selon King (1995),

"un agent est un assistant pour une personne ou un processus qui accomplit une action ou comble un besoin; il possède une capacité de prise de décision qui s'apparente aux intentions décrites chez l'humain."

Michael Wooldridge (1994), chercheur très connu dans le domaine des systèmes agents, propose une définition élémentaire d'agent :

"un agent est une composante matérielle ou logicielle qui est autonome (agit sans l'intervention directe d'un humain ou autre entité), possède des habiletés sociales (interagit avec d'autres agents), est réactive (les agents perçoivent leur environnement et réagissent en temps réel aux changements qui y surviennent), et sont pro-actifs (capables d'agir dans le but d'atteindre un but en prenant des initiatives)."

L'agent est donc vu ici comme un processus ayant accès à un ensemble d'informations et étant capable d'agir dans un environnement évolutif, à la manière de l'homme. Il est muni d'une certaine capacité de réflexion sur lui-même et sur autrui, ce qui lui permet d'ajuster son comportement selon les fluctuations du contexte. Il est capable d'intervenir dans un champ d'activité restreint, mais bien cerné. On peut donc parler d'un agent comme étant un spécialiste d'une tâche donnée.

2.3.2 Modularité et intelligence

L'une des grandes attentes des chercheurs face aux agents est la possibilité d'utiliser ceux-ci dans le cadre de systèmes multi-agents. Selon Benslimane (1993), un système multi-agents est :

« un ensemble d'entités appelées agents, coopérant entre eux aux fins d'exécution d'une activité dite "intelligente" ... l'intelligence du système se trouve distribuée à travers les différents agents du système. »

Munindar Singh (1991, 1993, 1994), qui a publié de nombreux travaux portant sur les mécanismes et langages de communication inter agents, soutient que

l'attrait de ces systèmes vient non seulement du fait qu'ils constituent des systèmes distribués, mais aussi parce qu'ils vont permettre le développement de systèmes intelligents de façon indépendante les uns des autres, ces systèmes pouvant être réutilisés comme composantes de systèmes plus complexes. La définition des systèmes multi-agents données par Singh complète celle de Benslimane en mentionnant que « *chacune des tâches identifiées dans le système est confiée à un agent, les agents devant collaborer afin d'exécuter la tâche globale à laquelle le système est dédié* » (Singh, 1993). Toujours selon cet auteur, les systèmes bâtis selon la philosophie orientée agent seront plus robustes que les systèmes intelligents traditionnels, puisque leur mise en place et leur validation seront plus simples de par leur modularité, plus simples aussi parce que *l'agent intelligent pourra être situé sur le même site que les données qu'il utilise et où les décisions doivent être prises* (Singh, 1994). De plus, contrairement aux composantes des systèmes distribués non intelligents, « *les parties utilisées dans les systèmes multi-agents (les résolveurs) sont explicitement conscients de la distribution des composants et peuvent même prendre des décisions sur la base de cette information* » (Chaib-draa, 1993).

Une société d'agents peut être soit ouverte, soit fermée. On dit qu'une société est fermée si l'organisation de cette société, c'est-à-dire les agents qui la composent et les interrelations qui existent entre ces agents, est connue au moment de la conception du système informatique et demeure inchangée lors de chaque utilisation de ce système. Il

s'agit donc d'équipes de travail permanentes. Les sociétés ouvertes, quant à elles, sont comparables à des comités ad hoc. Elles sont composées dynamiquement à partir de librairies d'agents (Lander, 1994) et ce pour créer une application répondant aux besoins du moment, que ce soit ceux d'une autre application ou encore d'un utilisateur. L'élaboration d'un assistant intelligent pour la recherche d'information, offrant la souplesse présentée au chapitre précédent, s'apparente donc à une société d'agents ouverte.

Une société d'agents est aussi soit homogène, soit hétérogène. On dit d'une société qu'elle est homogène lorsque tous ses agents utilisent le même langage de communication, la même technique de représentation de leurs connaissances basée sur une syntaxe et une sémantique communes, et s'entendent sur une technique de contrôle des activités du groupe. À l'inverse, un système multi-agents est dit hétérogène si ses agents utilisent des langages de communication et des méthodes de représentation de connaissances dont les syntaxes ou sémantiques varient, ou encore s'ils sont basés sur des techniques de contrôle non uniformes. Bien que leur degré de complexité puissent grandement varier de l'une à l'autre, les sociétés homogènes sont de loin moins complexes et beaucoup plus faciles à réaliser que les sociétés hétérogènes. Mentionnons d'ailleurs que les problèmes liés aux sociétés homogènes constituent un sous-ensemble de ceux inhérents aux sociétés hétérogènes.

2.3.3 Agents et systèmes adaptatifs

Basés sur la modularité et la réutilisation de leurs composantes, les systèmes multi-agents présentent aussi un autre grand intérêt : ils facilitent l'adaptation d'un système informatique face à ses différents utilisateurs. Mentionnons ici quelques concepts clefs reliant les systèmes multi-agents et les systèmes adaptatifs :

- un agent capable de raisonner sur autrui (d'autres agents) sera aussi capable de raisonner sur les services qu'il est susceptible de rendre à un utilisateur humain; cette machine, capable de raisonner intelligemment, pourra aller jusqu'à maximiser sa fonction d'utilité vis-à-vis de son utilisateur (Chaib-draa, 1993) ;
- un agent capable d'apprentissage peut adapter son comportement au profil de son utilisateur (besoins et préférences);
- un tel agent, capable de raisonnement et d'apprentissage, peut devenir un outil très efficace dans les fonctions d'aide et de guidage offertes par un logiciel à ses utilisateurs.

2.3.4 Systèmes multi-agents hétérogènes

Si la souplesse des systèmes multi-agents hétérogènes permet de répondre à un plus vaste éventail de problèmes et sous-problèmes que les systèmes monolithiques ou multi-agents homogènes, leur caractère hétérogène augmente cependant la complexité de la solution élaborée. Les systèmes multi-agents hétérogènes comportent donc avantages et désavantages directement reliés à leur caractère modulaire et hétérogène.

2.3.4.1 Avantages et désavantages

Les principaux attraits d'une modélisation par agents sont fortement reliés aux avantages présentés par l'intelligence artificielle distribuée, ainsi qu'aux systèmes distribués de façon générale. En voici quelques-uns (Singh, 1994; Chaib-draa, 1993) :

- simplifier la modélisation de systèmes complexes en ramenant de tels systèmes à un ensemble de sous-systèmes pouvant être développés de façon indépendante et comportant une complexité individuelle inférieure à celle du problème global;
- rendre modulaire l'implantation d'un système en sous-systèmes de façon à accroître le potentiel de réutilisation de ces sous-systèmes;
- faciliter la localisation d'un processus intelligent sur le site où les données d'intérêt sont situées.

Susan Lander (1992, 1994), qui complétait récemment une thèse de doctorat sur la possibilité de créer des systèmes multi-agents hétérogènes, cible bien les avantages et problèmes reliés à la philosophie de ce type de systèmes. Parmi les avantages, elle mentionne :

- réutilisation d'agents : les agents étant des entités de traitement complètes qui se spécialisent dans un type d'expertise donné, ils peuvent être intégrés dans différents systèmes nécessitant l'accès à ce type d'expertise;
- faible coût : les systèmes composés d'agents hétérogènes ont le potentiel d'être adaptatifs, facilement modifiables et peu coûteux une fois les problèmes de partage d'information, de coordination et de résolution de conflits résolus;

- création dynamique des équipes de travail : ces systèmes sont comparables à des équipes de spécialistes humains dans lesquelles les agents experts sont réutilisables et peuvent être sélectionnés dynamiquement à partir de librairies et intégrés dans le groupe de travail moyennant un coup minimal d'implantation;
- adaptation dynamique de l'équipe en cours de processus : les agents peuvent être ajoutés ou enlevés efficacement et à faible coût à un système en cours d'utilisation dont les spécifications ou ressources ont été modifiées;
- augmentation de la performance avec le temps : tout comme les spécialistes humains, les agents apprennent avec le temps et l'expérience de travail, devenant ainsi plus performants; le coût d'ingénierie d'un agent s'en trouve donc amorti;
- recherche distribuée de solution : les agents travaillent en parallèle à la recherche d'une solution au problème global traité par l'équipe; puisque les agents possèdent des connaissances et savoir-faire variés, il en résulte une recherche plus complète de l'espace solution que dans le cas d'un système à base de connaissances centralisée.

Bien que ces avantages soient indéniables, ils ne sont pas sans problèmes et soulèvent même certaines questions qui ne semblent pas encore résolues. En effet, Lander et Lesser (1994) mentionnent :

- capacité à communiquer : malgré leur hétérogénéité, les agents doivent être en mesure de se transmettre de l'information au sujet des progrès réalisés dans la résolution du problème global, de coordonner leurs actions, de résoudre les conflits

découlant d'information incomplètes ou non consistantes et de se doter de critères d'évaluation de l'information ;

- capacité d'interprétation de l'information : les agents doivent non seulement être en mesure de recevoir de l'information, mais aussi être munis du savoir-faire leur permettant d'utiliser cette information à bon escient;
- compromis entre partage et complexité d'information : plus l'information à partager est complexe et dynamique, plus le coût de partage de cette information est élevé; le compromis s'exprime en termes d'amélioration de la performance si les agents sont bien informés, par opposition à une dégradation de la performance inhérente à la surcharge de communication;
- complexité de l'intégration des agents : la création d'agents pouvant se comprendre suffisamment bien pour arriver à s'entraider dans leurs tâches respectives est hautement complexe;
- minimisation des restrictions imposées à la conception : les restrictions imposées lors de la conception des agents doivent être gardées minimales de façon à restreindre les besoins de coordination entre les concepteurs d'agents; ainsi, on peut imposer aux concepteurs d'agents des contraintes quant au langage de communication à utiliser pour interagir avec les agents de l'environnement, mais on doit éviter de forcer une méthodologie de représentation et de manipulation de connaissances à l'intérieur des agents et laisser cette décision aux concepteurs afin que le choix soit effectué en fonction des tâches de l'agent ;

- optimisation de la distribution des rôles : l'assignation des sous-tâches entre les agents doit être guidée par l'efficacité de la société résultante dans la résolution du problème visé, et cette assignation doit être revue et optimisée au besoin en cours de processus; dans cette optique, Chaib-draa (1993) mentionne que l'organisation d'une société doit permettre une assignation *optimale*, *consistante* et *complète* des différentes sous-tâches du problème global.

2.3.4.2 Sources d'hétérogénéité

Les avantages cités par Lander et Lesser (1994) impliquent donc que les systèmes multi-agents ouverts et hétérogènes permettent de créer sur mesure des systèmes informatiques complexes par la juxtaposition d'éléments intelligents relativement simples créés par différents concepteurs. Il suffirait alors de mettre ces éléments, les agents intelligents, en communication de façon à leur permettre d'échanger leurs connaissances respectives et de s'échanger des services afin d'atteindre le ou les buts qu'ils se sont fixés, ou qu'on leur a fixés. De cette façon, les agents seraient réutilisables, les copies d'un même agent pourraient évoluer différemment selon leur contexte d'utilisation, et on pourrait même voir des agents novices (nouvellement créés) profiter du savoir-faire d'agents plus expérimentés (Maes 1994; Arcand et Pelletier, 1995).

Cependant, comme la liste des désavantages l'indique, les problèmes reliés à la nature hétérogène même de ces systèmes sont relativement complexes, principalement en ce qui a trait à la communication entre les agents. Mais, avant de s'attaquer à ce

problème de communication, il est important de bien cerner les bases de l'hétérogénéité présente entre les agents.

Edmonds (1994) identifie 3 types d'hétérogénéité : *syntaxique, sémantique* et *de contrôle*. L'hétérogénéité de syntaxe fait référence au fait que les agents peuvent utiliser différentes techniques de représentation de connaissances (par exemple, réseau de neurones et système expert). L'hétérogénéité sémantique signifie que la même information peut prendre une signification différente d'un agent à l'autre. Finalement, l'hétérogénéité de contrôle implique que de la confusion peut s'installer au sein de la société si les agents utilisent des mécanismes différents pour inférer les connaissances en groupe. En résumé, l'hétérogénéité s'installe lorsque les agents n'ont pas le même langage ou le même schème de pensée.

Cette hétérogénéité est due au fait que les agents sont créés par des individus différents et travaillant principalement de façon indépendante (Rosenchein, 1994). Puisqu'il n'y a pas encore de normes établies quant à la façon dont les agents doivent communiquer et interpréter l'information collective (c'est-à-dire non déduite par l'agent lui-même mais plutôt produite par la société), la création d'une société d'agents à partir de tels agents résulte en un véritable casse-tête d'intégration. De plus, il devient alors presque impossible de faire une réingénierie partielle d'une telle société d'agents (Singh, 1994).

Le fait que ces problèmes soient complexes à résoudre n'implique pas qu'une homogénéisation complète entre les agents soit souhaitable. En effet, bien que

l'homogénéité au niveau du langage de communication inter-agents soit grandement souhaitable et probablement réalisable (le langage de communication KQML décrit dans la section 2.3.6.2 et à l'Annexe 1 en est un exemple), l'homogénéité dans la technique de représentation des connaissances utilisées par les agents deviendrait vite trop restrictive. Ceci se justifie par le fait que, à ma connaissance, aucune des techniques de représentation de connaissances, qu'elle soit de nature symboliste, connexioniste ou traditionnelle (structure de données et programmation procédurale, objets, etc.), n'a encore été prouvée comme étant optimale pour l'ensemble des tâches habituellement traitées par ces techniques. Ainsi, si plusieurs techniques symbolistes sont reconnues pour la qualité des explications qu'elles peuvent fournir quant au raisonnement relié à une décision, ces techniques peuvent difficilement permettre de généraliser les connaissances et d'en acquérir de nouvelles. D'un autre côté, les techniques connexionistes ont de fortes capacités de généralisation et d'apprentissage, mais peuvent difficilement expliquer le détail de leur processus de raisonnement. Il en résulte que la technique de représentation de connaissances d'un agent doit être choisie explicitement en fonction de la tâche à compléter et non en fonction de l'environnement dans lequel cet agent peut être utilisé. Ce raisonnement se trouve d'autant plus renforcé si l'on considère un agent comme pouvant travailler à l'intérieur de sociétés différentes.

Les agents doivent donc être vus comme des boîtes noires pouvant mettre à profit leur savoir-faire dans le cadre d'une société. Ces boîtes noires doivent aussi posséder un ensemble de connaissances sur le(s) langages(s) et mécanisme(s) à utiliser pour échanger

avec les autres agents. La section 2.3.6 tentera de mettre en valeur certaines des stratégies qui ont été utilisées concernant l'établissement de ces langages et mécanismes de communication. Mais tout d'abord, faisons un bref survol de travaux réalisés concernant les architectures de systèmes multi-agents.

2.3.5 Architectures de systèmes multi-agents

La définition d'une architecture d'agents peut être située à trois niveaux : typologie et structuration; communication; coopération (Benslimane, 1993). Le premier niveau vise à définir les agents et à faire une analyse de l'activité à être implantée par les agents en complétant la liste d'agents nécessaires ainsi que la structure de leur organisation. La communication détermine comment s'effectuent les échanges entre les agents. Finalement, la coopération établit le(s) mode(s) de négociation et de coopération utilisé(s) entre les agents. À notre connaissance, il semble qu'à ce jour il n'existe aucune architecture comportant tous ces niveaux et pouvant servir d'architecture multi-agents universelle. Cependant, de nombreux travaux, théoriques ou pratiques, ont été complétés en ce sens. Ce qui suit constitue un bref survol de recherches effectuées sur ces architectures. D'autres travaux sur les architectures multi-agents développées dans le domaine de la recherche intelligente d'information sont présentés à l'Annexe E.

Wooldridge

Selon Wooldridge (1994), une théorie formelle sur les systèmes multi-agents devrait intégrer les rôles suivants :

- fournir un outil de spécification pour décrire et utiliser les systèmes multi-agents;
- fournir un outil de spécification et (à la limite) de vérification des propriétés de systèmes multi-agents;
- fournir une base sur laquelle des théories plus complètes d'activités de sociétés, d'interaction et de coopération pourraient être développées.

En somme, une telle théorie tient plus de la définition d'un langage permettant la définition d'agents et de systèmes multi-agents que la définition de l'architecture multi-agents proprement dit.

Lander et Lesser

Lander et Lesser (1994), quant à eux, définissent la notion d'architecture de systèmes multi-agents en termes plus concrets. Ils spécifient, entre autres, que cette architecture doit :

- en être une qui supporte l'intégration d'agents hétérogènes et réutilisables, et doit fournir un algorithme générique pour la résolution distribuée et en collaboration de problèmes entre les agents;
- fournir des procédures d'ajout et de retrait d'agents d'une société (c'est-à-dire permettre la création de sociétés ouvertes);
- intégrer une stratégie de résolution de problèmes connue par tous les agents et, pour chaque tâche découlant du problème, avoir accès à au moins un agent possédant l'expertise pour compléter cette tâche (distribution complète des tâches);

- permettre la composition d'ensembles d'agents (les sociétés) regroupant les expertises nécessaires à la résolution du problème, sans que les agents aient à posséder des connaissances à priori sur les autres agents de l'ensemble, c'est-à-dire connaître tous les agents avant qu'une collaboration devienne nécessaires, ainsi que sur les compétences de ceux-ci;
- comprendre une stratégie de résolution de conflits.

Lander et Lesser (1992, 1994) soutiennent que l'architecture développée dans le cadre du projet TEAM atteint l'ensemble des buts ci-avant mentionnés. Afin de permettre une pleine autonomie des agents, ceux-ci communiquent par l'entremise d'une mémoire commune semblable à la technique de tableau noir proposée par Nii (1993) et expliquée dans la section 2.3.6.2. Cette mémoire commune permet de départager l'information propre à chaque agent, celle disponible à l'ensemble des agents, ainsi que l'information du contrôleur de la société. De cette façon, les agents communiquent par le biais d'un intermédiaire et n'ont pas à se soucier du détail d'implantation des autres agents.

L'architecture est basée sur deux types de cycles : cycle des agents et cycle du contrôleur. Dans le cadre d'un cycle des agents, chaque agent utilise l'information de la mémoire commune pour déterminer quelles stratégies de recherche et de négociation il est en mesure d'appliquer. Il ajoute alors ces stratégies à son agenda personnel, puis applique la stratégie de plus haute priorité et retourne le résultat, s'il y a lieu. Ce cycle est imposé à chacun des agents de la société.

Lors de son cycle, le contrôleur fait une mise à jour de la mémoire commune en se basant sur les messages reçus des agents, puis propage les effets de ces changements sur les objets de la mémoire commune. Le contrôleur est donc un facilitateur d'échange d'information et non un contrôleur proprement dit des activités de négociation des agents. Le contrôle demeure décentralisé, ce qui répond aux buts de modularité et d'autonomie des agents.

Bien que cette architecture semble facile à implanter à première vue, elle ne résout pas le problème inhérent à ce type de communication centralisée, c'est-à-dire le goulot d'étranglement que peut devenir le contrôleur de communication.

Maes

Les travaux de la chercheure Patty Maes du MIT sont surtout orientés vers la définition d'agents devant interagir directement avec les utilisateurs et, plus spécifiquement, sur ce qu'elle appelle les « *believable agents* ». Maes (1994) propose de définir une architecture multi-agents en terme de méthodologie (la société est ici vue comme un agent, les agents composant la société sont appelés modules) :

> *"Une méthodologie particulière pour construire des agents. Elle détermine comment les agents peuvent être décomposés pour la construction d'un ensemble de modules et comment ces modules doivent interagir. L'ensemble final des modules et de leurs interactions doit fournir une méthode pour définir de quelle façon les données des capteurs et l'état interne des agents déterminent les actions et les états futurs des agents. Une architecture doit*

comprendre les techniques et les algorithmes devant supporter cette méthodologie." (Wooldridge, 1994)

Maes identifie aussi un agent comme comprenant un ensemble de modules de compétence ou noeuds de connaissances, chaque module étant défini en termes de pré et post conditions, et ayant un niveau d'activation associé. Les pré et post conditions relient les modules de façon à créer l'équivalent d'un réseau logique, et le niveau d'activation (un nombre réel) exprime la valeur de pertinence d'un module dans une situation particulière (voir (Arcand et Pelletier, 1995) pour un autre exemple d'utilisation du niveau de pertinence). Ce type d'architecture s'inspire grandement des architectures de réseaux de neurones avec contrôle totalement distribué.

Brocks

Brooks propose principalement une architecture qu'il qualifie de *réactive*, c'est-à-dire qui ne nécessite pas un contrôle central de type symboliste. Il soutient aussi que, comme nous l'avons mentionné à la section 2.3.4.2, l'intelligence des agents n'a pas à être basée seulement sur l'utilisation de techniques de représentation symboliste de connaissances. Selon lui, l'intelligence d'un système ne prend pas sa source dans la représentation que les agents se font des connaissances, mais plutôt dans les interactions qui surviennent entre les agents. L'intelligence est donc une propriété émergente de certains systèmes complexes, puisqu'elle ne réside pas dans les agents individuels mais plutôt dans le monde composé de ces agents. Brooks (1991) utilisent deux termes pour caractériser cette théorie : *situatedness* et *embodiement*.

L'architecture proposée par Brooks (1991), une architecture englobante, est composée d'une hiérarchie de comportements. Par architecture englobante, on sous-entend que chaque comportement, à l'exception des comportements de dernier niveau, sont en fait des regroupements de comportements des niveaux inférieurs. Les comportements sont donc définis de façon récursive. Les comportements, c'est-à-dire les agents, entrent en compétition pour prendre le contrôle (le contrôle s'exerce sur un robot dans l'exemple présenté par Brooks). Les agents de plus bas niveaux représentent des comportements plus primitifs et ont préséance sur les agents de niveaux supérieurs. La communication s'effectue par le biais de la structure hiérarchique. Cette architecture s'apparente à l'idée proposée par Chapman (Wooldridge, 1994) selon laquelle la majorité des décisions tiennent de la routine et devraient être encodées dans les structures de bas niveau de la hiérarchie, en permettant seulement des mises à jour périodiques du reste de la hiérarchie.

Shoham

L'un des premiers chercheurs à développer une architecture formelle pour les systèmes multi-agents fut Yoav Shoham (1993), bien connu pour son modèle de programmation orienté agents (*Agent oriented programming : AOP*). Ce modèle est fortement basé sur les notions d'intentions (désir de compléter une action), ainsi que sur des techniques de logique utilisées en intelligence artificielle symboliste. En effet, l'idée principale de ce modèle est de programmer les agents directement en termes de composantes mentales, c'est-à-dire les connaissances de l'agent, et d'intentions, soit les

activités que l'agent désire compléter sur ses connaissances. Son modèle comprend trois principales parties : un système de logique pour définir les composantes mentales des agents (avec syntaxe et sémantique précises); un interpréteur pour la programmation des agents (comportant des primitives telles que *REQUEST* et *INFORM*); un module d'« agentification » qui permet de compiler les agents et d'en faire des systèmes exécutables de bas niveau ainsi que de convertir différents processus informatiques en des agents programmables.

Shoham explique que son modèle peut être vu comme une spécialisation de la programmation orientée objet. La spécialisation prend forme dans l'utilisation de la théorie des actes du langage (Singh, 1991). Cette théorie propose de caractériser les échanges entre les agents en termes de type de message selon un modèle exprimé par les linguistes et les philosophes portant sur les types de messages échangées entre les humains. Les types de messages sont définis selon la force qui leur est associée, pouvant passer de l'*assertion*, soit l'émission d'information, à la *directive*, soit l'émission d'un ordre. Ainsi, les agents se distinguent des objets par les suppositions qu'ils sont capables de faire à partir de leurs connaissances et qui les conduisent à échanger des messages de différents types avec d'autres agents, de façon à provoquer des effets précis dans l'environnement. Les objets non munis d'intelligence n'ont pas cette capacité d'introspection qu'ont les agents. Chaque fois qu'un agents effectue un cycle d'activité, il complète deux étapes : il lit le message courant et fait la mise à jour de son état mental; il exécute ensuite les engagements pris au moment présent.

Urzelai

Dans le cadre de son projet MAKILA, Urzelai (1992) a bâti un outil pour le design et l'implantation de sociétés composées d'agents coopératifs. Ces sociétés d'agents sont orientées vers la résolution de problèmes (*goal-driven*). Les agents communiquent par l'entremise d'une mémoire centrale, soit un tableau noir tel que présenté à la section 2.3.6.2. Un peu à la manière du modèle proposé par Lander et Lesser (1994), le contrôle est distribué parmi les agents, en ce sens que ces derniers utilisent l'information affichée sur le tableau noir pour mettre à jour leurs connaissances internes et ainsi décider des actions à exécuter. Chaque agent est responsable d'afficher ses informations et de lire celles qui sont susceptibles de l'intéresser. Comme dans tous les modèles vus précédemment, chaque agent est spécialisé dans la résolution d'une classe de problèmes.

Deux types d'agents sont utilisés dans cette architecture : les agents internes (mise en place de la société comme par exemple l'agent responsable d'éliminer l'interblocage) et les agents définis par l'utilisateur. Lors d'un cycle d'activité, chaque agent complète deux types de tâches : celles reliées à la résolution du problème, c'est-à-dire l'exécution des actions planifiées, et celles de gestion de la communication, c'est-à-dire l'assimilation et l'affichage d'information.

Selon Urzelai, l'architecture de MAKILA permet une organisation dynamique des sociétés en fonction des problèmes à résoudre et supporte donc la création de sociétés ouvertes. Un système créé en suivant ce modèle est en mesure de réagir rapidement et à un coût minimal à la disparition ou à la création d'un agent. Le seul coût associé au

retrait d'un agent est celui du temps nécessaire pour trouver un autre agent capable de résoudre la même classe de problèmes.

General Magic

Le premier langage commercial pour la définition de sociétés d'agents est apparu sur le marché en 1994. Il s'agit de *Telescript*, commercialisé et maintenant distribué gratuitement par *General Magic*, une compagnie californienne lancée à l'origine par le géant Apple. Telescript constitue un langage de communication pour les ordinateurs d'un réseau de télécommunications et permet le passage d'exécutables d'un ordinateur à l'autre, indépendamment des plates-formes communicantes. Tout ceci est implanté grâce au système d'exploitation *Magic Cap*, lequel s'occupe d'assurer la transparence voulue. General Magic offre aussi un outil de développement d'agents.

Dans ce contexte, l'agent est considéré comme un programme exécutable pouvant s'exécuter n'importe où sur le réseau. On l'appelle agent parce qu'il est une entité agissant sur un ordinateur au nom d'un autre ordinateur (Davis, 1994). L'idée qui sous-tend Telescript est de permettre le développement, par des compagnies commerciales, d'agents offrant des services spécifiques et distribués auprès du grand public. Des exemples de tels produits pourraient être un agent de surveillance de l'arrivée de vols à un aéroport donné, ou encore un agent capable de magasiner des articles en comparant des listes de prix disponibles sur le réseau. Telescript vise donc la commercialisation d'agents mobiles et spécialisés de type assistants personnels.

2.3.6 Communication entre agents

Les agents, puisqu'ils ne sont pas gérés par une structure centrale, ne disposent pas automatiquement d'un mécanisme leur permettant d'échanger information et contrôle d'exécution. Il devient donc nécessaire d'élaborer des méthodes, qu'elles soient implicites ou explicites, permettant ces échanges afin qu'une société d'agents puisse atteindre des buts fixés. Cette section d'abord justifie le besoin d'un langage commun, puis présente deux techniques de communication inter-agents proposées par les chercheurs du domaine, l'une implicite (le tableau noir), et l'autre explicite (l'échange de messages).

2.3.6.1 Nécessité d'un langage commun

Afin de pouvoir collaborer, les agents hétérogènes doivent avoir accès à des techniques leurs permettant de communiquer. Cette communication vise l'échange de données, d'information logiques ou de commandes (Genesereth, 1994). Pour que cette communication soit efficace, la signification d'un message doit être la même pour l'ensemble des agents, contrairement à ce que l'on retrouve dans la programmation orientée objet dans laquelle un message peut être interprété différemment d'un objet à l'autre. Pour ce faire, il faut élaborer un langage commun à l'ensemble des agents. L'implantation de ce langage ne doit cependant pas faire partie intégrante de l'élaboration de chacun des agents, mais plutôt être vue comme la mise en place d'une interface entre les agents ayant à interagir. On peut aussi qualifier ce langage de protocole de communication inter-agents. Grâce à ce langage, les agents sont en mesure

de baser leurs messages sur une syntaxe et une sémantique communes, même si ce langage ne constitue pas la « langue première » de ces agents.

Bien que la communication dans les systèmes multi-agents puisse être mise en place de façon procédurale, c'est-à-dire par l'échange d'exécutables autonomes (cette technique est utilisée par Telescript), la majorité des travaux de recherche s'intéressant à la communication inter-agents portent sur la communication par messages déclaratifs. Dans ce cas, les agents émettent des messages dont la syntaxe et la sémantique sont bien précises. Ces messages sont d'ailleurs souvent basés sur deux composantes : une étiquette qui donne le type du message, et le contenu proprement dit du message.

2.3.6.2 Communication par messages déclaratifs

Deux grandes méthodes déclaratives de communication inter-agents ont été étudiées par les chercheurs : communication par tableau noir (blackboard) (Nii, 1993) et communication par messages (Chaib-draa, 1993; Wooldridge, 1992). Ces deux méthodes sont décrites ci-après.

Communication par tableau noir

La communication par tableau noir, développée par Nii (1993), consiste principalement à permettre aux agents l'accès à une structure de données commune. Le tableau noir devient alors une source d'information, de résultats partiels, d'hypothèses pour chacun des agents. Dans ce type de communication, les échanges d'information entre les agents ne s'effectuent qu'au moyen des connaissances et hypothèses affichées

sur le tableau. "Par ce moyen, les agents se partagent les solutions partielles ... " (Chaib-draa, 1993). Un système d'agenda rattaché au tableau noir s'occupe de planifier l'ordre des tâches des agents, ainsi que d'évoquer leur exécution. Deux sources d'information sont alors disponibles aux agents, soit leur base de connaissances privée ou locale, et la base de connaissances publique du tableau. On parle donc de distribution partielle des connaissances.

Le tableau noir, bien que peu complexe à implanter, comporte une faiblesse indéniable : il peut vite devenir un *goulot d'étranglement* lorsque plusieurs agents tentent d'accéder aux informations communes. En effet, ce tableau, comme toute structure de donnée centralisée, ne peut être accédée en écriture simultanément par plusieurs agents sans risquer de corrompre l'intégrité des données. Un agent accédant le tableau noir en écriture bloquera donc les activités des autres agents jusqu'à ce qu'il ait terminé l'écriture. Ce contrôle du tableau noir s'effectue habituellement par l'utilisation de sémaphores qui donne l'exclusivité d'utilisation du tableau noir à un agent ayant besoin d'y accéder en écriture. Il est facile de constater que si les informations affichées sur le tableau noir sont très dynamiques, et donc que le tableau noir se trouve fréquemment monopolisé par un agent, les autres agents passeront beaucoup de temps à attendre, ce qui les ralentira dans leurs activités.

Par contre, ce même tableau comporte deux avantages importants : *uniformisation de la communication entre les agents* par imposition d'un langage commun *et indépendance de chaque agent face à ses interlocuteurs*. En effet, l'utilisation du tableau

noir uniformise la communication entre les agents hétérogènes puisque tout agent connaissant le langage de communication du tableau est en mesure de communiquer avec tous les autres agents, indépendamment des techniques de représentation de connaissances internes utilisées par chacun. De plus, l'agent ayant de nouvelles informations à transmettre à la société n'a pas à connaître ses interlocuteurs puisque le transfert des messages est géré par le tableau noir. Il suffit donc à l'agent de connaître l'adresse du tableau noir pour partager ses informations avec les autres agents. Ceci peut représenter un grand avantage lorsque l'organisation de la société d'agents est très dynamique.

Dans le but de centraliser les activités des agents autour du tableau noir, les modèles classiques de tableau noir, tel que le modèle *BB1* montré à la Figure 2.2, intègrent un module de contrôle décomposé en différentes parties : gestion des zones de connaissances, agenda des activités du groupe d'agents reliés au tableau noir et planificateur des activités.

La gestion des connaissances s'effectue par l'entremise d'un tableau noir sur lequel les agents ayant la connaissance du domaine échangent leurs informations. L'agenda des activités des agents, et par conséquent l'agenda du groupe d'agents, est contrôlé par des agents internes spécialisés dans cette tâche. Ces agents de contrôle analysent l'information affichée sur le tableau noir des connaissances ainsi que le contenu courant de l'agenda du groupe et utilisent différentes heuristiques pour déterminer 3 niveaux d'activités. Le premier niveau, les stratégies, représentent la planification à long terme

du groupe d'agents. Ces stratégies sont itérativement subdivisées en sous-stratégies. Le deuxième niveau, les focus, sont les noeuds terminaux des stratégies et forment les buts des agents. Finalement, le troisième niveau est composé des heuristiques qui accompagnent chaque focus.

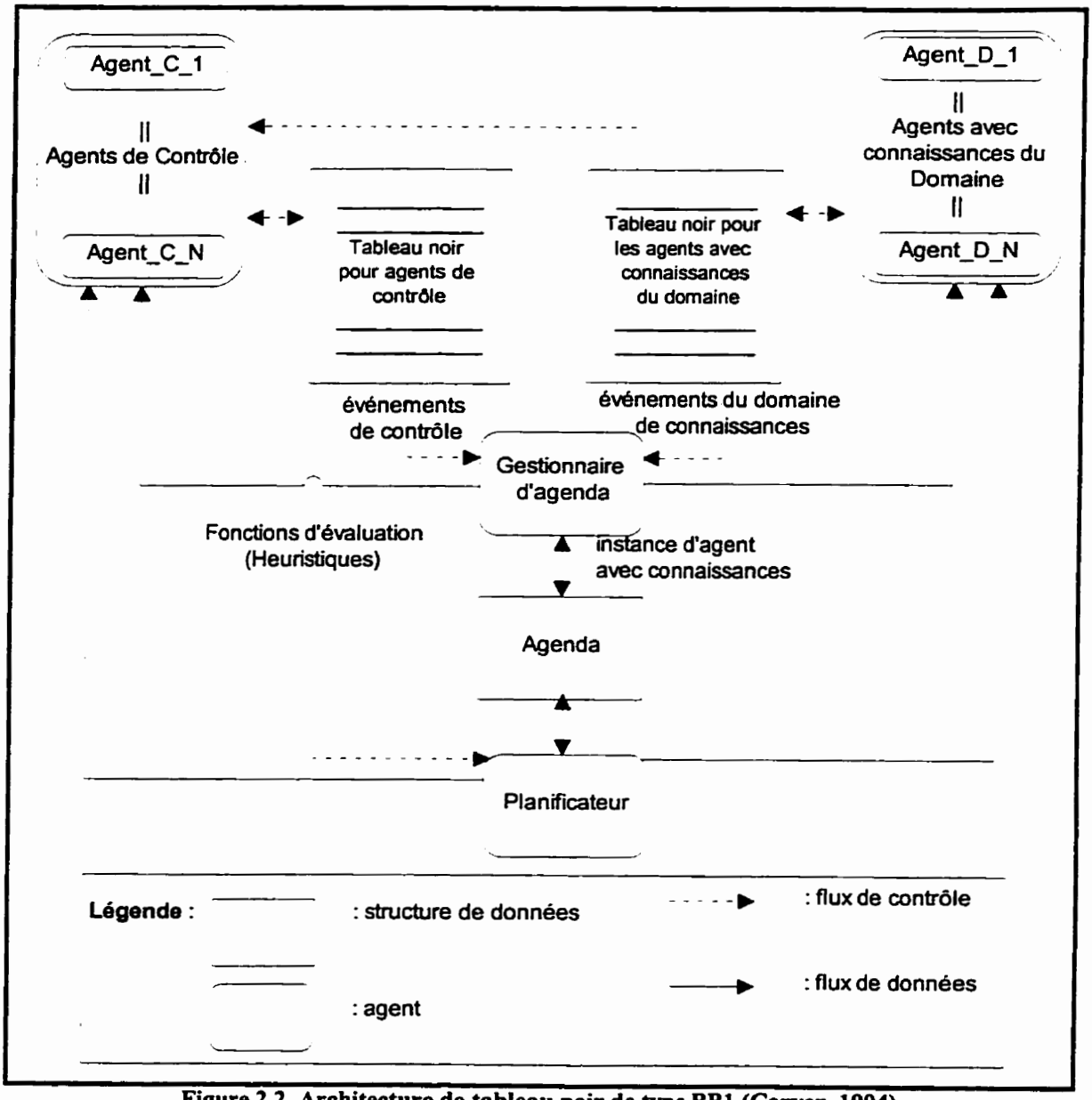


Figure 2.2. Architecture de tableau noir de type BB1 (Carver, 1994)

Un tableau noir est mis à la disposition des agents de contrôle afin qu'ils puissent échanger leurs connaissances de planification d'activités. Une fois déterminées par les agents de contrôle, les activités sont inscrites à l'agenda global et communiquées au planificateur d'exécution des activités, lequel s'assurera d'invoquer les agents de connaissances ou de contrôle au moment opportun afin qu'ils participent à l'atteinte des buts visés. Un outil permettant l'implantation de tableaux noirs selon l'architecture BB1 est d'ailleurs disponible sur le marché (*GBB*, distribué par Blackboard Technology Group, Inc.).

Plusieurs variantes de la communication par tableau noir ont été proposées. L'une des plus populaires est utilisée dans les sociétés d'agents à organisation hiérarchique telles que celles proposées par Brooks. Il s'agit alors de munir chaque noeud composite de la hiérarchie, une société, d'un tableau noir. Ce tableau noir permet aux membres de la société de communiquer entre eux. De cette façon, l'information est véhiculée entre les noeuds d'un même niveau. De plus, ce tableau noir, ou encore un agent spécialisé en la matière, filtre les informations affichées sur le tableau noir afin de parvenir au tableau noir du niveau supérieur ainsi qu'aux tableaux des niveaux inférieurs les éléments susceptibles d'intéresser, assurant alors la communication d'un niveau à l'autre de la hiérarchie. Les grands avantages de cette structure de communication sont qu'elle permet une division efficace de la tâche entre les agents, tout en *diminuant les possibilités de goulot d'étranglement*, chaque tableau noir étant rattaché à un nombre restreint d'agents.

Un exemple d'utilisation de la hiérarchie de tableau noir peut être trouvé dans (Dai, 1993).

Communication par messages

La seconde méthode, qui s'appuie sur une distribution totale des connaissances, consiste en la communication par messages. Très près de la communication telle qu'implantée en programmation par objet, la communication par messages s'effectue par l'échange direct d'informations et de commandes entre les agents. Les agents n'ont alors qu'une base de connaissances locale, les connaissances des autres agents leur parvenant aux moyens des messages échangés. Les connaissances sont donc totalement distribuées. Dans ce type de communication, les agents doivent savoir avec qui et quand entrer en communication. Parmi les avantages que comporte cette méthode de communication, citons : *permettre aux agents d'être plus autonomes*, *d'exercer plus facilement le contrôle sur leurs échanges d'information* et *permettre une meilleure distribution du contrôle des activités de la société*. En effet, puisque les agents échangent directement leurs messages, ils contrôlent à qui ils font parvenir leurs connaissances, à qui ils acceptent de rendre service, etc. Cette méthode de communication évite aussi le problème de goulot d'étranglement inhérent au tableau noir.

Cependant, tout comme le tableau noir, la communication par messages ne vient pas sans ses désavantages dont : *risques d'ambiguïté ou d'impossibilité à utiliser le message lors de l'interprétation par l'agent destinataire*, *augmentation du trafic entre les agents* et *nécessité pour chaque agent de connaître l'adresse de ses interlocuteurs*. Bien que

l'augmentation du trafic soit peu contrôlable, il est possible de réduire les risques d'ambiguïté face au langage de communication en publiant des langages de communication dont les syntaxes et sémantiques sont connues et peu ambiguës, et en basant la communication entre les agents sur ces langages connus. De même, certaines architectures multi-agents facilitent le passage de messages entre agents en leur offrant les services d'un facilitateur de communication. Ce facilitateur peut être vu comme un agent connaissant l'adresse de communication de chaque agent. Lorsqu'il désire faire parvenir un message à un agent destinataire, l'agent expéditeur peut faire appel au facilitateur de communication pour obtenir l'adresse courante du destinataire, lequel donne la dernière adresse connue du destinataire à l'agent en faisant la requête, puis l'agent expédie son message à cette adresse.

Les langages de communication développés pour fins de communication entre agents autonomes sont appelés les langages de communication inter-agents. Parmi les langages en cours de développement dans le monde, KQML (Knowledge Query and Manipulation Language) (Finin, 1993; Cohen 1995; Mayfield, 1995) s'affiche de plus en plus comme le standard à suivre. Développé par une équipe de chercheurs de l'université de Maryland, KQML a déjà été utilisé dans le cadre de dizaines de projets multi-agents à travers le monde, dont deux de grande envergure : ARCHON en Angleterre (Jennings, 1995) et UMDL aux États-Unis (Digital Library Project Research Proposal, 1995). Il semble d'ailleurs que l'enthousiasme pour KQML ne fait qu'augmenter avec le temps. Cette popularité est due à la fois à l'avancement de ce langage par rapport à ce qui existe

présentement dans le domaine, à la facilité avec laquelle l'information sur le sujet peut être obtenue via Internet, et aussi due au fait que ce langage est le fruit de chercheurs de haut calibre dans le domaine des systèmes multi-agents.

Le but premier de KQML, comme celui de tout langage de communication inter-agents, est d'offrir la flexibilité et l'ordre de grandeur nécessaires aux nouveaux réseaux d'information de façon à faciliter l'intégration de la technologie multi-agents distribuée sur ces réseaux. Un message tel que défini dans KQML est un énoncé performatif, c'est-à-dire un message comprenant à la fois information et demande d'action, puisque l'envoi même du message est censé faire déclencher une action dans le système. Il s'agit en fait d'un message enveloppant à la fois l'*information* sur laquelle porte le message, le *contexte de communication* (expéditeur, destinataire, demande de réponse, etc.) et l'*intention de l'expéditeur* (l'énoncé performatif lui-même). Un certain nombre d'énoncés performatifs font partie du langage et chaque agent reconnaît un sous-ensemble de ces énoncés. Chacun des énoncés est identifié à l'aide d'un mot réservé de façon à ce que les agents puissent le reconnaître *: tell, untell, reply, deny, ask-about*, etc. Bien que les agents puissent choisir quels énoncés ils désirent supporter, ils doivent cependant les implanter de façon standard. Ces énoncés ont été classifiés par les auteurs de KQML selon différentes catégories : *énoncés de base* (tell, deny, untell), *énoncés de systèmes de gestion de bases de données* (insert, delete, delete-one, delete-all), *énoncés de requête de base* (evaluate, reply, ask-if, ask-about, ask-one, sorry), *énoncés de réponse de base* (error, sorry), *énoncés de requête multi-réponses* (stream-about, stream-

all, eos), *énoncés directifs de base* (achieve, unachieve), *énoncés générateurs* (standby, ready, next, rest, discard, generator), *énoncés de publication de capacité* (advertise), *énoncés de notification* (subscribe, monitor), *énoncés de réseau* (register, unregister, forward, pipe, break, transport-address), et *énoncés de facilitation* (broker-one, broker-all, recommend-one, recommend-all, recruit-one, recruit-all). Quelques exemples de messages KQML sont présentés à la Figure 2.3 ainsi qu'en Annexe A. Une liste exhaustive des énoncés performatifs peut être trouvée dans le rapport de spécification du langage KQML.

Différents paramètres peuvent être ajoutés à un énoncé performatif KQML. Chaque paramètre est précédé par un mot clé, ce mot clé débutant toujours par « : ». Plusieurs de ces mots clés implantent le niveau communication du message. C'est le cas notamment des mots clés *sender*, *receiver*, *reply-with* et *in-reply-to*. Le mot clé *content*, quant à lui, précède l'information véhiculée par le message et implante la couche contenu du message. L'ordre dans lequel les mots clés sont utilisés n'a pas d'importance puisqu'ils sont traités à tour de rôle par l'agent destinataire. Une définition formelle en format BNF du langage KQML est disponible dans le rapport de spécification de KQML ainsi qu'à l'Annexe A. Voici quelques exemples de mots réservés pour l'identification des paramètres :

:sender <mot>	identification de l'agent émetteur du message ;
:receiver <mot>	identification de l'agent destinataire du message ;

:reply-with <expression>	mentionne si l'émetteur attend une réponse et, si oui, l'étiquette d'identification de la réponse ;
:force <word>	précise la durée pour laquelle l'énoncé demeure valide; si <word> prend la valeur *permanent*, l'émetteur ne démentira jamais cet énoncé performatif.

L'ensemble de l'énoncé, de ses mots réservés et de leurs paramètres est transféré sous forme d'une chaîne de caractères ASCII afin d'éviter la nécessité de traduction entre divers modes de communication. Ces chaînes de caractères sont alors acheminées par un mécanisme de communication, donc de transfert physique d'information, commun aux agents communiquant : communication par objets, par protocole TCP/IP, par CORBA (Mowbray et Zahavi, 1995), etc. KQML, tel que défini présentement, permet la définition de nouveaux énoncés pour satisfaire les besoins particuliers d'agents ou groupes d'agents. Bien que ceci confère une grande souplesse au langage, il implique du même coup une plus grande variété d'énoncés et donc, un plus grand risque d'ambiguïté entre les agents, chaque agent n'étant pas tenu de comprendre chacun des énoncés rendus publics. Ce problème a d'ailleurs été soulevé par de nombreux chercheurs dont Mayfield (1995) et Cohen (1995). Ces mêmes critiques mentionnent aussi que KQML comprend à la base trop d'énoncés performatifs différents. Ainsi, plutôt que de tenter de définir des énoncés pour chacun des cas possibles, on suggère de restreindre le langage à quelques énoncés de base, le reste des énoncés devant être

construits à partir des énoncés de base. Ceci faciliterait l'élimination des différentes interprétations pouvant se glisser d'un agent à l'autre. La question est sous étude par le groupe de KQML.

Si un langage comme KQML permet l'échange de messages entre agents en standardisant la syntaxe et la sémantique de l'*enveloppe* du message, il n'impose cependant pas de restrictions quant à la façon d'exprimer le contenu de l'information proprement dite véhiculée par ce message, soit le paramètre accompagnant le mot réservé *content*. Par exemple, l'énoncé performatif *tell*, permettant à un émetteur de transmettre des connaissances à un destinataire, implique que les connaissances doivent être formatées selon un standard compréhensible par les deux agents communicants. Les travaux ayant pour but de définir de tels standards, qui est en fait un deuxième niveau de langage et de mots réservés, sont loin d'être aussi avancés que ceux des langages d'échange de messages. Bien que la syntaxe du langage KQML pourrait être utilisée à ce niveau aussi, elle n'est cependant pas suffisamment riche pour permettre l'expression de messages très complexes tels que des requêtes de bases de données et des expressions mathématiques devant être évaluées par le destinataire. L'équipe de recherche de KQML propose donc un langage plus riche pour exprimer la valeur du paramètre *content* d'un message KQML, soit le langage KIF (Knowledge Interchange Format) (Genesereth et Fikes, 1992). KIF, basé sur une syntaxe s'apparentant à celle du langage LISP, est cependant loin d'être aussi populaire que KQML. Sa syntaxe est plus complexe que celle de KQML et cette complexité, nécessitant un interpréteur très élaboré, peut être évitée

dans la plupart des cas par la définition d'un langage plus simple. Quelle que soit la syntaxe du langage choisi pour l'échange de l'information entre les agents, le contenu du message, c'est-à-dire les mots utilisés pour exprimer les connaissances ainsi que leurs valeurs, doivent provenir d'un vocabulaire commun aux agents communicants. Ce vocabulaire commun, appelé « ontologie », donne aux agents un cadre de référence à partir duquel ils peuvent partager et interpréter des informations. Une ontologie peut être développée pour chacun des domaines de connaissance pour lesquels des groupes d'agents font des interventions : recherche générique d'information, marchés boursiers, météo, commerce électronique, etc. Donc, en résumé, deux agents voulant échanger des message KQML devront connaître tous deux le langage et l'ontologie permettant d'exprimer la valeur associée au paramètre *content* des énoncés performatifs échangés. Le Figure 2.3 donne quelques exemples de messages KQML (Finnin, 1993).

Exemples de message KQML de type *subscribe* :

1. **(subscribe**
   :sender A
   :receiver B
   :reply-with s1
   :language KQML
   :ontology K10
   :**content** **(stream-about**
   :language KIF
   :ontology motors
   :content *motor1*))

2. **(tell**
   :sender B
   :receiver A
   :language KIF
   :ontology motors
   :in-reply-to s1
   :**content** (=(val (*torque motor*) (*sim-time 5*)) (scalar *12kgf*))

Explication des énoncés :

1. Dans son message *subscribe*, l'agent *A* s'inscrit auprès de l'agent *B* afin que *B* lui fournisse toutes les informations qu'il connaît et qui répondent à la requête exprimée dans le message KQML *stream-about* fourni comme valeur au paramètre *content* du message *subscribe*. Il s'agit donc de deux messages KQML imbriqués. Comme l'indique les paramètres du message *subscribe*, la valeur associée au paramètre *content* est exprimée en langage *KQML* à l'aide de l'ontologie *K10* et les résultats retournés devront être identifiés à l'aide de l'étiquette *s1*. Le message imbriqué *stream-about* fournit à B la demande d'information proprement dite. Cette demande d'information porte sur le concept *motor1* tel qu'indiqué dans la valeur associée au paramètre *content*. La syntaxe de la demande est exprimée à l'aide du langage *KIF* et sa sémantique, c'est-à-dire les mots utilisés pour exprimer la requête (*motor1* dans ce cas), proviennent de l'ontologie *motors*.

2. Le message *tell* est un exemple de réponse pouvant être fournie par *B* à la demande de *A*. La valeur associée au paramètre *in-reply-to* indique à *A* le message de requête auquel ces résultats sont associés. Les résultats, présentés dans la valeur associée au paramètre content, suivent la syntaxe du langage *KIF* et la sémantique *motors*. KIF fournit l'encadrement nécessaire au format du message et à l'utilisation de quelques mots réservés permettant de bien comprendre le contenu du message. Les mots *val* et *scalar* sont des exemples de mots réservés par *KIF*. L'ontologie *motors* donne une signification très précise à des mots spécifiques ainsi qu'aux valeurs pouvant leur être associées. Les éléments provenant de l'ontologie *motors* sont présentés en italique.

Figure 2.3. Exemples de messages KQML (Finin, 1993).

CHAPITRE 3

MODÉLISATION DE L'ASSISTANT INTELLIGENT DE RECHERCHE D'INFORMATION

Tout assistant de recherche repose à la fois sur la disponibilité de sources d'information sur supports électroniques (CD-ROM, disquettes, ...), ainsi que sur l'implantation d'un certain nombre d'aptitudes spécifiques : connaissance des outils de recherche sur ces sources d'information ; appel à ces sources d'information ; compilation, filtrage, et mise en forme des résultats ; etc. Grâce à ces capacités et à l'accès aux sources d'information, un tel assistant, humain ou automatisé, est en mesure d'effectuer de façon autonome une recherche d'information sur l'ensemble des sources connues, cachant ainsi à l'utilisateur la complexité reliée à la recherche. Dans le cadre de la modélisation d'un assistant informatisé, le contenu des sources d'information est externe à cet assistant tandis que les capacités spécifiques servent de fondement à l'architecture choisie pour cet assistant. Ce chapitre vise, en premier lieu, à détailler les tâches et sous-tâches d'un assistant intelligent de recherche d'information et, en second lieu, à présenter un modèle conceptuel multi-agents supportant l'ensemble des tâches de cet assistant.

3.1 Tâches et sous-tâches de l'assistant

L'objectif principal d'un assistant de recherche d'information consiste à faciliter, pour l'utilisateur, la recherche d'information parmi des sources généralement hétérogènes. Cet assistant sert donc d'intermédiaire entre un utilisateur et l'information proprement dite. Afin d'accomplir ce rôle d'intermédiaire, l'assistant doit être en mesure de compléter un cycle de base se résumant aux trois tâches suivantes : interpréter la requête de l'utilisateur ; effectuer la recherche d'information ; présenter les résultats à l'utilisateur. La Figure 3.1 présente ce cycle.

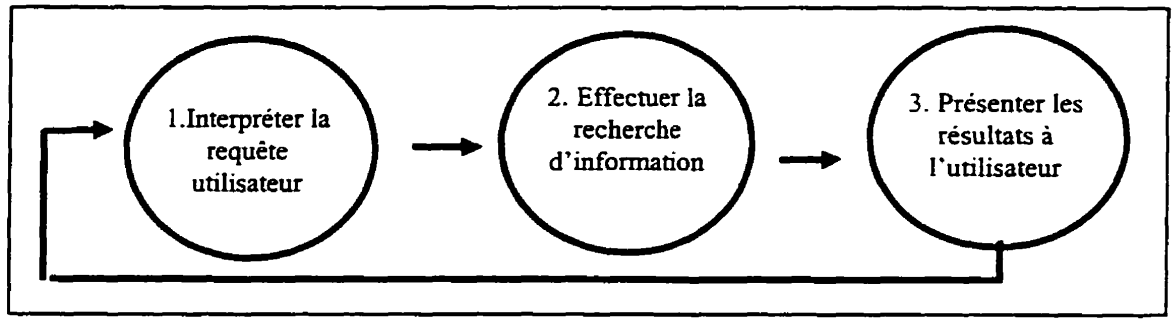


Figure 3.1.Cycle des tâches de base d'un assistant de recherche d'information.

Ce cycle, très simple à première vue, comporte en fait trois super tâches, chacune étant composée d'une série de sous-tâches de complexité variable. Cette complexité dépend principalement des contraintes associées à la sous-tâche en question, et donc du niveau d'intelligence requis de la part de l'assistant. Les sous-tâches et contraintes identifiées ci-après constituent en quelque sorte les critères de spécification de l'architecture recherchée. À la lumière de ces informations, le modèle conceptuel est

présenté comme une solution intelligente distribuée pouvant répondre à l'ensemble des critères établis.

Le cycle présenté à la Figure 3.1 constitue en fait le premier niveau d'une analyse hiérarchique de la tâche qui a été complétée afin de bien définir les buts et besoins de chacun des éléments du cycle. Pour chacune des trois super tâches identifiées, une analyse hiérarchique de la tâche a été effectuée. Une analyse hiérarchique de la tâche s'effectue en décrivant les buts et les sous-buts d'une tâche jusqu'à ce qu'un niveau de détails satisfaisant soit atteint (Pelletier, 1995). Cette méthode d'analyse a été développée par Annet et Duncan (1967) et est largement utilisée lors du développement de systèmes de formation. L'identification des buts et sous-buts peut être complétée soit en observant des utilisateurs effectuant la tâche à analyser, soit, lorsqu'il n'est pas possible d'observer des utilisateurs, en se basant sur une évaluation intuitive ou heuristique de la tâche et en raffinant cette analyse avec les utilisateurs du système informatique dès qu'une implantation partielle de ce dernier est possible. Dans le cadre de ce mémoire, la deuxième technique a été choisie. La décomposition résultante des trois analyses est présentée dans les Figures 3.2 à 3.4. Ces figures mettent en évidence certaines caractéristiques identifiées chez l'assistant humain, présentées à la section 1.1.2, et jugées nécessaires chez un assistant intelligent de recherche d'information, dont :

- connaissance des sources d'information ;
- connaissance du client;

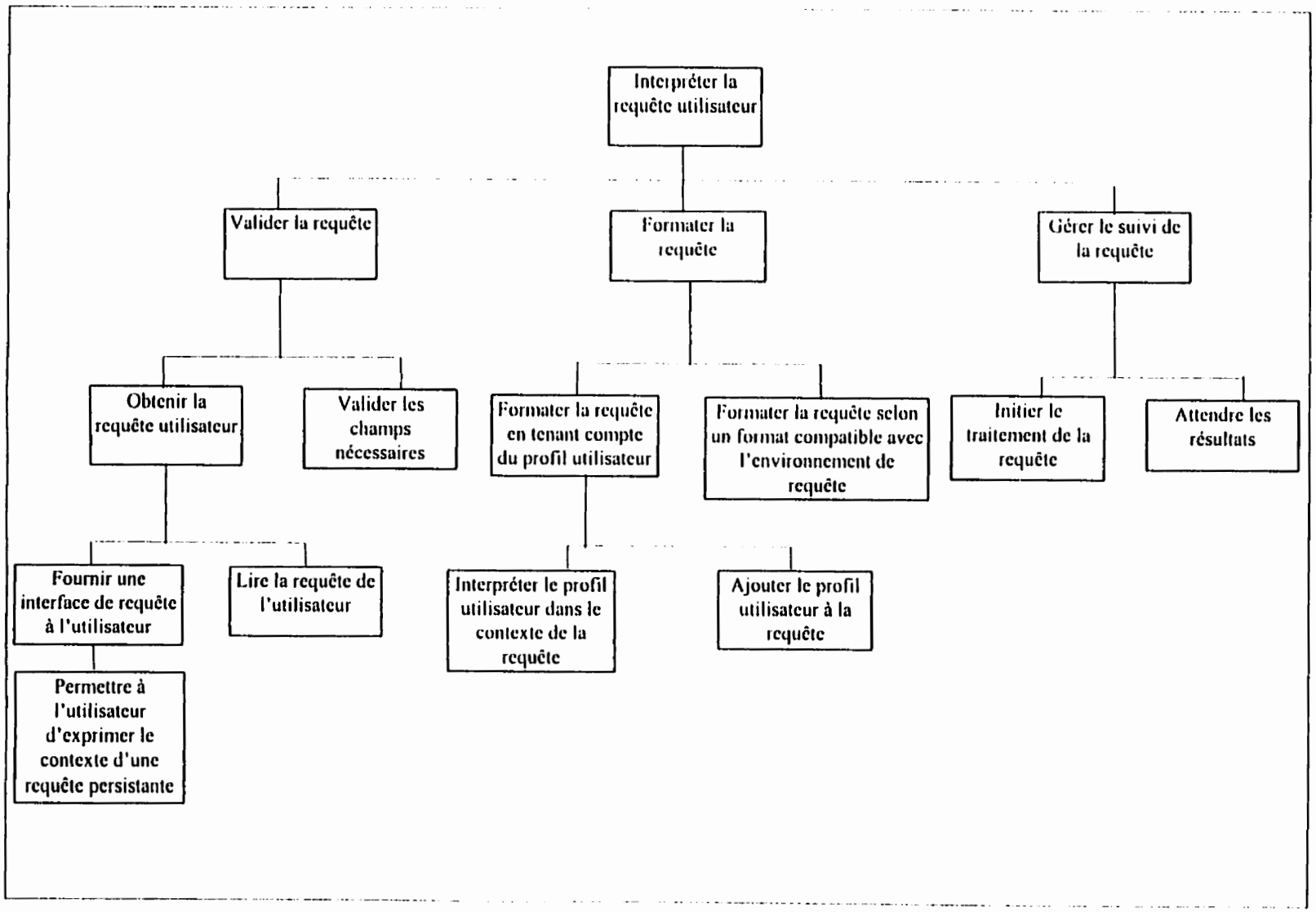


Figure 3.2. Analyse hiérarchique sommaire de la tâche « Interpréter la requête utilisateur ».

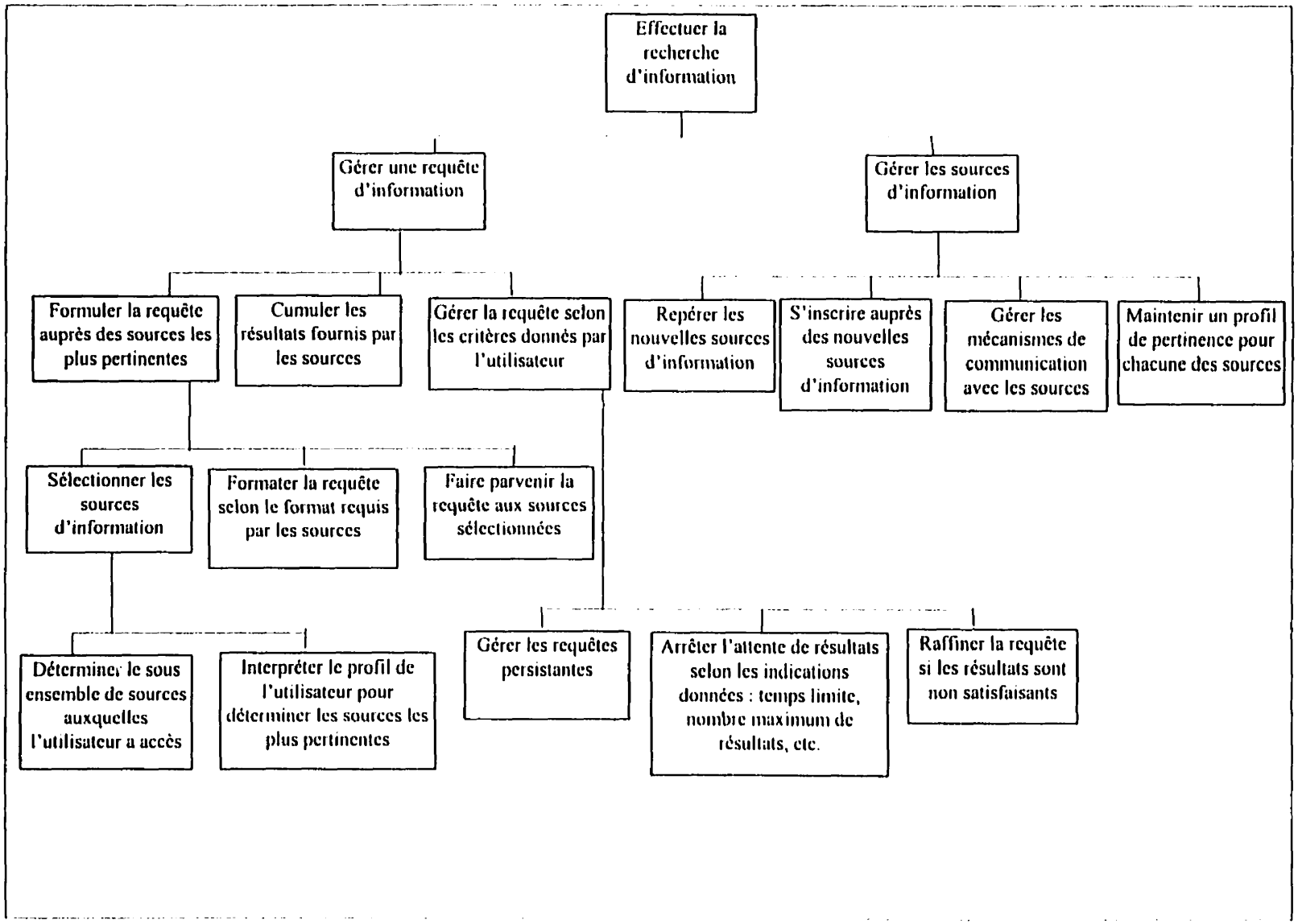


Figure 3.3. Analyse hiérarchique sommaire de la tâche « Effectuer la recherche d'information ».

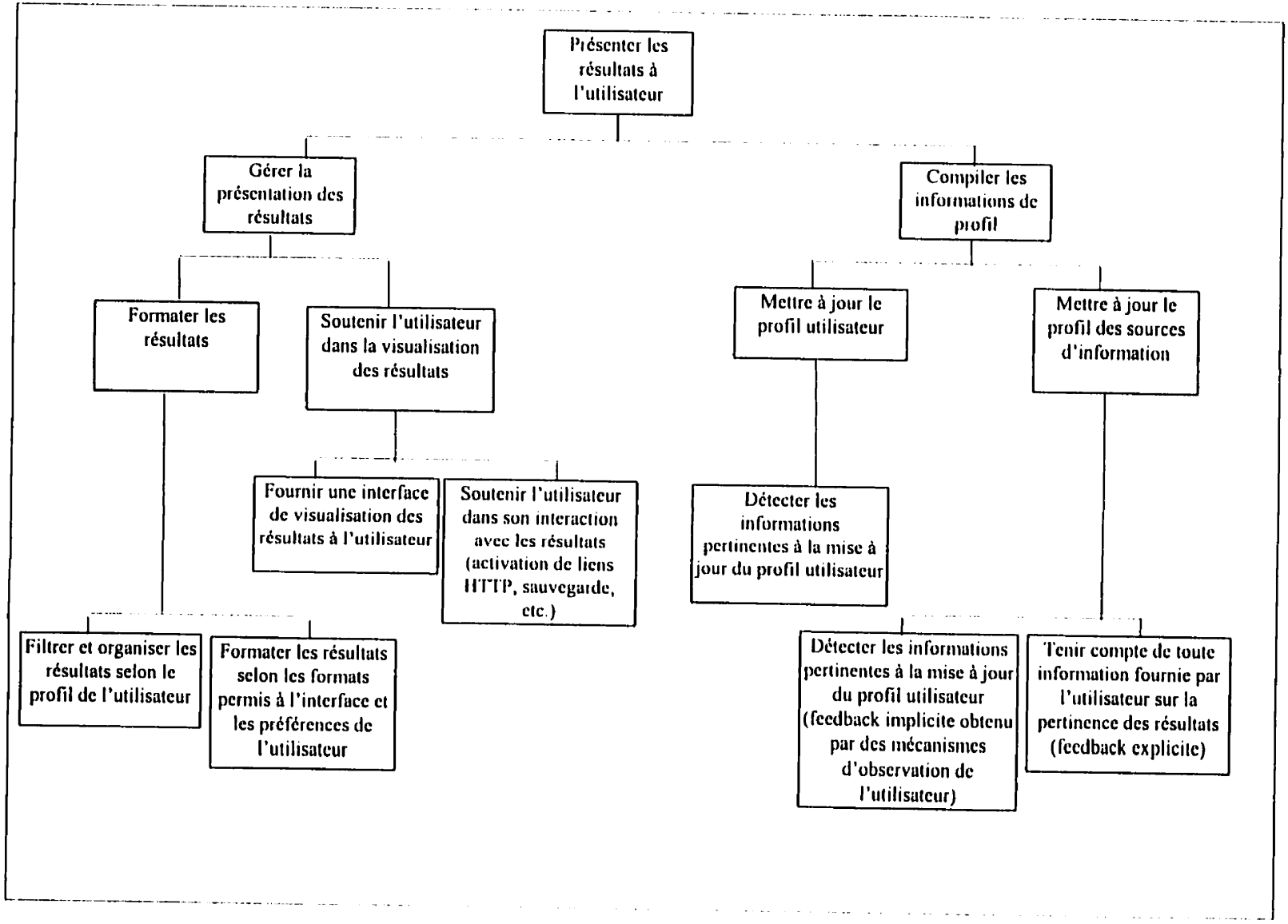


Figure 3.4. Analyse hiérarchique sommaire de la tâche « Présenter les résultats à l'utilisateur ».

- capacité à présenter adéquatement des résultats et à expliquer le cheminement de recherche;
- autonomie et pro-activité.

En plus de ces caractéristiques, l'assistant doit posséder des aptitudes particulières devant lui permettre de surmonter certains obstacles, dont (voir section 1.1.3) :

- indépendance face à l'hétérogénéité des sources d'information;
- indépendance face à la disponibilité des sources connues;
- indépendance de l'agent face à l'interface de communication utilisée pour exprimer la requête;
- capacité à traiter de façon parallèle plusieurs requêtes provenant d'un ou de plusieurs utilisateurs;
- niveau d'intelligence adapté à la complexité de la tâche.

3.2 Architecture multi-agents de recherche d'information

L'architecture présentée ici repose sur la mise en place d'un environnement de rencontre pour supporter la collaboration et la coordination entre des agents spécialisés autonomes. L'environnement, appelé ISAME, doit être persistant de telle sorte que les agents puissent, au besoin, y afficher des demandes de service ou les résultats d'un service rendu. La persistance fait ici référence au fait que l'environnement ISAME doit demeurer disponible sur une base continue, c'est-à-dire sans interruption et pour une durée indéfinie. Dans le cas où l'environnement doit être mis en attente pour une période donnée, l'état de ISAME doit être sauvegardé puis restauré lors de son retour en activité

de façon à ce que les mémoires à long terme et à court terme de ISAME (les agents inscrits, les messages en transit, etc.) soient conservées. L'ensemble des tâches de l'assistant n'incombe donc pas à une entité informatique donnée, mais plutôt à un travail d'équipe d'agents spécialisés. Comme le veut la philosophie multi-agents, aucun de ces agents n'est responsable pour la totalité des tâches de l'assistant de recherche, ni pour l'ensemble du contrôle des activités devant permettre l'accomplissement du but ultime, soit de fournir de l'information à l'utilisateur. Plutôt, les agents, par le biais de messages, font appel aux services de l'un et de l'autre et c'est par cette chaîne de collaboration que les buts de l'assistant sont rencontrés. Les cinq principes de base de l'architecture ISAME, décrits ci-après, sont :

- architecture ouverte imposant le minimum de contraintes aux agents ;
- intégration de quatre grandes classes d'agents ;
- autonomie des agents prise pour acquis ;
- communication par messages déclaratifs à l'aide de messages de type KQML, ainsi que d'un langage d'expression de contenu d'information et d'ontologies spécifiques à ISAME, soit le langage ISAME-L et les ontologies Biblio-virtuelle (*VirtualLibrary*) et Agent ;
- optimisation de l'environnement par activités de « ramasse miettes » .

3.2.1 Philosophie de l'architecture proposée

La philosophie qui sert de fondement à ISAME est de constituer une architecture ouverte soutenant les activités multi-agents de type recherche d'information. Ce soutien

est effectué en fournissant au cœur de l'environnement une série d'agents internes, c'est-à-dire créés lors de la création de l'environnement ISAME, ayant les compétences pour faciliter la recherche efficace d'information selon les tâches et sous tâches identifiées précédemment. Bien que ISAME soit présenté ici en terme de recherche d'information seulement, l'environnement pourrait être facilement élargi pour faciliter plusieurs autres types d'activités tels le commerce électronique ou la gestion de réseaux, à condition que ces activités soient basées sur une approche multi-agents. Afin de respecter cette ouverture, mais aussi pour donner le maximum de latitude aux concepteurs d'agents qui désirent utiliser ISAME, l'architecture impose un minimum de contraintes aux agents. Ces contraintes sont :

- comprendre les langages KQML et ISAME-L, ainsi que les ontologies Agent et/ou Biblio-virtuelle (*VirtualLibrary*), tels que présentés ci-dessous et en Annexes A à D ;
- publier la liste des énoncés performatifs de types KQML qu'il supporte ;
- publier une adresse physique à laquelle cet agent peut être rejoint ;
- lire et écrire de façon *autonome* (c'est-à-dire sans sollicitation de la part de ISAME) des messages selon le mécanisme de boîtes aux lettres défini dans ISAME.

S'il respecte ces contraintes, un agent pourra s'enregistrer auprès de ISAME et entrer en communication avec d'autres agents. Cependant, ii n'est pas garanti qu'un agent trouvera d'autres agents avec qui communiquer ! En effet, puisque ISAME ne

constitue qu'un lieu pour faciliter la rencontre d'agents, il n'a aucun contrôle sur qui s'enregistre auprès de l'environnement. Ainsi, il se peut qu'aucun agent de recherche d'information ne soit inscrit lorsqu'un agent utilisateur poste une requête d'information. L'agent utilisateur ne recevra donc aucune information en réponse à sa requête. Cependant, si un agent pouvant effectuer cette recherche d'information est enregistré auprès de l'environnement ISAME, il sera vraisemblablement en mesure de répondre à la requête d'information. Pour augmenter les chances de réponse à une requête d'information, l'utilisateur peut définir sa requête comme étant persistante. Cette requête demeurera donc valide dans ISAME jusqu'à ce que l'utilisateur la retire. Les résultats sont alors fournis à l'utilisateur selon la fréquence qu'il spécifie : tous les jours, plusieurs fois par jour, aussitôt qu'un nouveau résultat est trouvé, lorsqu'une nouvelle source d'information répond à la requête, etc.

3.2.2 Typologie des agents supportés dans ISAME

Dans le cadre de la recherche intelligente d'information, ISAME supporte quatre grandes catégories d'agents, chacune de ces catégories se différenciant des autres principalement par le type de services rendus par les agents qui en font partie. Ces quatre catégories d'agents sont les *Agents utilisateurs*, les *Agents information*, *les Agents de requêtes* et les *Agents de soutien* et sont définies comme suit :

1. *Agents utilisateurs (User Agents)* ou **UA** : Le savoir-faire des agents de type UA, que nous dénoterons UA_x dans ce chapitre où x identifie l'utilisateur auquel l'agent est associé, consiste à présenter les requêtes des utilisateurs dans

l'environnement multi-agents en servant de canal de communication entre l'utilisateur et l'environnement ISAME. Il existe un UA_x par utilisateur inscrit auprès de ISAME. Les UA_x permettent l'indépendance de l'assistant de recherche face à l'interface utilisateur, puisque le UA_x sert d'intermédiaire entre les deux. De plus, les UA_x étant des agents enregistrés auprès de ISAME mais créés à l'extérieur de celui-ci, il leur est possible d'intégrer le niveau de sophistication nécessaire au pré-traitement de l'information tel que désiré dans le contexte d'utilisation de l'information par l'utilisateur. Finalement, les UA_x se chargent eux-mêmes de la présentation des résultats, ainsi que du soutien à l'utilisateur dans la manipulation de ces résultats.

2. *Agents information (Information Agents)* ou **IA** : Les IA, que nous dénoterons IA_y dans ce chapitre où y identifie la source d'information représentée par l'agent, établissent le lien entre les sources d'information et l'environnement ISAME. Chaque source d'information doit inscrire son IA_y auprès de ISAME pour recevoir des requêtes. Tout comme les UA_x, les IA_y sont créés à l'extérieur de ISAME mais enregistrés auprès de celui-ci. Ces IA_y fournissent l'interface homogène nécessaire à l'interaction entre les agents de l'environnement ISAME et les sources d'information hétérogènes, et possèdent les connaissances nécessaires sur les outils de requêtes à utiliser sur les sources représentées. Ainsi, les sources d'information deviennent connues dans ISAME par l'enregistrement

de leur IA_y. De leur côté, ces IA_y assurent l'indépendance de ISAME face à l'hétérogénéité des sources, ainsi que de la disponibilité de celles-ci.

3. *Agents requêtes (Query Agents)* ou **QA** : Les QA sont des agents internes à ISAME qui veillent aux intérêts des requêtes des utilisateurs. Ils seront dénotés QA_{xn} dans ce chapitre, x représentant l'agent utilisateur ayant formulé la requête et n le numéro spécifique de cette requête dans l'environnement ISAME. Les QA_{xn} sont créés à partir de requêtes ponctuelles ou persistantes provenant des utilisateurs. Une requête est dite ponctuelle si elle n'est traitée qu'une seule fois et disparaît de l'environnement une fois que des résultats ont été présentés à l'utilisateur. Une requête persistante demeure dans l'environnement jusqu'à ce que l'utilisateur la retire et est traitée de façon régulière (1 fois par jour, lorsqu'un nouvel agent information devient disponible, etc.), l'utilisateur étant mis au courant de tout nouveau résultat. Les QA_{xn} se chargent d'amener la requête à terme et de retourner les résultats au UA_x. Entre autres tâches, ils communiquent avec le SFA (Agent filtre de sources ou *Source Filtering Agent*) pour déterminer les ensembles de IA_y auxquels la requête doit être acheminée, ils acheminent la requête aux IA_y, cumulent les résultats retournés, communiquent avec le RFA (Agent filtre de résultats ou *Result Filtering Agent*) pour faire filtrer et formater les résultats, puis retourne ces résultats au UA_x requérant.

4. *Agents de soutien (Support Agents)* ou **SA** : Les SA facilitent la coordination et la communication entre les trois autres catégories d'agents. Des exemples de SA sont :

   - *Facilitateur de communication (Communication Broker Agent)* ou **CBA** : Assure le routage des messages entre les agents actifs dans ISAME. Tout message est d'abord expédié au CBA qui s'occupe ensuite de l'acheminer au(x) destinataire(s) visé(s). Grâce au CBA, les agents interagissant dans ISAME n'ont pas à se connaître l'un l'autre, puisque le CBA se charge d'identifier les destinataires susceptibles d'être intéressés par le message expédié. Cependant, un agent peut toujours mentionner de façon explicite le destinataire du message. Dans ce cas, le CBA n'expédiera le message qu'au destinataire désigné. L'utilité principale du CBA est d'éviter que tous les agents aient à s'enregistrer auprès de tous les autres agents, ainsi que de conserver de l'information sur les autres agents disponibles.
   - *Agents registraires (Registry Agents)* ou **RA** : Ces agents, inspirés des *Registry agents* de l'architecture UMDL (Digital Library Project Research Proposal, 1995), gardent trace des agents en activité dans ISAME. Il existe un RA pour chacune des catégories d'agents suivantes : UA_x, IA_y et QA_{xn}. Grâce à ces registraires, le CBA peut facilement identifier les agents susceptibles d'être intéressés par un message donné. De plus, lorsqu'un agent enregistré auprès de ISAME désire modifier l'information qu'il publie sur

lui-même, il n'a qu'à enregistrer ces changements auprès de son agent registraire particulier afin qu'ils soient pris en compte dans l'ensemble de l'environnement. De même, s'il désire se retirer, il n'a qu'à en avertir son agent registraire et ce dernier s'occupera d'effacer toute trace de son passage dans l'environnement. Étant donné leur connaissance des agents dont la présence est dynamique dans ISAME (UA_x, QA_{xn} et IA_y), les registraires sont responsables des activités de ramasse miettes dans l'environnement. Ces activités de ramasse miettes, décrites en détail à la section 4.4, consistent à continuellement nettoyer l'environnement d'information concernant des agents qui ne sont plus accessibles ou qui se sont désistés auprès de l'environnement. Cette information comprend le profil de l'agent, les messages qui ne lui ont pas été expédiés, les message expédiés par cet agent et en cours de traitement par d'autres agents.

- *Agent filtre de sources (Source Filtering Agent)* ou **SFA** : À partir d'un profil utilisateur et des statistiques connues dans ISAME sur les IA_y, le SFA suggère un sous-ensemble de IA_y pour une requête donnée.
- *Agent filtre de résultats (Result Filtering Agent)* ou **RFA** : À partir des résultats amassés par un QA_{xn} et du profil utilisateur fourni par le UA_x duquel provient la requête, le RFA effectue un premier filtrage des résultats avant que ceux-ci soient expédiés au UA_x.

Deux de ces classes d'agents, les UA_x et les IA_y, sont de type externe. Il s'agit donc d'agents définis à l'extérieur de l'environnement ISAME et résidant sur des machines clients, mais qui possèdent l'interface de communication nécessaire à une interaction avec les autres agents enregistrés auprès de ISAME. Les QA_{xn} et SA sont dits de type interne, puisqu'ils sont créés à l'intérieur de ISAME, résident sur la machine serveur de l'environnement ISAME, et y demeurent jusqu'à ce qu'ils soient jugés inutiles.

3.2.3 Autonomie des agents

Les agents travaillent en parallèle et de façon autonome, chacun intervenant au besoin selon ses caractéristiques propres, les priorités qu'il s'est données et celles connues dans l'environnement. Ce travail consiste en deux grandes tâches : expédier du courrier et répondre au courrier entrant; mettre leur savoir-faire à profit et effectuer les tâches d'entretien.

1. expédier du courrier et répondre au courrier : Chaque agent a pour tâche de dépouiller le courrier qui provient de sa boîte aux lettres ainsi que d'y répondre.

   À cet effet, chacun des agents de l'environnement est muni d'une boîte aux lettres dans l'espace de communication (Communication Area ou CA) de ISAME. La boîte aux lettres est en fait une structure de données situées sur le serveur de l'environnement où se trouve le CBA et dans laquelle le CBA dépose tous les messages devant être livrés à l'agent. ISAME met aussi à la disposition de chacun des agents un commis, lequel s'occupe de livrer le courrier directement

à l'agent de façon régulière, lorsque cet agent est éveillé, c'est-à-dire, apte à compléter des activités. Ce commis, qui travaille de façon continue et autonome, se charge de résoudre l'adresse de l'agent auquel il est rattaché, et fait appel à une connexion réseau de type TCP/IP ou autre si nécessaire. Différents scénarios sont possibles si l'agent n'est pas en mesure de prendre ses messages au moment où le commis les expédie, c'est-à-dire si l'agent n'est pas éveillé. Ces scénarios, détaillés à la section 4.4, dépendent principalement du type de l'agent. Ainsi, si un IA_y n'est pas à l'écoute lorsqu'un message de requête d'information lui est expédié, ce message sera détruit. Cependant, s'il s'agit d'une réponse à une requête d'information et qui s'adresse à un UA_x, le message sera conservé et le commis tentera de rejoindre le UA_x plus tard. L'agent peut contrôler la fréquence d'envoi des messages ainsi que les priorités selon lesquelles ils doivent être triés en informant le registraire s'occupant de sa classe d'agents (UA, IA ou QA) de ses préférences, lequel en informe le commis en question. Ce mécanisme de commis assure l'indépendance de l'architecture face à la disponibilité des agents, et évite à l'agent de surveiller inutilement sa (ses) boîte(s) aux lettres.

Lorsqu'un agent désire expédier un message à un autre agent, il envoie ce message directement au CBA de l'environnement, lequel s'occupera de livrer le courrier dans la boîte aux lettres du ou des agents visés. Afin d'uniformiser la communication avec le CBA, tout message lui étant expédié doit passer par l'adresse unique publiée pour l'environnement ISAME, ceci s'appliquant aussi

pour les agents résidant sur le serveur où se trouve le CBA. Présentement, l'adresse unique publiée pour l'environnement ISAME consiste en une adresse de type TCP/IP. Tous les agents inscrits dans l'environnement ISAME doivent donc connaître le protocole de communication TCP/IP, du moins pour l'envoie de messages. Le CBA sert donc de point d'entrée unique au reste de l'environnement, tant pour les agents externes qu'internes. À cette fin, l'adresse TCP/IP du CBA doit demeurer publique en tout temps.

Lors de la réception d'un message, il se peut que le CBA ne puisse identifier de destinataire au message, ou qu'il n'arrive pas à interpréter le message. Il doit alors se charger de répondre à l'expéditeur en justifiant son refus d'effectuer la requête véhiculée par le message : incapacité à comprendre le message, erreur dans la composition du message, délai expiré, manque de ressources, etc. La section 3.4 fournit plus de détails quant au mécanismes d'échange de messages présents dans ISAME.

2. mettre leur savoir-faire à profit et effectuer les tâches d'entretien : Chacun des agents de ISAME doit, de façon autonome, mettre son savoir-faire au profit des autres agents en répondant à leurs requêtes; il doit aussi compléter les tâches d'entretien nécessaires au bon fonctionnement de la communauté d'agents. Ainsi, les UA_x doivent s'inscrire auprès de l'environnement et présenter les requêtes de l'utilisateur. Les IA_y doivent aussi s'inscrire auprès de l'environnement et continuellement mettre à jour leurs connaissances sur la (les) source

d'information qu'ils représentent. Les QA_{xn} doivent s'assurer que les conditions d'arrêt de recherche d'information ne sont pas atteintes (tous les IA_y ont fourni leurs résultats ou le délai accordé pour la requête est atteint) et, s'ils sont persistants, lancer une nouvelle requête à certains intervalles. Les registraires doivent s'assurer que les agents qui sont enregistrés auprès d'eux demeurent accessibles et éliminer ceux qui ne le sont plus.

Le succès de l'environnement dépend donc de l'exécution de ces deux catégories de tâches par l'ensemble des agents. Par l'échange de courrier, les agents collaborent et permettent le traitement de plusieurs requêtes d'un ou plusieurs utilisateurs de façon simultanée. D'un autre côté, l'équilibre et l'efficacité de l'environnement sont soutenus par l'exécution des tâches d'entretien.

3.2.4 Espace de communication (CA)

Les activités prenant place dans l'environnement ISAME sont toutes dirigées par l'échange de messages entre les agents. Aucun contrôle central n'est exercé, les agents s'échangeant services et information selon leurs besoins et bon vouloir. Afin de soutenir les agents dans cet échange de messages, l'environnement met à la disposition de chacun une boîte aux lettres par laquelle les messages parviennent à l'agent. Un commis de communication dédié est associé à chacune de ces boîtes aux lettres et s'occupe de livrer les messages directement au gestionnaire de communication de l'agent, soit la ou les entités (objets ou autres) qui s'occupent des activités de réception et d'expédition de messages sur la machine de résidence de l'agent. Ce système de livraison permet ainsi

une meilleure gestion du temps de chacun des agents, puisqu'ils n'ont pas à vérifier constamment leur boîte aux lettres afin de vérifier l'arrivée des messages. De plus, ce commis effectue quelques tâches complémentaires de pré filtrage, dans le but d'éviter un traitement de messages désuets par l'agent.

Comme vu précédemment, tous les messages sont distribués dans les boîtes aux lettres des destinataires par l'entremise du CBA. Si les destinataires sont mentionnés de façon explicite dans le message, le message sera livré dans les boîtes aux lettres correspondantes. Il y a une exception à cette règle : si un agent ne fait plus parti de l'environnement et que, par conséquent, sa boîte aux lettres a été détruite, le CBA retournera un message de type « sorry » à l'expéditeur, l'informant que le message ne peut plus être livré. Si le destinataire n'est pas mentionné de façon explicite, le CBA détermine la liste des agents pouvant être intéressés par le message. Ceci s'effectue en utilisant les listes des énoncés performatifs connus pour chacun des agents, cette liste étant connue du CBA dans le cas des agents de soutien, ou disponibles auprès des agents RA pour les agents UA_x, IA_y et QA_{xn}. Étant donné que, dans tous les cas, les agents doivent communiquer avec le CBA pour interagir avec les autres agents de l'environnement, l'adresse même du CBA devient la seule adresse publique nécessaire de l'environnement ISAME, celle devant être utilisée par les agents désirant s'inscrire auprès de l'environnement, expédier des messages à d'autres agents, ou encore se désister.

Afin que ces échanges de message s'effectuent sans heurts, un mécanisme de contrôle doit être mis en place pour assurer la synchronisation des activités dans les boîtes aux lettres. Cette synchronisation s'effectue par le commis, seul intervenant ayant directement accès à la boîte aux lettres. Le CBA inscrit donc les messages dans la boîte aux lettres par l'entremise du commis, le commis retire les messages de la boîte aux lettres pour les livrer à l'agent, et le commis est responsable de détruire les messages pour les activités de nettoyage. Cette synchronisation s'effectue par le biais de sémaphores utilisés par le commis sur les boîtes aux lettres.

Finalement, ISAME étant un environnement réparti, il incombe aux entités manipulant les messages pour les agents, soient d'un côté les processus spécifiques à la réception de messages à l'intérieur de l'agent, appelés gestionnaires de communication, et de l'autre côté les commis de boîtes aux lettres, de résoudre les problèmes inhérents à cette distribution. Ainsi, s'il est nécessaire d'utiliser une communication de type TCP/IP, les entités responsables doivent s'en occuper. Le protocole de communication utilisé est déterminé à la fois par les protocoles supportés par les commis de ISAME et par le protocole privilégié par le gestionnaire de communication de l'agent. Les premiers protocoles de communication supportés par ISAME seront TCP/IP et échanges de messages entre objets. La Figure 3.6 présente l'ensemble des situations possibles lors de l'échange de messages entre deux agents. Le protocole TCP/IP peut être utilisé indépendamment de la localisation physique du gestionnaire de communication et du commis (Figure 3.8), l'échange de messages entre objets n'étant possible que dans le cas

où l'agent, et donc son gestionnaire de communication, résident sur le serveur de l'environnement ISAME, lieu de résidence des commis (Figure 3.8).

3.2.5 Communication entre agents

La communication entre les agents s'établit donc par l'échange de messages transitant par le CBA et l'espace de communication avant de se rendre à l'agent destinataire. La syntaxe et la sémantique des messages sont basées sur des énoncés performatifs qui suivent les standards mis de l'avant par *KQML*. Le langage utilisé pour le paramètre « content » de certains énoncés performatifs est *ISAME-L*, et deux ontologies ont été définies, soit l'ontologie *Agent* pour le partage d'information relatif aux agents, et l'ontologie *Biblio-virtuelle* (*VirtualLibrary*) permettant d'exprimer les informations relatives à la recherche d'information proprement dite. Les principes de base du langage KQML, du langage ISAME-L, ainsi que des deux ontologies sont présentés dans les Annexes A à D. L'acheminement des messages par le CBA, les boîtes aux lettres et les commis constitue un modèle hybride des techniques de tableau noir et communication par messages. Il centralise les services et traitements communs à l'ensemble des agents, tels que filtrage et distribution de messages, et laisse le travail spécifique de filtrage et de traitement des données aux agents.

Des avantages du tableau noir, ISAME retient *une certaine uniformisation de la communication* et *le service centralisé de distribution d'information permettant aux agents d'être indépendants les uns des autres tout en évitant les possibilités de goulot d'étranglement.*

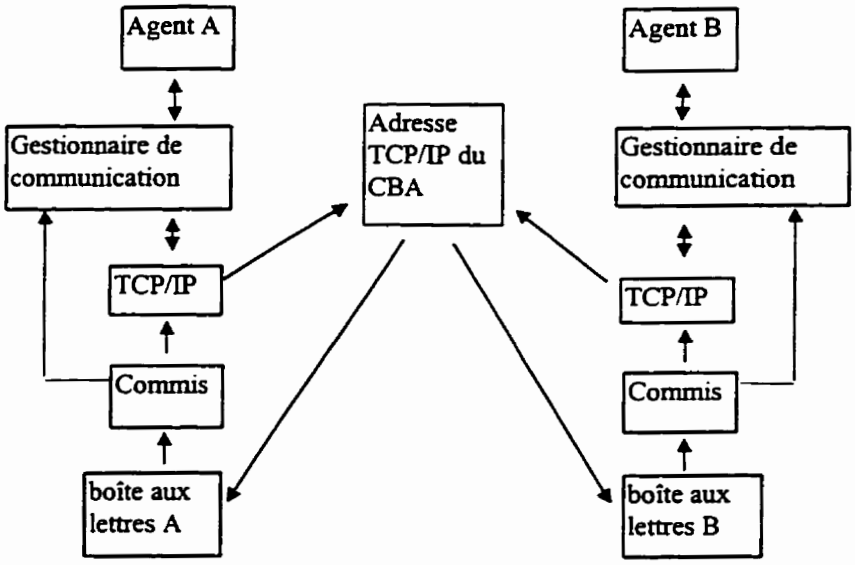


Figure 3.5 Communication bilatérale entre agents par l'entremise de gestionnaires de communication et de commis de boîtes aux lettres.

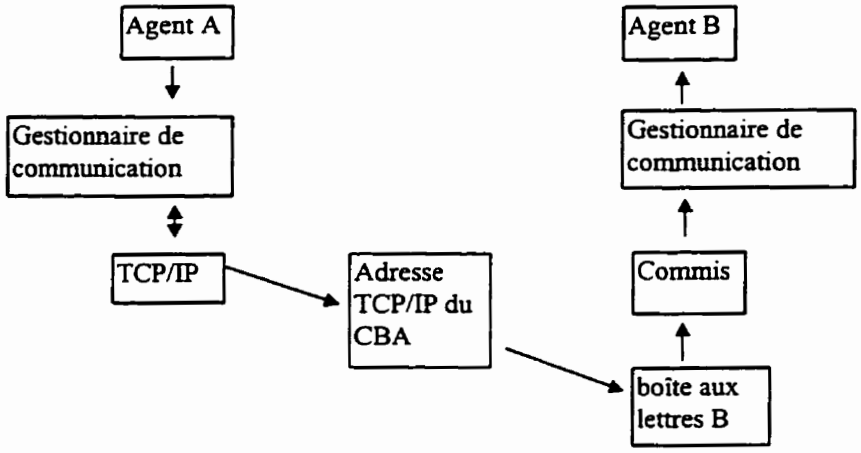


Figure 3.6 Échange d'un message par communication inter-processus entre deux agents, A et B, disponibles sur le serveur de l'environnement ISAME.

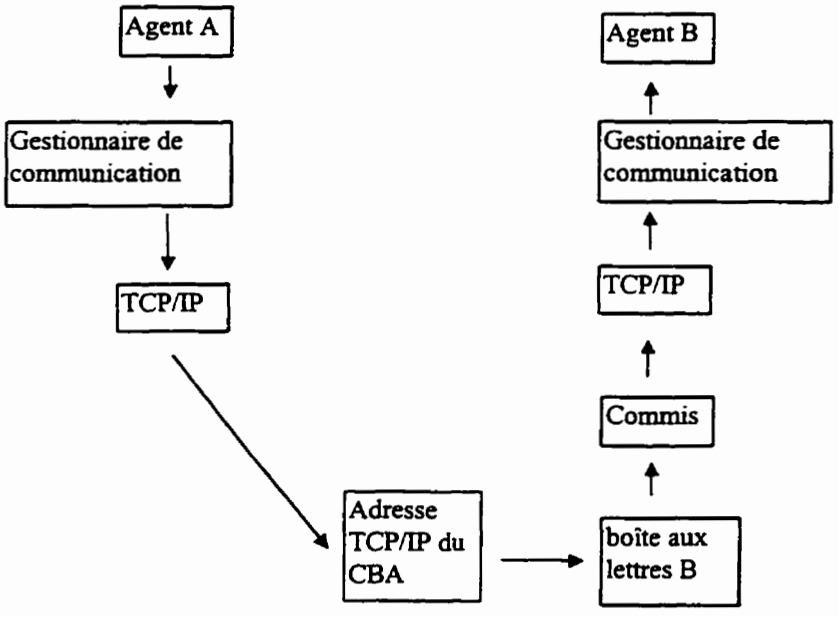


Figure 3.7 Échange de message par protocole TCP/IP permettant de faire abstraction de la localisation physique des agents, A et B étant disponibles sur le serveur de ISAME ou sur des machines clients distinctes.

1. uniformisation de la communication : Bien que le CBA n'effectue pas de contrôle quant au contenu de l'information échangée entre les agents, il s'assure cependant que tous les messages sont conformes au langage KQML. En effet, si un message ne satisfait pas la syntaxe du langage KQML ou son destinataire ne peut être identifié dans l'environnement ISAME, il sera détruit par le CBA. Dans le cas où le CBA est en mesure d'identifier l'expéditeur de message, il lui expédiera un message d'erreur indiquant pourquoi le message n'a pu être acheminé correctement.
2. service centralisé de distribution d'information : Tout message échangé dans ISAME doit transiter par le CBA avant de se rendre au destinataire. Ceci a pour avantage d'éviter à chacun des agents de connaître chacun de ses interlocuteurs éventuels et de connaître l'adresse précise d'un interlocuteur à un moment donné, tout en supportant

l'acheminement d'un message à un destinataire connu autant que la diffusion de messages d'intérêt public. Cependant, le contrôle n'est par centralisé par le CBA pour l'échange de données entre les agents, et aucun espace de données public n'est mis à la disposition des agents, évitant ainsi les problèmes de goulot d'étranglement reliés à cet espace. Bien que par les boîtes aux lettres un espace de données privé soit mis à la disposition de chacun des agents, cet espace ne comprend pas les connaissances de l'agent, celles-ci étant conservées dans l'agent, mais plutôt les messages destinés à l'agent. De plus, l'agent n'a pas à lire lui-même les messages inscrits dans sa boîte aux lettres puisque ces derniers lui sont livrés dans son propre environnement (sur la machine sur laquelle il réside et directement aux processus travaillant pour lui) par le commis qui lui est attitré. Les boîtes aux lettres servent donc à libérer le CBA de l'envoi des messages aux destinataires. Du même coup, tout comme dans la communication par tableau noir, ces boîtes aux lettres permettent aux agents d'échanger des messages de façon asynchrone, c'est-à-dire que le message peut être expédié même si le destinataire n'est pas prêt à le recevoir, la boîte aux lettres le conservant jusqu'à ce que l'agent soit en mesure de le recevoir auquel moment le commis se charge de l'expédier. Bien que la réception centralisée des messages par le CBA pourrait constituer un goulot d'étranglement, ce goulot peut être facilement désamorcé puisque plusieurs processus de réception de messages peuvent être mis en place de façon simultanée pour l'adresse publiée par le CBA, tout en fournissant au CBA la capacité d'acheminer plusieurs messages en parallèle grâce à l'utilisation de

processus autonomes légers (*thread*). Il est donc impossible pour un agent de bloquer l'ensemble des agents dans leurs activités.

De la communication par messages, ISAME conserve l'échange d'information par l'échange de messages proprement dit. Il *permet aux agents d'exercer un certain contrôle sur leurs échanges d'information tout en bénéficiant de services centralisés permettant du même coup une meilleure synchronisation de leurs activités*. Il conserve le désavantage *d'augmenter le trafic sur le réseau de télécommunication, mais dans le but de donner plus de liberté aux agents*.

1. contrôle laissé aux agents pour les échanges d'information : Dans le modèle de tableau noir traditionnel proposé par Nii, toute information inscrite sur le tableau devient accessible à l'ensemble des agents. Ceci nécessite donc à chacun des agents de filtrer toutes les informations afin de déterminer lesquelles peuvent lui être utiles. Cette méthode ne permet pas à l'agent qui affiche des informations de restreindre les utilisateurs de ces informations à un sous-ensembles d'agents donné. La communication par message permet cependant aux agents d'exercer ce contrôle. Cependant, lorsqu'un agent veut partager de l'information avec tous les agents de l'environnement, la communication par message n'offre pas de mécanisme centralisé permettant de faciliter ce travail. L'agent doit donc se charger d'expédier un message à chacun des agents, nécessitant donc que l'agent connaisse l'adresse de chacun des destinataires. Dans ISAME, l'utilisation du CBA permet à un agent à la fois de contrôler la distribution de l'information et, dans certains cas, de partager avec tous

les agents sans connaître l'adresse de chacun. Lorsqu'il désire expédier un message à un destinataire donné, l'expéditeur identifie ce destinataire dans son message KQML à l'aide du paramètre *receiver*, et le CBA se charge d'expédier le message à ce destinataire s'il est connu dans l'environnement ISAME. Si aucun destinataire n'est identifié, le CBA déterminera à partir de l'identification de l'expéditeur de message, identifié par le paramètre *sender* dans le message KQML, et le type d'énoncé perfomatif du message les agents pouvant être intéressés par ce message et se chargera de l'expédier à chacun d'entre eux.

2. augmentation du trafic et liberté des agents : Tout comme dans la communication par messages, l'utilisation du CBA dans ISAME provoque une augmentation du volume de messages échangés entre les agents. Cependant, cette augmentation d'échange de messages ne touche pas les agents externes de ISAME, mais plutôt les agents internes, n'imposant donc aucune baisse de performance du côté des agents représentants les utilisateurs et les sources d'information. D'autre part, l'utilisation d'un facilitateur de communication tel que le CBA permet d'offrir plus de liberté aux agents externes en mettant en place certaines optimisations de la communication par message. Ces optimisations prennent forme dans la mise en place des boîtes aux lettres et des commis qui évitent aux agents de se synchroniser lors d'un échange de message. L'expéditeur et le destinataire n'ont donc pas à être simultanément disponibles, et évite ainsi une écoute continue de la part des agents. De plus, grâce aux activité de ramasse miettes prenant place au cœur de l'environnement ISAME,

telles que présentées à la section 3.2.5, les risques de traitement de messages périmés par les agents sont diminués.

L'environnement ISAME supporte un sous-ensemble d'énoncés performatifs KQML, et chacun des agents interagissant dans ISAME comprend et utilise un certain nombres d'énoncés performatifs de ce sous-ensemble. Ce sous-ensemble est présenté à la section 4.2. Les listes d'énoncés performatifs compris par les agents de type QA_{xn} et SA sont connues dans l'environnement, puisque ces agents sont internes. Cependant les agents de types UA_x et IA_y, puisqu'ils sont externes à l'environnement, doivent publier ce sous-ensemble auprès de leur registraire respectif, ce registraire inscrivant pour chacun des agents la liste par défaut d'énoncés normalement traités par ce type d'agent. L'ordonnancement des messages de la boîte aux lettres s'effectue selon la technique *FIFO* (« First in First out »). Cette uniformisation des messages échangés entre les agents selon les formalismes KQML et ISAME-L permet d'atteindre une certaine homogénéité au niveau de la communication entre les agents, et ainsi cacher la grande diversité d'implantation des différents agents.

3.2.6 Nettoyage de l'environnement par activités de ramasse miettes

L'un des plus grands risques inhérent au type d'architecture choisie, c'est-à-dire par enregistrement des agents auprès d'un registraire et par la distribution de boîtes aux lettres, provient du fait que plusieurs éléments de cet environnement (agents, messages, boîtes aux lettres) peuvent devenir inaccessibles, désuets ou inutilisés avec le temps, et donc consommer des ressources qui pourraient être utilisées à meilleur escient. Il est

donc nécessaire d'instituer, dans le cadre de l'environnement, des mécanismes de nettoyage. Entre autres, ces mécanismes doivent :

- vérifier que tous les agents enregistrés donnent signe de vie régulièrement ;
- libérer les boîtes aux lettres non utilisées ;
- éliminer le courrier désuet de chacune des boîtes aux lettres existantes ;
- confirmer que les requêtes persistantes ont encore une utilité pour l'utilisateur desservi ;
- exclure les IA_y qui ont un niveau de performance trop bas depuis une période donnée.

Ces tâches de nettoyage incombent aux agents internes de ISAME, principalement les registraires et le CBA, en collaboration avec les commis de communication. Ce sont eux qui font le suivi sur les IA_y, UA_x et QA_{xn} à partir du moment de leur enregistrement dans ISAME, jusqu'à leur désistement volontaire ou forcé, et qui gèrent la logistique reliée aux boîtes aux lettres. Puisque les registraires et le CBA sont persistants et ont une durée de vie égale à celle de l'environnement ISAME, ils permettent une gestion cohérente de cet environnement. Les commis, bien que temporaires (leur durée de vie correspond à la durée de l'enregistrement de leur agent dans ISAME), possèdent des informations critiques pouvant faciliter le travail des registraires et du CBA. De fait, puisque les commis communiquent régulièrement avec leur agent, soit par le protocole TCP/IP ou localement, ils savent si ceux-ci demeurent accessibles ou non. Si un agent est déclaré inaccessible ou désuet, différentes actions sont intentées par le registraire,

soient : exclure l'agent de l'environnement, détruire sa boîte aux lettres et son commis, et demander au CBA qu'il avise tous les autres agents d'ignorer toute requête provenant de cet agent. S'il s'agit d'un agent de type UA_x, le registraire responsable des QA_{xn} se verra chargé d'éliminer tout agent QA_{xn} persistant rattaché au UA_x exclus.

Les commis ont aussi une autre utilité face à ce nettoyage continu. Puisqu'ils manipulent directement les messages de la boîte aux lettres, ils sont en mesure d'éliminer les messages désuets (exemples : une requête d'information suivie d'une annulation de la requête ; une requête provenant d'un agent qui n'est plus enregistré dans ISAME ; une requête dont le délai d'attente est atteint). Ces activités de nettoyage, ainsi que quelques autres, seront revues en détail dans la section 4.4.

Il est important de souligner ici les hypothèses fondamentales servant de base à l'efficacité des activités de ramasse miettes, c'est-à-dire : une durée de vie *indéfinie* et *exempte de panne* de la part des agents de types CBA et registraires, et une vie *exempte de pannes* de la part des commis et boîtes aux lettres. En effet, si l'un des agents CBA ou registraire venait à mourir ou à mal effectuer ses tâches de ramasse miettes, ou si l'un des commis devenait inaccessible ou appliquait mal les consignes, la qualité et l'efficacité de ces activités pourraient être remises en question. Dans le contexte du mémoire, la validité des hypothèses est prise pour acquis.

Le Tableau 3.1 présente un résumé des principaux éléments faisant parti d'ISAME, leur sigle respectif et leurs fonctions principales.

Tableau 3.1. Éléments clés de l'architecture ISAME.

Composante	*Sigle*	*Fonction(s) principale(s)*
Agent utilisateur (User Agent)	UA_x	Représenter un utilisateur x en exprimant ses requêtes d'information.
Agent information (Information Agent)	IA_y	Représenter une source d'information y et répondre à des requêtes utilisateurs.
Agent de requête (Query Agent)	QA_{xn}	Effectuer le suivi d'une requête n pour l'utilisateur x.
Agents de soutien (Support Agents)	SA	Ensemble des agents offrant des services internes à l'environnement ISAME. Catégories d'agents de soutien : CBA, RFA, SFA et RA.
Facilitateur de communication (Communication Broker Agent)	CBA	Acheminer les messages transitant entre les agents.
Agent filtre de sources (Source Filtering Agent)	SFA	Déterminer quel sous-ensemble d'agents information doit être invoqué pour répondre à une requête donnée.
Agent filtre de résultats (Result Filtering Agent)	RFA	Filtrer les résultats fournis par les IA_y à une requête.
Agent registraire (Registry Agent)	RA	Garder trace des agents en activité dans l'environnement ISAME. Liste des registraires : RA_{UA}, RA_{IA}, RA_{QA}.
Agent registraire des UA_x (UA Registry Agent)	RA_{UA}	Garder trace des agents de type UA_x.
Agent registraire des IA_y (IA Registry Agent)	RA_{IA}	Garder trace des agents de type IA_y.
Agent registraire des QA_{xn} (QA Registry Agent)	RA_{QA}	Garder trace des agents de type QA_{xn}.
Espace de communication (Communication Area)	CA	Mettre une boîte aux lettres à la disposition de chacun des agents.
Commis	---	Acheminer à l'agent qui y est relié les messages accumulés dans la boîte aux lettres.
Boîte aux lettres	---	Accumuler les messages destinés à l'agent qui lui est relié.

CHAPITRE 4

CHOIX ET CONSIDÉRATIONS D'IMPLANTATION

Afin de démontrer la faisabilité du modèle conceptuel présenté au chapitre précédent, une implantation partielle de ce modèle a été mise en place. Le présent chapitre regroupe les choix et considérations ayant servi de base à cette implantation : la définition formelle et la spécification de chacune des classes d'agents présentes dans ISAME, le choix des énoncés KQML supportés dans ISAME, ainsi que le détail du fonctionnement des activités de ramasse miettes prenant place dans l'environnement.

4.1 Définition formelle des ensembles d'agents reconnus dans ISAME

Quatre grandes classes d'agents ont donc été identifiées pour les activités de type recherche d'information prenant place dans ISAME : UA, QA, IA et SA. Cette section définit l'alphabet de ces ensembles d'agents, leurs éléments respectifs, ainsi qu'une série de fonctions utilisées pour la définition des catégories d'agents.

Définitions :

1. Grammaire des identificateurs :

< **lettre** > → a | b | ... | z | A | B | ... | Z

< **chiffre** > → 0 | 1 | ... | 9

< **caractère spécial** > → _ | - | ~ | \ | . |

< **identificateur** > → <lettre ou chiffre> {<lettre ou chiffre ou caractère spécial>}

2. Fonctions utilisées :

Trois fonctions sont définies ci-après pour faciliter la description des ensembles d'agents participant à ISAME, soit : etat(), provenance() et type().

- *etat(A)* :

Fonction indiquant l'état de l'agent A.

etat(A) ∈ E = {éveillé, dormant, porté disparu, désuet}.

Un seul état s'applique à un agent à tout instant donné.

Pour chacune des catégories d'agents présentes dans ISAME, les états possibles forment un sous-ensemble de E. La Figure 4.1 indique les transitions possibles entre les états d'un agent. La section 4.3 fournit les détails quant aux sous-ensembles d'états et de transitions s'appliquant à chacune des classes d'agents. Les quatre états possibles sont définis ainsi :

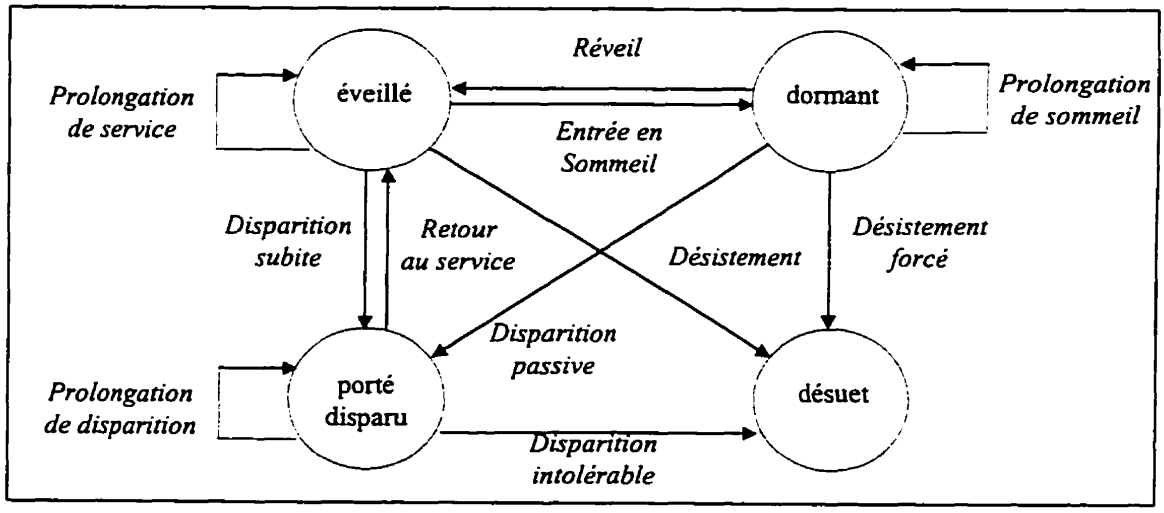


Figure 4.1 Automate de transitions sur les états des agents de ISAME.

éveillé :

Un agent est éveillé (*AWAKE*) durant les périodes au cours desquelles il interagit avec les autres agents de l'environnement ISAME, c'est-à-dire qu'il traite des demandes, envoie des messages, ou encore qu'il est en attente de messages.

dormant :

Un agent est dormant (*ASLEEP*) durant les périodes au cours desquelles il ne peut être rejoint par les autres agents de l'environnement ou par son commis de message. Un agent tombe en état dormant de façon volontaire (un agent utilisateur qui se débranche temporairement de ISAME, ou un agent requête persistant qui a terminé un cycle de requête mais qui n'a pas encore débuté le prochain cycle).

porté disparu :

Un agent est porté disparu (*LOST*) lorsqu'il n'a pu être rejoint par l'environnement ISAME ou son commis de message pour une période donnée, le rendant ainsi indésirable dans l'environnement ISAME. Un agent peut être déclaré *porté disparu* lorsqu'il a été *dormant* pour une période dépassant la limite permise par ISAME.

désuet :

Un agent est désuet (*DEAD*) lorsque ses activités ne sont plus requises ou désirées à l'intérieur de l'environnement ISAME. C'est le cas notamment

des agents requêtes qui ont terminé leur travail, des agents information dont le niveau de performance est jugé trop bas, ou d'un agent porté disparu depuis trop longtemps. L'état désuet est un état terminal, puisqu'il signifie la disparition définitive de l'agent concerné dans ISAME.

- *provenance(A)* :

Fonction indiquant la provenance de l'agent A.

provenance(A) $\in$ P = {interne, externe}.

Un agent dont la provenance est interne est un agent créé directement par l'environnement ISAME et résidant sur le serveur de l'environnement. Tous les autres agents ont une provenance dite externe.

La provenance ne peut être instanciée qu'une seule fois pour chacun des agents. Ainsi,

(provenance(A) = interne) $\vee$ (provenance(A) = externe)

- *type(A)* :

Fonction indiquant le type de l'agent A.

type (A) $\in$ T = {ponctuel, persistant, sans objet}.

Les types possibles d'un agent forment un sous-ensemble de T.

3. $\mathbf{U} = \{ U_a, U_b, ..., U_n \}$:

L'ensemble des utilisateurs ayant accès à l'environnement ISAME, où

- a, b, ..., n sont des chaînes de caractères non nulles identifiant les utilisateurs. Dans le cas où ces chaînes de caractères représenteraient l'adresse électronique de

l'utilisateur, le format BNF (Backus-Naur Form) des chaînes serait le suivant (Sudkamp, 1991) :

<utilisateur> → <identificateur> {@} {<identificateur>},

4. **UA** = $\{UA_a, UA_b, ..., UA_n\}$:

L'ensemble des agents utilisateurs (User Agent ou UA) actifs dans ISAME et connus de l'agent RA_{UA}, où

- UA_a identifie l'agent utilisateur de l'utilisateur U_a,
- $\forall UA_i \in UA$, etat(UA_i) $\in$ E,
- $\forall UA_i \in UA$, provenance(UA_i) = externe,
- $\forall UA_i \in UA$, type(UA_i) = sans objet,
- le nombre d'éléments de UA pouvant être inscrits simultanément dans un environnement ISAME n'est limité qu'aux ressources disponibles sur le serveur de l'environnement.

5. **QA** = $\{Qa_{a1}, Qa_{a2}, ..., QA_{bi}, ..., QA_{n1}, ..., QA_{nm}\}$:

L'ensemble des agents représentant des requêtes (Query Agent ou QA) actives dans ISAME et connues de l'agent RA_{QA}, où

- QA_{am} représente la $m^{ième}$ requête de l'agent UA_a enregistrée auprès de l'agent RA_{QA}.
- 1, ..., m sont des entiers positifs identifiant le numéro de requête et défini comme suit :

< requêteID > → < chiffre > {< chiffre >}*,

- $\forall QA_{ai} \in QA$, etat(QA_{ai}) $\in \Phi$; $\Phi \subset E$; Φ = {éveillé, dormant, désuet },
- $\forall QA_{ai} \in QA$, provenance(QA_{ai}) = interne,
- $\forall QA_{ai} \in QA$, type(QA_{ai}) $\in$ {ponctuel, persistant}.
- le nombre d'éléments de QA pouvant être accueillis simultanément dans un environnement ISAME n'est limité qu'aux ressources disponibles sur le serveur de l'environnement.

6. **IA** = {IA_a, IA_b, ..., IA_n} :

L'ensemble des agents représentant des sources d'information (Information Agent ou IA) actifs dans ISAME et connues de l'agent RA_{IA}, où

- IA_a représente la source d'information a,

  a, ..., n sont des chaînes de caractères non nulles identifiant les sources,

- $\forall IA_a \in IA$, etat(IA_a) $\in$ E,
- $\forall IA_a \in IA$, provenance(IA_a) = externe,
- $\forall IA_a \in IA$, type(IA_a) = sans objet,
- le nombre d'éléments de IA pouvant être accueillis simultanément dans un environnement ISAME n'est limité qu'aux ressources disponibles sur le serveur de l'environnement.

7. **SA** = { CBA, RA, SFA, RFA, ...} :

L'ensemble des agents de soutien (Support Agent ou SA) de l'environnement ISAME, où

- CBA (Communication Broker Agent) désigne l'agent responsable du routage des messages entre les agents connus dans l'environnement ISAME,
- RA désigne l'ensemble des agents registraires (Registry Agent),
- SFA désigne l'agent filtre de sources (Source Filtering Agent), un agent spécialisé dans le filtrage de sources d'information,
- RFA désigne l'agent filtre de résultats (Result Filtering Agent), un agent spécialisé dans le filtrage de résultats de recherche d'information,
- etat(CBA) = etat(SFA) = etat(RFA) = éveillé,
- provenance(CBA) = provenance(SFA) = provenance(RFA) = interne.

8. **RA** = { RA_{UA}, RA_{QA}, RA_{IA}} :

L'ensemble des agents registraires (Registry Agent ou RA) utilisés dans l'environnement ISAME,

- RA $\subset$ SA,
- Cardinalité(RA) = 3,
- RA_{UA} est responsable de l'enregistrement des agents UA_i dans l'environnement ISAME, où

  $\Rightarrow$ Cardinalité(RA_{UA}) =

  Cardinalité {UA_i enregistré auprès de RA_{UA} / etat(UA_i) = éveillé} +

  Cardinalité {UA_i enregistré auprès de RA_{UA} / etat(UA_i) = dormant} +

  Cardinalité {UA_i enregistré auprès de RA_{UA} / etat(UA_i) = porté disparu} +

  Cardinalité {UA_i enregistré auprès de RA_{UA} / etat(UA_i) = désuet} =

$\sum_{k \in E}$ Cardinalité {UA_i enregistré auprès de RA_{UA} / etat(UA_i) = k},

- RA_{QA} est responsable de l'enregistrement des agents QA_{ij} dans l'environnement ISAME, où

  $\Rightarrow$ Cardinalité(RA_{QA}) =

  Cardinalité {QA_{ij} enregistré auprès de RA_{QA} / etat(QA_{ij}) = éveillé} +

  Cardinalité {QA_{ij} enregistré auprès de RA_{QA} / etat(QA_{ij}) = dormant} +

  Cardinalité {QA_{ij} enregistré auprès de RA_{QA} / etat(QA_{ij}) = désuet} =

  $\sum_{k \in \Phi}$ Cardinalité {QA_{ij} enregistré auprès de RA_{QA} / etat(QA_{ij}) = k},

- RA_{IA} est responsable de l'enregistrement des agents IA_i dans l'environnement ISAME, où

  $\Rightarrow$ Cardinalité(RA_{IA}) =

  Cardinalité {IA_i enregistré auprès de RA_{IA} / etat(IA_i) = éveillé} +

  Cardinalité {IA_i enregistré auprès de RA_{IA} / etat(IA_i) = dormant} +

  Cardinalité {IA_i enregistré auprès de RA_{IA} / etat(IA_i) = porté disparu} +

  Cardinalité {IA_i / etat(IA_i) = désuet} =

  $\sum_{k \in E}$ Cardinalité {IA_i enregistré auprès de RA_{IA} / etat(IA_i) = k},

- etat(RA_{UA}) = etat(RA_{QA}) = etat(RA_{IA}) = éveillé,
- provenance(RA_{UA}) = provenance(RA_{QA}) = provenance(RA_{IA}) = interne,
- type(RA_{UA}) = type(RA_{QA}) = type(RA_{IA}) = sans objet.

La Figure 4.2 donne une vue d'ensemble des éléments principaux faisant partie de chacun des environnements ISAME.

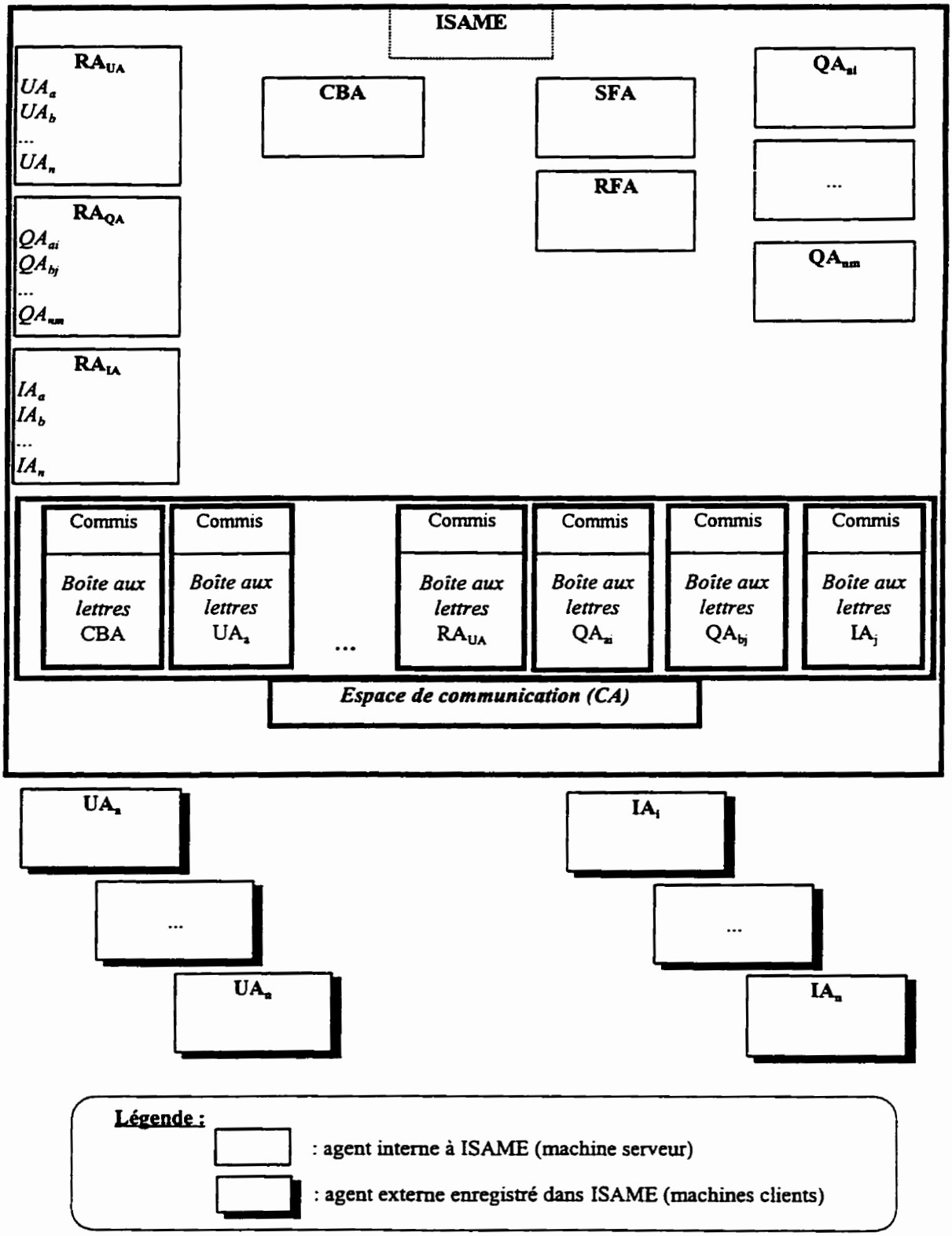


Figure 4.2. Composantes principales de l'environnement ISAME.

4.2 Sous-ensemble d'énoncés KQML retenu pour ISAME

Pour entrer en communication avec les autres agents de l'environnement, chacun des agents inscrits auprès de l'environnement ISAME doit parler les langages KQML et ISAME-L, et comprendre l'ontologie Agent. De plus, les agents impliqués dans les activités de recherche d'information, tels les agents dans UA et les agents dans IA, doivent aussi comprendre l'ontologie Biblio-virtuelle (VirtualLibrary). Étant donné l'ambiguïté possible avec laquelle les énoncés performatifs du langage KQML peuvent être interprétés, il apparaît judicieux de restreindre à un minimum la variété des énoncés utilisés, et d'éviter d'étendre le langage KQML par l'ajout de nouveaux messages performatifs.

Afin d'éviter les problèmes reliés à KQML dans le cadre de ISAME, seulement un sous-ensemble d'énoncés performatifs est supporté par ISAME. Ce sous-ensemble a été établi en complétant les quatre étapes suivantes :

1. analyse des besoins d'énoncés performatifs des agents : pour chacune des classes d'agents définie dans ISAME, une analyse de chacune des tâches relevant de cette classe d'agent a été effectuée afin de déterminer les besoins d'échange d'information avec d'autres agents pour l'accomplissement de cette tâche : requête d'information auprès d'un autre agent, partage d'information avec un ou plusieurs agents, besoin d'enregistrement ou de désistement auprès de l'environnement, envoi d'une réponse à une requête d'information, réception d'une réponse ou d'une information, etc.;

2. sélection d'énoncés performatifs pour chacune des tâches : pour chacun des besoins d'échange de message identifiés, l'énoncé performatif KQML dont la définition officielle (Finin, 1993) se rapproche le plus du besoin a été sélectionné pour implantation dans la classe d'agents;
3. validation de cohérence entre les énoncés performatifs des agents : une fois les énoncés performatifs sélectionnés pour l'ensemble des classes d'agents connues dans ISAME, une validation a été effectuée afin d'assurer que l'utilisation d'un énoncé performatif demeure cohérente entre les différentes classes d'agents et que le sous-ensemble d'énoncés sélectionnés demeure minimal tout en couvrant l'ensemble des besoins des agents.

L'ensemble des énoncés performatifs contient les éléments suivants : *advertise*, *ask-about*, *broker-one*, *deny*, *eos*, *error*, *evaluate*, *recommend-all*, *register*, *reply*, *sorry*, *stream-about*, *subscribe*, *tell*, *transport-address* et *unregister*. Le but général et le format de chacun de ces messages performatifs sont présentés en Annexe A.

Chacun des agents n'est cependant pas forcé d'intégrer tous ces énoncés performatifs, mais il est fortement recommandé qu'il supporte un certain nombre de énoncés performatifs lui permettant d'offrir ses services aux autres agents, et de clairement exprimer ses besoins et services. La section 4.3 spécifie le sous-ensemble minimal d'énoncés performatifs recommandé pour chacun des types d'agents interagissant dans ISAME. Une description plus générale du langage KQML, du langage ISAME-L et des ontologies est donnée dans les Annexes A à D.

4.3 Spécification des classes d'agents

Chacun des agents inscrits auprès d'ISAME est spécifique de par le cœur de son implantation. Cependant, les agents appartenant à une même classe d'agents partage des caractéristiques communes qui en font des spécialistes reconnus pour un savoir-faire donné.

Ces caractéristiques communes prennent forme dans les énoncés performatifs compris et utilisés par les agents, leur savoir-faire spécifique, les états qui leur sont reconnus, leur provenance, etc. Bien que ces caractéristiques soient nécessairement implantées telles que décrites ci-dessous pour l'ensemble des agents internes, les concepteurs d'agents externes ne sont pas tenus de respecter toutes ces caractéristiques. Néanmoins, le respect de ces contraintes est fortement recommandé pour une interaction efficace de ces agents externes avec les autres agents inscrits auprès de ISAME.

Les paragraphes qui suivent détaillent les caractéristiques spécifiques souhaitables pour chacune de ces classes d'agents présentement acceptés dans ISAME. Ces caractéristiques portent sur les moments de début et de fin d'activités des agents, leur lieu de résidence, leur savoir-faire et les énoncés performatifs qu'ils devraient idéalement être en mesure de manipuler. Dans le reste du texte, UA_x, QA_{xn} et IA_y représentent respectivement tout élément de l'ensemble UA, de l'ensemble QA et de l'ensemble IA.

A. Les agents $UA_x \in UA$

Entrée en activité :

Quand : Par enregistrement auprès du RA_{UA} à la demande de son propriétaire.

Comment : Par envoi d'un message contenant l'énoncé performatif « register » au RA_{UA} de l'environnement.

Fin des activités :

Quand : Par désistement auprès du RA_{UA} à la demande de son propriétaire .

Comment : Par envoi d'un message contenant l'énoncé performatif « unregister » au RA_{UA} de l'environnement.

ou

Quand : Par rejet de la part du RA_{UA} lorsque l'état de l'agent est « désuet ».

Comment : L'agent est automatiquement enlevé de la liste des agents connus du RA_{UA} et l'environnement est nettoyé de toutes les traces laissées par cet agent.

Conséquences : Dans les deux cas, tout QA_{xn} travaillant pour le UA_x est aussi expulsé de l'environnement, la boîte aux lettres du UA_x et son commis sont détruits, et toute information connue sur l'agent UA_x est détruite. Par conséquent, toute requête d'information en traitement ou en suspend pour ce UA_x est terminée, tout message en attente dans la boîte aux lettres est détruit, et le profil accumulé jusqu'à ce jour par le RA_{UA} est perdu sans possibilité de récupération.

Lieu de résidence :

Poste de l'utilisateur.

Savoir-faire :

Implanté par envoi et réception de messages :

- s'enregistrer et se désister auprès d'environnements ISAME.

- informer l'environnement qu'il est éveillé et prêt à recevoir des messages.
- poster des requêtes d'information ponctuelles ou persistantes.
- poster des mises à jour du profil de l'utilisateur qu'il représente.
- annuler des requêtes d'information.
- interpréter les résultats reçus pour des requêtes d'information.
- poster des informations concernant la perception de l'utilisateur en fonction de résultats obtenus.

Autres savoir-faire :

- interagir avec l'utilisateur par le biais d'une interface utilisateur.
- capter une requête d'information exprimée par l'utilisateur.
- afficher les résultats de requête à l'utilisateur.
- supporter l'utilisateur dans son interaction avec les résultats de requête.
- capter les préférences de l'utilisateur en lui permettant de répondre à des choix explicites ou en observant son interaction avec les résultats.

Le Tableau 4.1 dresse une liste des énoncés performatifs manipulés par les agents de ce type ainsi que leur contexte d'utilisation. Ce tableau indique différentes requêtes d'information pouvant être postées par le UA_x auprès du CBA. Ces différentes requêtes sont mises en place par l'imbrication de énoncés performatifs. Quatre types de requêtes d'information peuvent être expédiés par un UA_x, nommément :

1. Trouver un agent pouvant effectuer une requête d'information ponctuelle, avec retour des résultats en un seul message. Le squelette d'une telle requête serait :

(broker-one :content (ask-about ...) ...).

2. Trouver un agent pouvant effectuer une requête d'information ponctuelle, avec retour des résultats en plusieurs messages. Le squelette d'une telle requête serait :

   (broker-one :content (stream-about ...) ...).

3. Trouver un agent pouvant effectuer une requête d'information persistante, avec retour des résultats en un seul message. Le squelette d'une telle requête serait :

   (broker-one :content (subscribe :content (ask-about ...) ...) ...).

4. Trouver un agent pouvant effectuer une requête d'information persistante, avec retour des résultats en plusieurs messages. Le squelette d'une telle requête serait :

   (broker-one :content (subscribe :content (stream-about ...) ...) ...).

B. Les agents $QA_{xn} \in QA$

Entrée en activités :

Quand : Lorsqu'une nouvelle requête d'information est postée dans l'environnement et qu'un agent de type QA_{xn} est créé par le CBA.

Comment : Par envoi d'un message contenant l'énoncé performatif « register » au RA_{QA} de l'environnement.

Fin des activités :

Quand : Par désistement auprès du RA_{QA}.

Comment : Par envoi d'un message contenant l'énoncé performatif « unregister » au RA_{UA} de l'environnement.

Tableau 4.1. Énoncés performatifs de base pour les agents ∈ UA.

Énoncé	*Quand*	*Utilisation spécifique et valeurs obligatoires pour les paramètres*
advertise	*envoi*	Informer le RA_{UA} des énoncés performatifs compris par l'agent. Destinataire : « RA_{UA} ».
ask-about	*envoi*	Requête d'information adressée à l'environnement ISAME par l'entremise du CBA. Tous les résultats de la requête devront être expédiés par le CBA en un seul envoi. Destinataire : aucun* .
broker-one	*envoi*	Demande adressée au CBA pour qu'une requête d'information soit prise en charge par un agent responsable. Destinataire : « CBA »
deny	*envoi*	Annuler une demande d'information. Destinataire : « CBA » .
eos	*réception*	Le CBA annonce la fin des résultats correspondant à une demande d'information « stream-about ».
error	*envoi* *réception*	
register	*envoi*	Enregistrement auprès de l'agent RA_{UA}. Destinataire : « RA_{UA} ».
reply	*réception*	Réception d'information en réponse à une requête.
sorry	*envoi* *réception*	
stream-about	*envoi*	Requête d'information adressée à l'environnement ISAME par l'entremise du CBA. Les résultats de la requête sont expédiés un à un par le CBA. La fin des résultats est atteinte lorsque le CBA envoie un message de type « eos ». Destinataire : aucun.

* Lorsqu'aucun destinataire n'est spécifié, le CBA se charge de déterminer les destinataires possibles.

Tableau 4.1. Énoncés performatifs de base pour les agents ∈ UA (suite).

Énoncé	*Quand*	*Utilisation spécifique et valeurs obligatoires pour les paramètres*
subscribe	*envoi*	Lorsqu'une requête d'information (« ask-about » ou « stream-about ») est insérée dans un message de type « subscribe », cette requête est automatiquement de type persistante et demeure valide jusqu'à la réception d'un message de type « deny ». Destinataire : aucun.
tell	*envoi*	Envoi d'information au RA_{UA} concernant le profil de l'utilisateur représenté par le UA_x, incluant les jugements de l'utilisateur sur les résultats reçus lors du traitement de requêtes. L'agent informe l'environnement de sa connexion ou déconnexion temporaire (l'agent demeure enregistré auprès de l'environnement). Destinataire : « RA_{UA} »
	réception	Réception d'information concernant l'environnement (ex. : l'expulsion d'un agent dont certains messages sont en traitement dans le système). Réception de la confirmation d'enregistrement auprès du RA_{UA} .
transport-address	*envoi*	Le UA_x informe le RA_{UA} de l'adresse à laquelle il peut être rejoint. Destinataire : « RA_{UA} ».
unregister	*envoi*	Désistement auprès de l'agent RA_{UA}. Destinataire : « RA_{UA} ».

ou

Quand : Par rejet de la part du RA_{QA} lorsque l'état de l'agent est « désuet » parce que la requête n'a plus à être traitée pour l'une des raisons suivantes : le cycle de la requête est terminé, l'agent UA_x associé a mis fin à ses activités ou a été expulsé.

Comment : L'agent est automatiquement enlevé de la liste des agents connus du RA_{QA} et l'environnement est nettoyé de toutes les traces laissées par cet agent.

ou

Quand : Par rejet de la part du RA_{QA} lorsque l'état de l'agent est « désuet » parce que l'agent est devenu inaccessible depuis trop longtemps.

Comment : L'agent est automatiquement enlevé de la liste des agents connus du RA_{QA} et l'environnement est nettoyé de toutes les traces laissées par cet agent.

Conséquences : Dans les deux premiers cas, aucune conséquence fâcheuse sur le UA_x associé puisque ce dernier a reçu l'information désirée ou s'est lui-même désisté auprès de l'environnement. Dans le dernier cas, la requête du UA_x ne sera pas traitée par l'environnement et devra être expédiée à nouveau si le UA_x désire recevoir une réponse, puisque ISAME ne réachemine pas automatiquement la requête vers une nouvelle instance de QA_{xn}. Dans tous les cas, le profil conservé sur le QA_{xn} par le RA_{QA}, la boîte aux lettres et le commis mis à la disposition du QA_{xn} sont détruits, y compris tout message en attente dans la boîte aux lettres

Lieu de résidence :

Sur le serveur de l'environnement ISAME.

Savoir-faire :

Implanté par envoi ou réception de messages :

- faire appel à l'agent SFA pour déterminer les IA_y qui doivent être utilisés.
- poster la requête d'information auprès des IA_y.
- cumuler les résultats de recherche des différents IA_y qui ont été interrogés.
- informer le RA_{IA} de son degré de satisfaction concernant les IA_y qui ont été interrogés.

- faire appel à l'agent RFA pour filtrer et formater les résultats.
- retour de résultats au UA_x requérant.

Autres savoir-faire :

- contrôler la durée d'attente des résultats fournis par les IA_y qui ont été interrogés.
- appliquer des stratégies de recherche si les résultats ne correspondent pas aux résultats requis par le UA_x requérant.
- si type(QA_{xn}) = persistant, contrôler le rythme de réactivation de la requête.

Le Tableau 4.2 dresse une liste des énoncés performatifs manipulés par les agents de ce type ainsi que leur contexte d'utilisation.

C. Les agents $IA_y \in IA$

Entrée en activités :

Quand : Par enregistrement auprès du RA_{IA} à la demande de son propriétaire.

Comment : Par envoi d'un message contenant l'énoncé performatif « register » au RA_{IA} de l'environnement.

Fin des activités :

Quand : Par désistement auprès du RA_{IA} à la demande de son propriétaire.

Comment : Par envoi d'un message contenant l'énoncé performatif « unregister » au RA_{IA} de l'environnement.

Tableau 4.2. Énoncés performatifs de base pour les agents ∈ QA.

Énoncé	*Quand*	*Utilisation spécifique et valeurs obligatoires pour les paramètres*
ask-about	*envoi* *réception*	Demande d'information auprès d'un agent de type IA_y. Les résultats proviendront du IA_y dans un seul message. Destinataire : un agent de type IA_y. Demande d'information provenant d'un UA_x. Les résultats devront être expédiés en un seul message. Expéditeur : un agent de type UA_x.
deny	*envoi* *réception*	Annuler une demande d'information auprès des agents SFA ou RFA. Destinataire : « RFA » ou « SFA » Annulation d'une demande d'information. Expéditeur : un agent de type UA_x.
eos	*envoi* *réception*	Informer le UA_x que ce message met fin à une série de messages répondant à une requête de type « stream-about ». Destinataire : le UA_x ayant posté la requête. Un IA_y informe qu'il a expédié toutes les informations en réponse à un énoncé de type « stream-about ». Expéditeur : un agent de type IA_y.
error	*envoi* *réception*	
evaluate	*envoi*	Demander à l'agent RFA de filtrer et formater les résultats obtenus pour une requête. Destinataire : « RFA ».
recommend-all	*envoi*	Demander à l'agent SFA qu'il recommande les IA_y les plus intéressants pour une requête donnée. Destinataire : « SFA ».
register	*envoi*	Enregistrement auprès de l'agent RA_{QA}. Destinataire : « RA_{QA} ».
sorry	*envoi* *réception*	

Tableau 4.2. Énoncés performatifs de base pour les agents ∈ QA (suite).

Énoncé	*Quand*	*Utilisation spécifique et valeurs obligatoires pour les paramètres*
reply	*envoi*	Retour d'information à un UA_x en réponse à une requête
		Destinataire : le UA_x ayant posté la requête.
	réception	Retour d'information en réponse à une requête exprimée par le QA_{xn}.
		Expéditeurs : IA_y, SFA, RFA.
stream-about	*envoi*	Demande d'information auprès d'un agent de type IA_y. Les résultats proviendront du IA_y dans plusieurs messages dont le dernier sera de type « eos ».
	réception	Destinataire : un agent de type IA_y.
		Demande d'information provenant d'un UA_x. Les résultats seront fournis dans plusieurs messages dont le dernier sera de type « eos ».
		Expéditeur : un agent de type UA_x.
subscribe	*envoi*	Demande d'information de type « ask-about » ou « stream-about » auprès d'un IA_y. Cette requête est automatiquement de type persistant et demeure valide tant que le QA_{xn} n'envoie pas de message de type « deny » pour en annuler l'effet.
	réception	Destinataire : un agent de type IA_y.
		Demande d'information de type « ask-about » ou « stream-about » en provenance d'un UA_x par l'entremise du CBA. Cette requête est automatiquement de type persistant et demeure valide tant que l'expéditeur n'envoie pas de message de type « deny » par l'entremise du CBA pour en annuler l'effet.
		Expéditeur : un agent de type UA_x.
tell	*envoi*	Informer le RA_{QA} de son degré de satisfaction sur le travail des Ia_y.
		Destinataire : « RA_{QA} ».
	réception	Réception de la confirmation d'enregistrement auprès du RA_{QA}.
		Expéditeur : « RA_{QA} ».
unregister	*envoi*	Désistement auprès de l'agent RA_{QA}.
		Destinataire : « RA_{QA} ».

ou

Quand : Par rejet de la part du RA_{IA} lorsque l'état de l'agent est « désuet ».

Comment : L'agent est automatiquement enlevé de la liste des agents connus du RA_{IA} et l'environnement est nettoyé de toutes les traces laissées par cet agent.

Conséquences : Dans les deux cas, le profil conservé sur le IA_y par le RA_{IA}, la boîte aux lettres et le commis mis à la disposition du IA_y sont détruits, y compris tout message en attente dans la boîte aux lettres. Puisque chaque QA_{xn} est informé de l'expulsion d'un IA_y, il est en mesure de répondre adéquatement à cette expulsion.

Lieu de résidence :

Poste du fournisseur d'information.

Savoir-faire :

Implanté par envoi ou réception de messages :

- s'enregistrer et se désister auprès de l'environnement ISAME.
- indiquer qu'il est éveillé et prêt à recevoir des messages.
- poster des résultats concernant des requêtes d'information reçues.

Autres savoir-faire :

- modifier la représentation de la requête pour qu'elle corresponde au format interne de la source d'information.
- formater les résultats obtenus selon le format ISAME-L.
- faire la mise à jour de ses connaissances sur la (les) source(s) d'information qu'il représente.

Le Tableau 4.3 dresse une liste des énoncés performatifs manipulés par les agents de ce type ainsi que leur contexte d'utilisation.

D. L'agent CBA

Entrée en activités :

Quand : À la création de l'environnement ISAME.

Fin des activités :

Quand : À la destruction de l'environnement ISAME.

Lieu de résidence :

Sur le serveur de l'environnement ISAME.

Savoir-faire :

Implanté par envoi ou réception de messages :

- faciliter le traitement de requêtes d'information provenant d'un UA_x en créant dans le système un QA_{xn} pouvant se charger de la requête dans ISAME.
- filtrer les messages provenant de chacun des agents de ISAME.
- si des erreurs sont détectées lors du filtrage d'un message, ou si aucun destinataire ne peut être identifié, en avertir l'expéditeur.
- rediriger les messages échangés entre les agents de ISAME.
- informer les agents de l'exclusion ou du retrait d'un agent enregistré dans ISAME.

Tableau 4.3. Énoncés performatifs de base pour les agents ∈ IA.

Énoncé	*Quand*	*Utilisation spécifique et valeurs obligatoires pour les paramètres*
advertise	*envoi*	Informer le RA_{IA} des messages performatifs compris par l'agent. Destinataire : « RA_{IA} ».
ask-about	*réception*	Demande d'information provenant d'un QA_{xn}. Les résultats devront être expédiés en un seul message. Expéditeur : un agent de type QA_{xn}.
deny	*réception*	Annulation d'une demande d'information de la part d'un QA_{xn}. Expéditeur : le QA_{xn} ayant précédemment posté la requête.
eos	*envoi*	Informer le QA_{xn} que ce message met fin à une série de messages répondant à une requête de type « stream-about ». Destinataire : le QA_{xn} ayant posté la requête.
error	*envoi* *réception*	
register	*envoi*	Enregistrement auprès de l'agent RA_{IA}. Destinataire : « RA_{IA} ».
reply	*envoi*	Retour d'information à un UA_x en réponse à une requête d'information. Destinataire : le UA_x ayant posté la requête.
sorry	*envoi* *réception*	
stream-about	*réception*	Demande d'information provenant d'un QA_{xn}. Les résultats proviendront du IA_y dans plusieurs messages dont le dernier sera de type « eos ». Expéditeur : un agent de type QA_{xn}.
subscribe	*réception*	Demande d'information de type « ask-about » ou « stream-about » en provenance d'un QA_{xn}. Cette requête est automatiquement de type persistant et demeure valide tant que l'expéditeur n'envoie pas de message de type « deny » pour en annuler l'effet. Expéditeur : un agent de type QA_{xn}.

Tableau 4.3. Énoncés performatifs de base pour les agents ∈ IA (suite).

Énoncé	*Quand*	*Utilisation spécifique et valeurs obligatoires pour les paramètres*
tell	*envoi*	L'agent informe l'environnement de sa connexion ou déconnexion temporaire (l'agent demeure enregistré auprès de l'environnement). L'agent informe l'environnement d'information devant être prise en compte dans son profil : formats de données manipulés, description textuelle des sources exploitées, etc. Destinataire : « RA_{QA} ».
	réception	Réception d'information concernant l'environnement (ex. : l'expulsion d'un agent dont certains messages sont en traitement dans le système).
transport-address	*envoi*	Le IA_y informe le RA_{IA} de l'adresse à laquelle il peut être rejoint. Destinataire : « RA_{IA} ».
unregister	*envoi*	Désistement auprès de l'agent RA_{IA}. Destinataire : « RA_{IA} ».

Autres savoir-faire :

- créer un QA_{xn} pour chacune des nouvelles requêtes d'information reçues.
- nettoyer l'environnement de toute trace d'agent exclus par l'un des registraires en informant tous les autres agents de cette exclusion.

Le Tableau 4.4 dresse une liste des énoncés performatifs manipulés par les agents de ce type ainsi que leur contexte d'utilisation.

E. Les agents RA_{UA}, RA_{QA}, RA_{IA}

Entrée en activités :

Quand : À la création de l'environnement ISAME.

Fin des activités :

Quand : À la destruction de l'environnement ISAME.

Tableau 4.4. Énoncés performatifs de base pour les agents de type CBA.

Énoncé	*Quand*	*Utilisation spécifique et valeurs obligatoires pour les paramètres*
ask-about	*envoi*	Demande d'information auprès d'un agent registraire concernant l'un des agents inscrit. Destinataire : un agent registraire.
broker-one	*réception*	Réception de la part d'un UA_x d'une requête d'information devant être prise en charge par un agent responsable. Expéditeur : un agent de type UA_x.
error	*Envoi* *réception*	
sorry	*envoi* *réception*	
tell	*envoi* *réception*	Informer les agents actifs de ISAME du retrait ou de l'expulsion d'un agent. Destinataires : tous les agents. Réception d'un message de la part d'un agent registraire informant du retrait ou de l'expulsion d'un agent. Expéditeur : un agent registraire.

Lieu de résidence :

Sur le serveur de l'environnement ISAME.

Savoir-faire :

Implanté par envoi ou réception de messages :

- accueillir les nouveaux agents désirant s'enregistrer auprès de l'environnement.
- s'informer auprès du commis de communication de l'état de l'agent et maintenir cet état à jour dans sa base de connaissances.

- le RA_{UA} doit maintenir à jour le profil de l'utilisateur représenté par chacun des UA_x qu'il connaît selon les informations fournies par ce UA_x et son commis.
- le RA_{IA} doit maintenir à jour le profil de la source représentée par chacun des IA_y qu'il connaît selon les informations fournies par son commis, les UA_x et les QA_{xn}.
- le RA_{QA} doit maintenir à jour le profil de la source représentée par chacun des QA_{xn} qu'il connaît selon les informations fournies par son commis.

Autres savoir-faire :

- maintenir à jour les informations décrivant les agents enregistrés : profil, nom, adresse, état, description textuelle, etc.
- mettre à la disposition des nouveaux agents une boîte aux lettres avec commis associé dans le CA.
- détruire la boîte aux lettres et le commis d'un agent expulsé ou démissionnant.
- exclure de la liste d'agents enregistrés tout agent dont l'état est désuet.
- informer le CBA de l'exclusion d'un agent.

Le Tableau 4.5 dresse une liste des énoncés performatifs manipulés par les agents de ce type ainsi que leur contexte d'utilisation.

Tableau 4.5 Énoncés performatifs de base pour les agents ∈ RA.

Énoncé	***Quand***	***Utilisation spécifique et valeurs obligatoires pour les paramètres***
advertise	*réception*	Réception d'un message d'information annonçant les énoncés performatifs pouvant être traités par l'agent expéditeur. Expéditeur : l'un des agents étant sous la responsabilité de cet agent registraire.
ask-about	*réception*	Réception de la part du CBA d'une requête d'information concernant un agent inscrit dans l'environnement. Réception de la part du SFA ou du RFA d'une requête d'un profil relié à un agent (ex. : le profil d'un IA_y). Expéditeur : « CBA ».
deny	*réception*	Annulation d'une demande d'information de la part du SFA, RFA ou CBA. Expéditeur : l'agent ayant précédemment posté la requête.
error	*envoi* *réception*	
register	*réception*	Réception d'une demande d'enregistrement de la part d'un agent du type surveillé par le registraire. Expéditeur : l'un des agents étant sous la responsabilité du ce registraire.
reply	*envoi*	Retour d'information au SFA ou RFA ou CBA. Destinataire : « RFA » ou « SFA » ou « CBA ».
sorry	*envoi* *réception*	
tell	*envoi* *réception*	Retour d'information aux requêtes reçues des agents CBA, SFA ou RFA. Confirmer à un agent son enregistrement auprès de ISAME. Informer le CBA du désistement ou de l'expulsion d'un agent. Destinataire : « CBA », « RFA » ou « SFA » ou l'agent enregistré. Réception d'information concernant les agents sous la responsabilité du registraire, ou réception de la part du CBA d'un message informant de l'expulsion d'un agent. Expéditeur : le CBA ou un agent de type UA_x, QA_{xn} ou IA_y.

Tableau 4.5 Énoncés performatifs de base pour les agents ∈ RA (suite).

Énoncé	*Quand*	*Utilisation spécifique et valeurs obligatoires pour les paramètres*
unregister	*réception*	Réception d'une annulation d'enregistrement de la part d'un agent du type surveillé par ce registraire. Expéditeur : un agent du type sous la responsabilité de ce registraire.

F. L'agent SFA

Entrée en activités :

Quand : À la création de l'environnement ISAME.

Fin des activités :

Quand : À la destruction de l'environnement ISAME.

Lieu de résidence :

Sur le serveur de l'environnement ISAME.

Savoir-faire :

Implanté par envoi ou réception de messages :

- déterminer l'ensemble des IA_y devant être utilisés pour une requête d'information étant donné le profil de chacun des IA_y, le profil de l'utilisateur ayant exprimé la requête, ainsi que le sous-ensemble de IA_y auxquels l'utilisateur a accès.
- communiquer avec les agents registraires pour obtenir les profils des utilisateurs et des IA_y.
- informer le QA_{xn} associé à la requête du sous-ensemble de IA_y recommandé.

Le Tableau 4.6 dresse une liste des énoncés performatifs manipulés par les agents de type SFA ainsi que leur contexte d'utilisation.

G. L'agent RFA

Entrée en activités :

Quand : À la création de l'environnement ISAME.

Fin des activités :

Quand : À la destruction de l'environnement ISAME.

Lieu de résidence :

Sur le serveur de l'environnement ISAME.

Savoir-faire :

Implanté par envoi ou réception de messages :

- filtrer les résultats de la recherche d'information selon le profil de chacun des IA_y ayant donné des résultats et le profil de l'utilisateur ayant exprimé la requête.
- formater les résultats selon le format connu pour cette requête ou le format par défaut indiqué dans le profil de l'utilisateur.
- communiquer avec les registraires pour obtenir les profils des UA_x et des IA_y.
- retourner au QA_{xn} les résultats de filtrage et de formatage d'information.

Le Tableau 4.7 dresse une liste des énoncés performatifs manipulés par les agents de ce type ainsi que leur contexte d'utilisation.

La Figure 4.3 montre schématiquement comment les différents types de messages mentionnés dans cette section sont acheminés de l'expéditeur au destinataire par l'entremise du CBA.

Tableau 4.6 Énoncés performatifs de base pour les agents de type SFA.

Énoncé	***Quand***	***Utilisation spécifique et valeurs obligatoires pour les paramètres***
ask-about	*envoi*	Demander au RA_{IA} la liste des IA_y disponibles. Demander au RA_{IA} le profil d'un agent IA_y. Demander au RA_{UA} le profil d'un agent utilisateur. Destinataire : « RA_{IA} » ou « RA_{UA} ».
deny	*envoi* *réception*	Annuler une demande d'information auprès des agents RA_{IA} ou RA_{UA}. Destinataire : « RA_{IA} » ou « RA_{UA} ». Annulation d'une demande de recommandation de la part d'un QA_{xn}. Expéditeur : le QA_{xn} ayant précédemment posté la requête.
error	*envoi* *réception*	
recommend-all	*réception*	Demande de recommandation de IA_y de la part d'un QA_{xn}. Expéditeur : un agent de type QA_{xn}.
reply	*envoi* *réception*	Envoi des recommandations à un QA_{xn} pour la liste des IA_y à utiliser pour une requête donnée. Destinataire : un agent de type QA_{xn}. Réception des informations demandées aux registraires concernant les profils et la liste des IA_y, ou le profil d'un UA_x. Expéditeur : « RA_{IA} » « RA_{UA} ».
sorry	*envoi* *réception*	

Tableau 4.7 Énoncés performatifs de base pour les agents de type RFA

Énoncé	*Quand*	*Utilisation spécifique et valeurs obligatoires pour les paramètres*
ask-about	*envoi*	Demander au RA_{IA} le profil d'un agent IA_y. Demander au RA_{UA} le profil d'un UA_x. Destinataire : « RA_{IA} » ou « RA_{UA} ».
deny	*envoi* *réception*	Annuler une demande d'information auprès des agents RA_{IA} ou RA_{UA}. Destinataire : « RA_{IA} » ou « RA_{UA} ». Annulation d'une demande d'évaluation de la part d'un QA_{xn}. Expéditeur : le QA_{xn} ayant précédemment posté la requête.
error	*envoi* *réception*	
evaluate	*réception*	Réception d'une demande de filtrage et de formatage d'information de la part d'un QA_{xn}. Expéditeur : un agent de type QA_{xn}.
reply	*envoi* *réception*	Envoi du résultat de filtrage à un QA_{xn}. Destinataire : un agent de type QA_{xn}. Réception des informations demandées aux registraires concernant les profils et la liste des IA_y. Expéditeur : « RA_{IA} » ou « RA_{UA} ».
sorry	*envoi* *réception*	

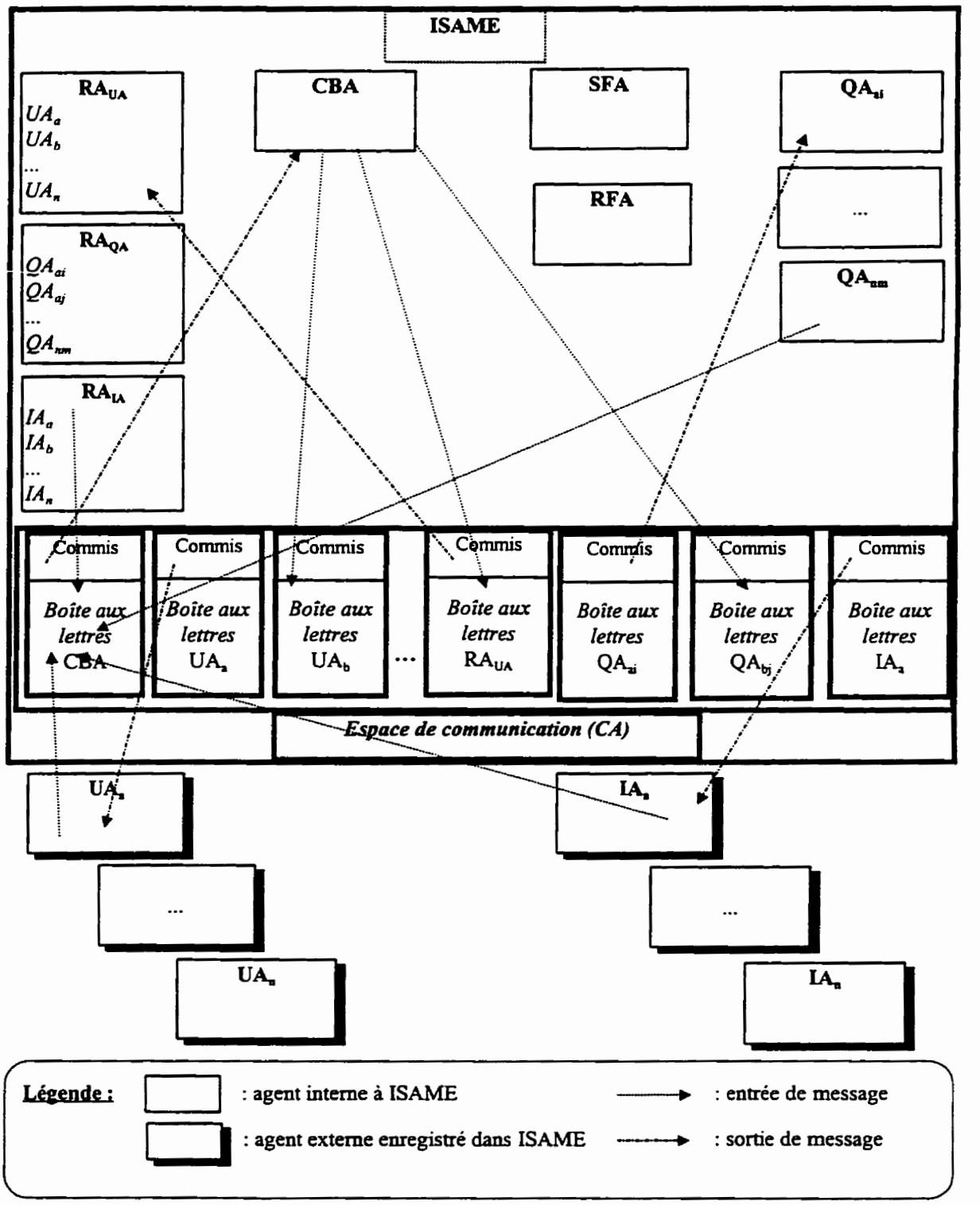


Figure 4.3. Échange de messages entre les agents de l'architecture ISAME

4.4 Activités de ramasse miettes

Afin d'éviter que l'environnement ISAME s'alourdisse d'éléments devenus désuets, un mécanisme de « ramasse miettes » (garbage collection) multi facettes doit être mis en place. Ce mécanisme, qui doit opérer de façon continue, vise à s'assurer que les événements suivants sont captés et contrés par une action appropriée, de façon à ce que les ressources de l'environnement soient toujours utilisées au meilleur escient :

- un agent ne peut plus être rejoint et ses messages s'accumulent ou dorment dans sa boîte aux lettres et consomment de l'espace;
- la boîte aux lettres et le commis dédiés à un agent disparu demeurent actifs et dépensent inutilement du temps machine ;

un agent n'est plus actif dans l'environnement, mais des messages qu'il avait envoyés avant sa disparition demeurent en traitement par d'autres agents, dépensant ainsi espace et temps machine.

Afin de couvrir l'ensemble de ces événements indésirables, les mécanismes de ramasse miettes sont basés sur des activités de surveillance de l'état des agents. Les types d'agents faisant l'objet d'une surveillance sont au nombre de trois, soient les UA_x, QA_{xn} et IA_y. Les agents de type SA, étant éveillés en permanence dans l'environnement, sont exempts de surveillance. Le scénario de surveillance est basé sur l'ensemble des états permis pour chacun des types d'agents. Cette surveillance est exercée par les registraires, en collaboration avec les commis rattachés aux boîtes aux lettres de chacun des agents. Avec l'hypothèse que le commis travaille pour un agent donné mais demeure

fidèle aux consignes du registraire l'ayant créé, on peut demander au commis de rapporter tout signe d'inaccessibilité de la part de son agent. Ainsi, le registraire est en position de déterminer à quel moment un agent devient un cas «louche», et peut mettre en place un système de surveillance ou de vigie. Cette vigie consiste en un compteur de la durée d'inaccessibilité d'un agent et d'une alarme lorsque le seuil de tolérance est dépassé. À ce moment, le registraire peut prendre action face au délit de l'agent, soit en mettant en place un deuxième niveau de vigie, soit en expulsant l'agent et en nettoyant l'environnement de toute trace laissée par cet agent. Les Figures 4.4 à 4.6 montrent les sous-ensembles de transitions possibles pour les agents visés par la surveillance, soient ceux de types UA_x, QA_{xn} et IA_y. Les Tableaux 4.8 à 4.10 fournissent les détails quant aux causes et effets de chacune des transitions possibles.

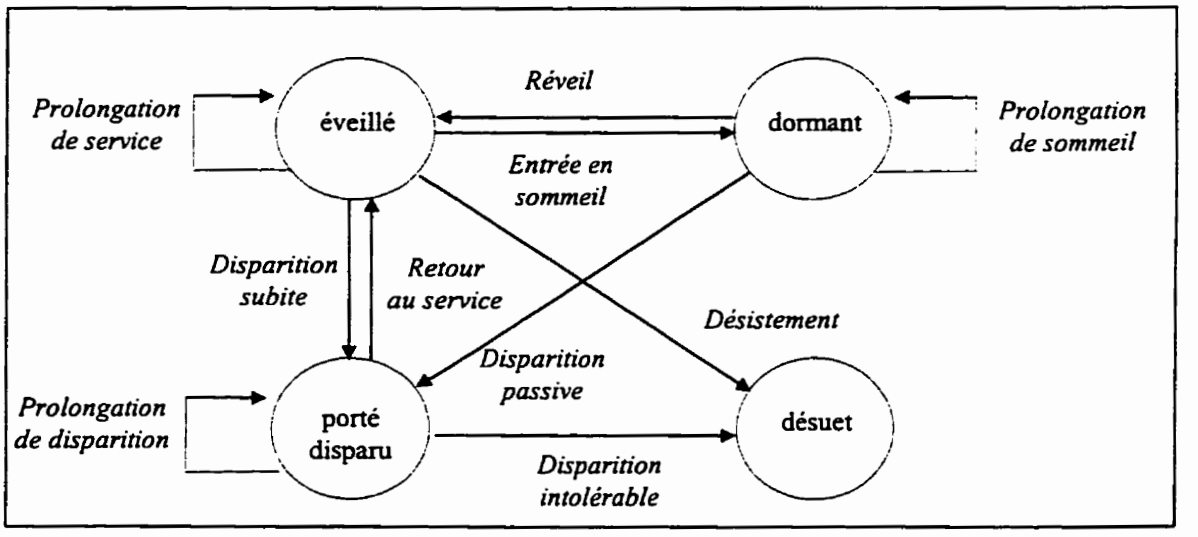


Figure 4.4. Automate de transitions d'états pour les agents dans UA.

Tableau 4.8 Causes et effets des transitions d'états pour les agents dans UA.

Transition	***Causes de la transition***	***Effets de la transition***
Entrée en sommeil	Le UA_x transmet un message de type « tell » informant le RA_{UA} de sa déconnexion temporaire.	■ La vigie de sursis 1 est mise en place par Le RA_{UA}. ■ Le commis est mis en attente.
Disparition subite	Le UA_x ne répond pas aux appels du commis et le commis en avertit le RA_{UA}.	■ La vigie de sursis 2 est débutée. ■ Le commis demeure en attente.
Réveil ou Retour au service	Le UA_x transmet un message de type « tell » informant le RA_{UA} de sa connexion.	■ La vigie de sursis 1 ou 2 est annulée. ■ Le commis débute la transmission des messages.
Disparition passive	La vigie 1 sonne l'alerte.	La vigie de sursis 2 est débutée.
Désistement ou Disparition intolérable	*Désistement* : Le UA_x transmet un message de type « tell » informant le RA_{UA} de sa déconnexion permanente. *Disparition intolérable* : La vigie 2 sonne l'alerte.	■ Le RA_{UA} expulse l'agent. ■ La boîte aux lettres et le commis sont détruits. ■ L'environnement est nettoyé des traces de l'agent.

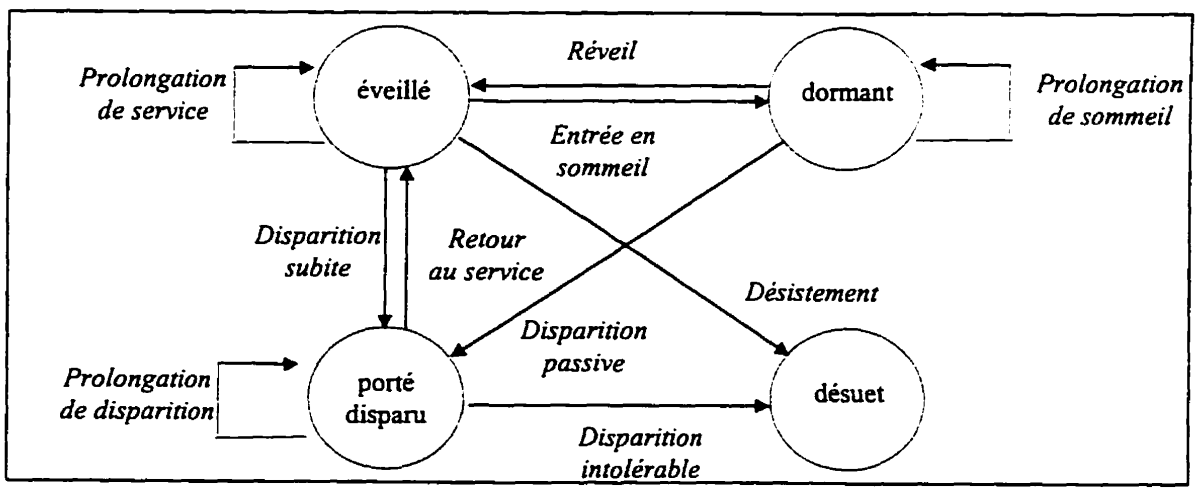


Figure 4.5. Automate de transitions d'états pour les agents dans IA.

Tableau 4.9 Causes et effets des transitions d'états pour les agents dans IA.

Transition	***Causes de la transition***	***Effets de la transition***
Entrée en sommeil	Le IA_y transmet un message de type « tell » informant le RA_{IA} de sa déconnexion temporaire.	■ La vigie de sursis 1 est mise en place par le RA_{IA}. ■ La boîte aux lettres est vidée et bloquée et le commis est mis en attente.
Disparition subite	Le IA_y ne répond pas aux appels du commis et le commis en avertit le RA_{IA}.	■ La vigie de sursis 2 est débutée. ■ La boîte aux lettres demeure bloquée et le commis demeure en attente.
Réveil ou Retour au service	Le IA_y transmet un message de type « tell » informant le RA_{IA} de sa connexion.	■ La vigie de sursis 1 ou 2 est annulée. ■ La boîte aux lettres est débloquée et le commis débute la transmission des messages.
Disparition passive	La vigie 1 sonne l'alerte.	La vigie de sursis 2 est débutée.
Désistement ou Disparition intolérable	*Désistement* : Le UA_x transmet un message de type « tell » informant le RA_{UA} de sa déconnexion permanente. *Disparition intolérable* : La vigie 2 sonne l'alerte.	■ Le RA_{IA} expulse l'agent. ■ La boîte aux lettres et le commis sont détruits. ■ L'environnement est nettoyé des traces de l'agent.

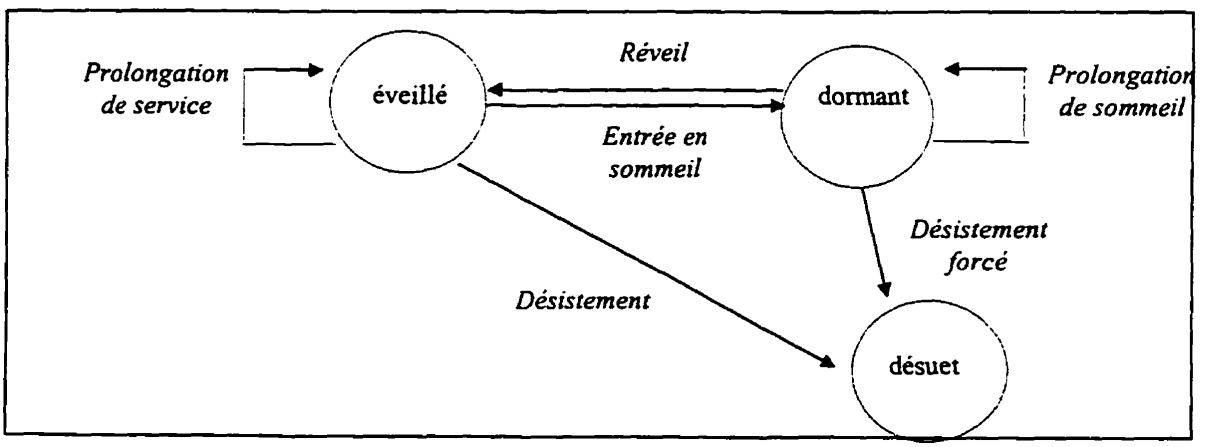


Figure 4.6. Automate de transitions d'états pour les agents dans QA.

Tableau 4.10. Causes et effets des transitions d'états pour les agents dans QA.

Transition	***Causes de la transition***	***Effets de la transition***
Entrée en sommeil	■ Le QA_{xn} est de type persistant et vient de terminer un cycle. ■ Le QA_{xn} transmet un message de type « tell » informant le RA_{QA} de sa déconnexion temporaire.	Le commis est mis en attente.
Réveil	■ Le QA_{xn} de type persistant doit débuter un nouveau cycle de requête. ■ Le QA_{xn} transmet un message de type « tell » informant le RA_{QA} de sa connexion.	Le commis débute la transmission des messages.
Désistement forcé	L'agent UA_x auquel la requête du QA_{xn} est associée quitte l'environnement ISAME.	■ Le RA_{QA} expulse l'agent. ■ La boîte aux lettres et le commis sont détruits. ■ L'environnement est nettoyé des traces de l'agent.
Désistement	■ La requête ponctuelle est terminée, la requête persistante est annulée. ■ Le QA_{xn} envoie un message de type « tell » informant le RA_{QA} de sa déconnexion permanente.	■ Le RA_{QA} expulse l'agent. ■ La boîte aux lettres et le commis sont détruits. ■ L'environnement est nettoyé des traces de l'agent.

CHAPITRE 5

ÉVALUATION DE L'ARCHITECTURE

Les travaux visant l'implantation de l'architecture présentée dans les chapitres 3 et 4 ont été concentrés sur la mise en place du coeur de l'environnement ISAME. En somme, le but principal de cette implantation fut de démontrer la faisabilité d'une telle architecture, ainsi que sa souplesse quant à l'aspect de services reliés à la recherche d'information par agents intelligents. Ce chapitre décrit, dans un premier temps, le sous-ensemble de l'architecture sélectionné pour l'implantation, une partie du modèle objet élaboré et mis en place, et l'environnement de cette implantation. Il présente ensuite un scénario concret d'utilisation de l'architecture, ainsi que les résultats obtenus. Il se termine par une évaluation détaillée de l'ensemble de l'architecture ainsi qu'une comparaison avec trois architectures connues.

5.1 Choix d'implantation

L'architecture présentée au cours des chapitres précédents tente de cerner l'ensemble des besoins relatifs à la recherche intelligente d'information par des agents autonomes. Bien qu'il soit souhaitable que tous ces éléments soient mis en place pour fournir un environnement optimal de travail pour tous les agents impliqués, le coeur de ce système, soit les agents de soutien (le CBA et les RAs), la communication par messages et les activités de ramasse miettes demeurent les pièces maîtresses de

l'architecture. Les travaux d'implantation ont donc été centrés sur ces éléments, lesquels sont décrits à la section 5.1.1. La section 5.1.2 présente l'environnement logiciel et machine avec lequel la programmation et les tests de mise en oeuvre ont été effectués.

5.1.1 Sous-ensemble de concepts implantés

Le sous-ensemble des concepts choisis pour fin d'implantation et de test de l'architecture proposée pour ISAME comprend l'aspect services des bibliothèques virtuelles, l'aspect manipulation de contenu représenté principalement par les agents utilisateurs et information n'ayant été implantés que très partiellement. Ce sous-ensemble a été sélectionné dans le but de permettre une évaluation réaliste de la souplesse reliée à l'architecture, c'est-à-dire :

1. démontrer que ISAME constitue un point d'accès unique à un ensemble de services et contenus de bibliothèque virtuelle;
2. évaluer si les agents utilisateurs et les agents information demeurent totalement indépendants les uns des autres, tant dans leur mode d'implantation que dans leur disponibilité, grâce au travail des agents de soutien;
3. démontrer que l'implantation des agents utilisateurs est indépendante de la complexité et de la diversité des services offerts par l'environnement ISAME ainsi que de leur implantation;
4. démontrer que les agents information et utilisateurs peuvent être inscrits auprès de l'environnement au fur et à mesure qu'ils deviennent disponibles;

5. démontrer que l'utilisation des boîtes aux lettres et des commis permet d'améliorer la livraison de messages aux agents non connectés en permanence;
6. démontrer que les activités de ramasse miettes permettent une certaine optimisation de l'environnement;
7. permettre une première évaluation de la méthode de distribution des tâches entre les agents de soutien et requêtes telles que prévue dans le modèle initial, ainsi qu'une évaluation de la qualité des services offerts par ISAME aux agents utilisateurs et information.

Les concepts mis en place dans cette phase du projet peuvent être regroupés en quatre grandes catégories : *langages et ontologies, types d'agent et messages associés, mécanismes de communication, et activités de ramasse miettes*. Pour chacune de ces catégories, une description des éléments sélectionnés ainsi que des modèles objets partiels des classes associées suivra. Une version plus complète de ces modèles basés sur la méthodologie OMT (Rumbaugh et al., 1991) peut être trouvée à l'Annexe F.

1. Langages et ontologies :

Les langages *KQML* et *ISAME-L* ainsi que les ontologies *Agent* et *VirtualLibrary* sont complètement supportés par l'implantation actuelle de ISAME. Ces langages et ontologies sont mis en oeuvre par une hiérarchie de classes, présentées à la Figure 5.1. Chacune de ces classes constitue un analyseur. Les analyseurs des langages *KQMLMessage* et *ISAMEMessage* permettent de valider la syntaxe des messages

échangés par les agents ainsi que la sémantique des « enveloppes » des messages. Les analyseurs *AgentOntology* et *VirtualLibraryOntology* s'occupent de valider la sémantique du contenu des messages ISAME. Chacun des agents implantés comprend l'un et/ou l'autre des langages et ontologies, et filtre chacun des messages qu'il reçoit avec les langages et ontologies appropriés. L'interprétation proprement dite des messages ainsi que le contrôle des activités associées à chacun des messages reçus s'effectuent par les classes dérivées de *MessageInterpreter*, lesquelles font appel aux bons analyseurs de langages et d'ontologies au besoin.

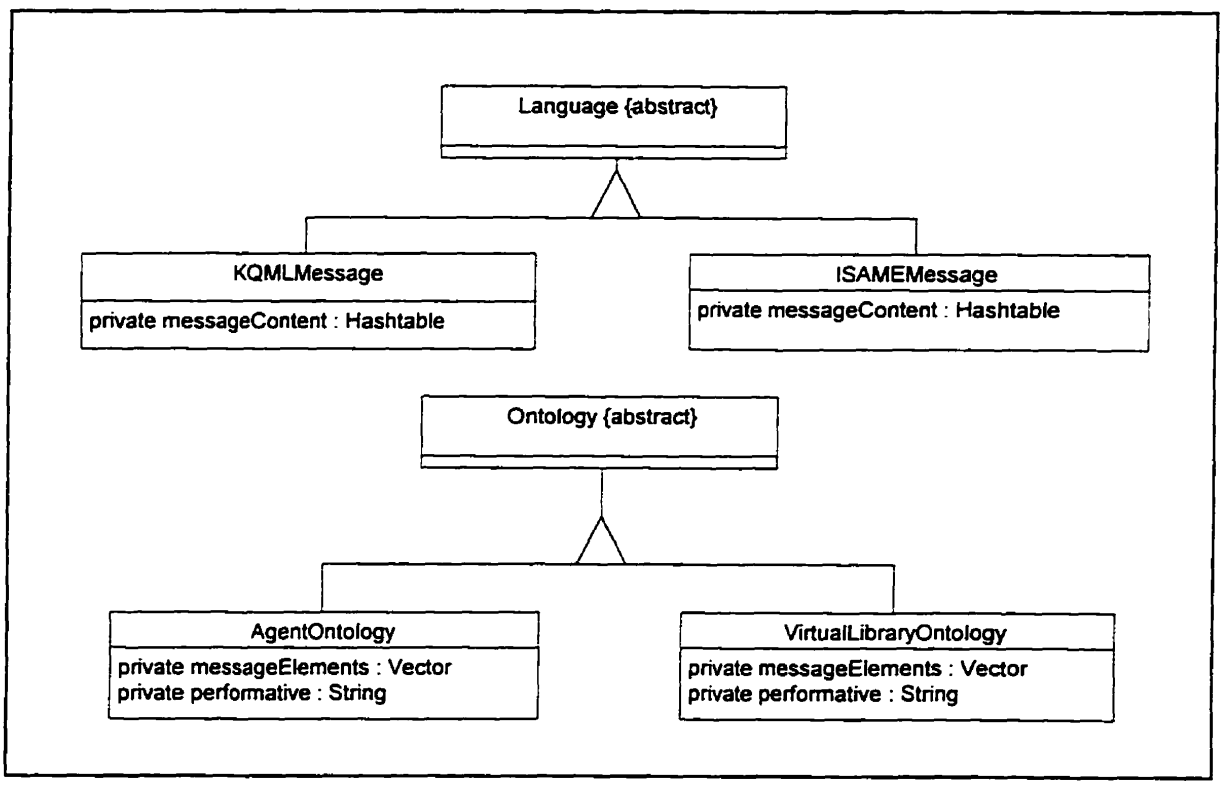


Figure 5.1. Hiérarchie des classes de langages et d'ontologies.

La classe abstraite *Language* de ISAME est tirée de la librairie d'objets fournie par JavaAgent. La classe *KQMLMessage* a été inspirée de celle fournie par JavaAgent

(Frost, 1995). Quelques modifications mineures ont été apportées à la classe originale, principalement dans le but de clarifier la lecture du programme et de modifier les validations effectuées sur la sémantique de chacun des messages KQML supportés par l'environnement ISAME (ISAME ne supporte qu'un sous-ensemble de messages KQML). La classe *ISAMEMessage* est elle aussi largement inspirée de la classe *KQMLMessage*, mais se distingue par le support de notions ensemblistes. Ainsi, les informations véhiculées présentés selon la syntaxe du langage ISAME-L sont exprimées soient par des sous-ensembles d'éléments, soit par des éléments atomiques.

2. Types d'agent et messages associés :

À l'exception des agents de soutien RFA et SFA, toutes les classes d'agents prévues à l'architecture sont présentes à l'heure actuelle dans ISAME. Du côté des agents de soutien, les trois agents dans RA, l'agent CBA et un agent QA_{xn} générique sont totalement implantés, à l'exception de la manipulation des requêtes persistantes chez les QA_{xn}. Le CBA constitue, comme il se doit, le point d'entrée unique à l'environnement ISAME pour tous les agents externes, puisque l'adresse *socket* publiée pour entrer en communication avec l'environnement est celle de la réception des messages pour le CBA. Le RAIA joue le rôle du SFA en offrant le service de recommandation de sources d'information aux agents QA_{xn}. Cette recommandation, grandement simplifiée comparativement à ce qu'il est prévu d'intégrer dans l'agent SFA, s'effectue sur la réception d'un message de type *recommend-all*. Finalement, en l'absence du RFA pour

effectuer un filtrage efficace des résultats de recherche d'information, les QA_{xn} complètent eux-mêmes un filtrage simplifié.

Des versions génériques des agents utilisateurs et information sont disponibles dans l'environnement. Ces agents génériques peuvent être dérivés pour créer des agents spécifiques aux besoins des utilisateurs ou sources d'information. Une version très simpliste d'un agent utilisateur, *SimpleUserAgent*, a été dérivée de la classe *UserAgent* afin de valider les mécanismes d'enregistrement et de désistement d'agents externes, ainsi que le cycle de vie des agents QA_{xn}. La Figure 5.2 présente la hiérarchie des classes d'agents.

La classe abstraite *MessageInterpreter* constitue un processus léger autonome (Thread) s'occupant d'interpréter et d'exécuter chacun des messages reçus par les agents. De nouvelles classes doivent être dérivées de *MessageInterpreter* pour chacune des classes d'agents afin de faire correspondre la manipulation du message au contexte et au rôle de chacun des agents. Ces interpréteurs constituent en fait le savoir-faire associé à chacun des agents. Plusieurs interpréteurs peuvent être actifs en parallèle pour un même agent, permettant ainsi à ce dernier de compléter plusieurs tâches simultanément. Chaque interpréteur possède son propre agenda d'activités qui est régi par la séquence d'activités associée au type de message traité. À la réception d'un message, l'agent effectue une série de validations : il vérifie la syntaxe du message KQML, vérifie si le langage, l'ontologie et l'énoncé performatif sont valides dans son contexte, puis vérifie si le message constitue une réponse à un message expédié plus tôt (auquel cas le message est

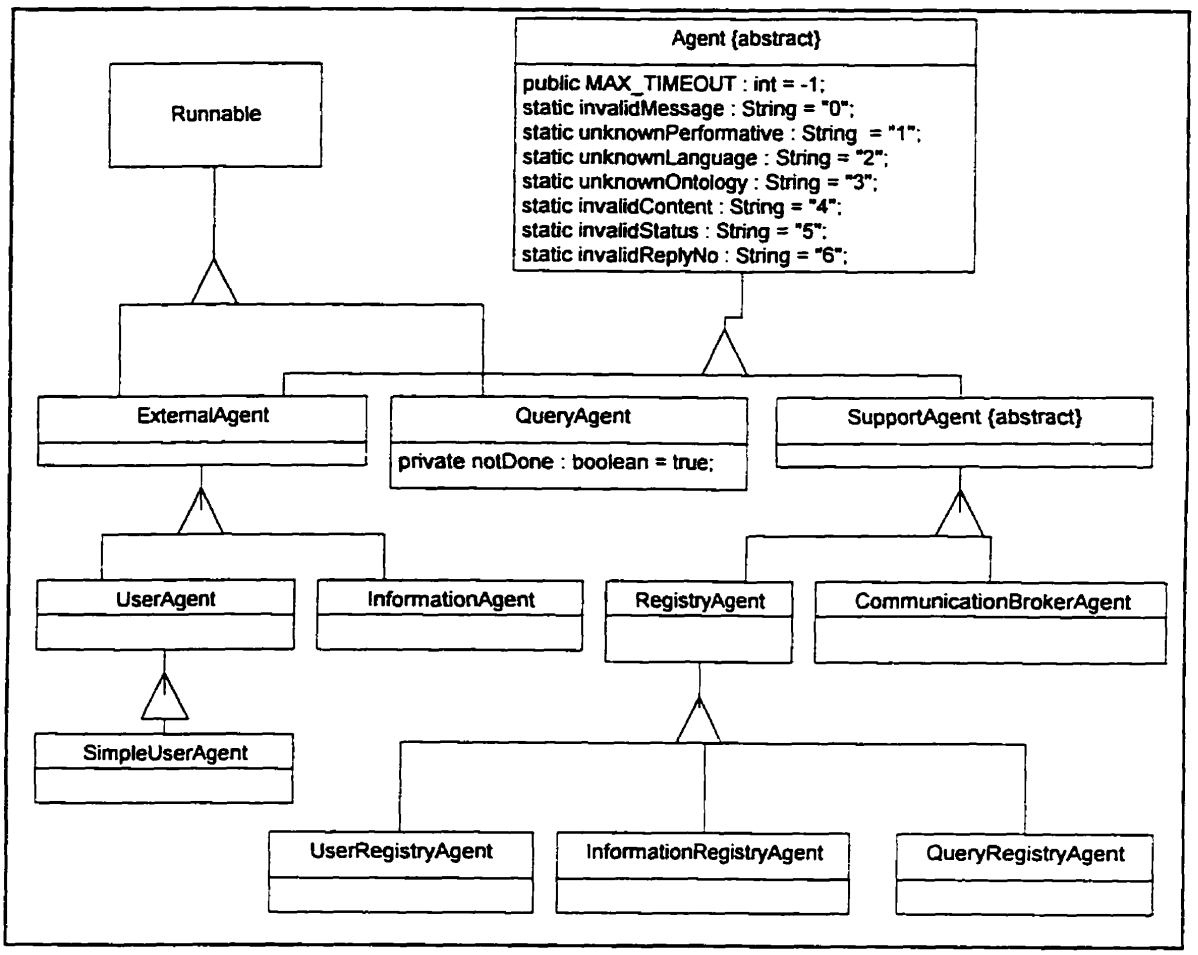


Figure 5.2. Hiérarchie des classes d'agents.

acheminé vers l'interpréteur en attente d'une réponse) ou s'il s'agit d'un message indépendant (auquel cas une nouvelle instance d'interpréteur sera créée pour prendre soin du message). L'ensemble des interpréteurs de messages présentés à la Figure 5.3 effectuent l'interprétation des messages par des méthodes procédurales associées (ex.: *interpretTellMessage*, *interpretBorkerOneMessage*). Tous les interpréteurs travaillant pour un agent lui sont associés par l'entremise d'un objet *MessageInterpreterQueue* pour assurer une gestion efficace. Afin d'implanter un autre type d'interpréteur, tel qu'un interpréteur dont le travail est basé sur une base de règles, il suffit simplement de dériver

la nouvelle classe de la classe abstraite *MessageInterpreter*, de redéfinir les méthodes *interpretMessage* et *handleReply*, puis de générer le bon type de *MessageInterpreter* dans l'agent associé. Ce type d'interpréteur n'a pas été testé dans le cadre des travaux présentés ici.

Chacune des classes présentement dérivées de *MessageInterpreter* interprète une partie ou tous les messages prévus dans l'architecture pour la classe qui lui est associée. Le Tableau 5.1 identifie, pour chacune des classes dérivées de *MessageInterpreter*, les énoncés performatifs qui sont présentement manipulés ou non par elle.

3. Répartition des agents et objets entre le serveur et les machines clients :

Dans l'architecture ISAME telle que définie jusqu'à ce jour, la machine serveur de l'environnement regroupe les agents de soutien, les agents requêtes, les boîtes aux lettres et les commis. Chacun des agents utilisateurs et information ainsi que son gestionnaire de communication se trouve sur une machine client, soit celle choisie par le concepteur de l'agent. La répartition des éléments entre le serveur et les clients suit une règle très simple : l'environnement ISAME est un environnement offrant des services aux agents utilisateurs et information définis par des institutions ou des individus qui veulent profiter des services de ISAME de façon ponctuelle ou continue. La fiabilité de ce serveur doit donc être assurée par un organisme ou un individu qui en prend la responsabilité. Cependant, ISAME n'est pas un environnement « d'hébergement » pour les agents, ce qui est le cas dans les environnements multi-agents dans lesquels les agents sont mobiles, c'est-à-dire que les agents voyagent physiquement d'un serveur à l'autre à

sont mobiles, c'est-à-dire que les agents voyagent physiquement d'un serveur à l'autre à la recherche d'information. Puisque la communication et l'échange d'information entre les agents s'établissent par l'échange de messages, il n'est pas nécessaire pour tous les agents d'être centralisées sur une même machine. Cette distribution des agents entre le serveur et les clients évite la nécessité d'un « super serveur » possédant suffisamment de ressources pour répondre aux besoins de tous les agents, simplifie les besoins de sécurité sur le serveur, et permet d'implanter les agents externes directement sur la machine choisie par leur concepteur facilitant ainsi le contrôle et la maintenance de ces agents. Enfin, les objets utilisés par un agent, que ce soit ses analyseurs de langage et d'ontologie, ses interpréteurs de messages, et ses connaissances, résident sur la même machine que l'agent.

Bien que les agents de soutien et les agents requêtes implantés dans le cadre de l'expérimentation présentée ici résident tous sur la même machine, la communication entre les commis et les gestionnaires de communication de ces agents est entièrement basée sur une communication de type TCP/IP telle que présentée à la Figure 3.8. Ainsi, comme il se doit, tous les gestionnaires de communication expédient leurs messages au CBA par TCP/IP, et les commis communiquent avec les gestionnaires par TCP/IP aussi. Seul le CBA communique directement avec les commis par communication locale. Le protocole de communication TCP/IP, comparativement à la celui d'échanges de messages entre objets, présentait en fait le scénario le plus complet puisqu'il implique plus de délais entre l'expédition et la réception d'un message, et l'utilisation d'objets,

entre objets. D'autre part, il permet d'effectuer les premiers pas vers un serveur ISAME distribué sur plusieurs machines, ce qui pourrait constituer un avantage certain lorsque d'autres types d'agents de soutien et un plus grand nombre d'agents requêtes seront en fonction.

4. Mécanismes de communication :

Deux groupes de classes sont nécessaires à l'implantation des mécanismes de communication entre les agents. D'une part, la réception et l'envoi des messages doivent être mis en place à l'intérieur de chacun des agents. D'autre part, le système de commis et de boîtes aux lettres permet au CBA d'acheminer les messages en transit dans l'environnement ISAME vers les agents destinataires.

La classe *CommunicationCenter* regroupe les outils nécessaires à l'intérieur de chacun des agents pour la réception et l'envoi des messages. Ce centre de communication est composé de deux objets : un récepteur, *MessageReceiver*, lequel saisit les messages provenant de l'externe, et un expéditeur, *MessageSender*, qui expédie les messages vers l'extérieur. Lorsqu'un message est reçu par son *MessageReceiver*, il est inséré dans une file d'attente dans l'ordre FIFO (First In First Out), et un gestionnaire de messages associé à

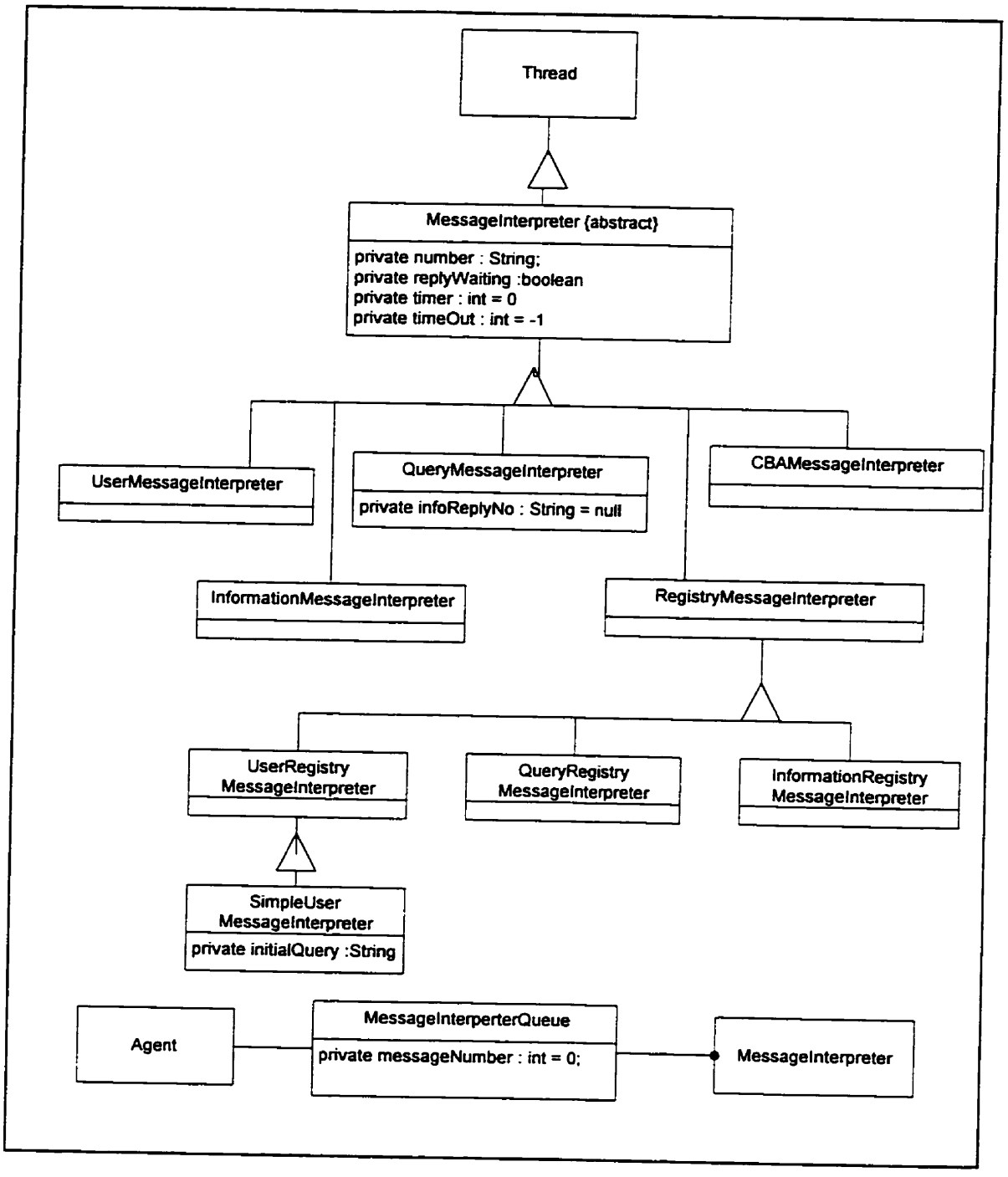


Figure 5.3. Hiérarchie des classes d'interpréteurs de messages et association avec la classe Agent par l'intermédiaire de la classe MessageInterpreterQueue.

Tableau 5.1. Manipulation actuelle d'énoncés performatifs par les interpréteurs de messages.

Classe d'interpréteur	*Énoncés performatifs manipulées*	*Énoncés performatifs non manipulées*
UserMessage Interpreter	envoi : error, register, sorry, unregister réception : error, sorry	envoi : advertise, ask-about, broker-one, deny, stream-about, subscribe, tell, transport-address réception : eos, reply, tell
SimpleUserMessage Interpreter	envoi : ask-about, broker-one, error, register, sorry, unregister réception : error, sorry	envoi : advertise, deny, stream-about, subscribe, tell, transport-address réception : eos, reply, tell
InformationMessage Interpreter	envoi : error, sorry, register, unregister réception : error, sorry	envoi : advertise, reply, tell, transport-address réception : ask-about, deny, stream-about, subscribe, tell,
QueryMessage Interpreter	envoi : ask-about, deny, error, recommend-all, register, reply, sorry, unregister réception : ask-about, deny, error, reply, sorry, tell	envoi : eos, evaluate, stream-about, subscribe, tell réception : eos, stream-about, subscribe
CBAMessage Interpreter	envoi : ask-about, error, sorry, tell réception : broker-one, error, reply, sorry, tell,	

Tableau 5.1. Manipulation actuelle d'énoncés performatifs par les interpréteurs de messages (suite).

Classe d'interpréteur	*Énoncés performatifs manipulées*	*Énoncés performatifs non manipulées*
RegistryMessage Interpreter	envoi : error, reply, sorry, tell réception : ask-about, error, register, sorry, tell, unregister	réception : advertise, deny

l'agent, un *MessageHandler*, se charge de le retirer de la file et de le remettre à l'agent pour qu'il soit traité. Lorsqu'il reçoit un message de la part de son *MessageHandler*, l'agent effectue les vérifications nécessaires et l'achemine vers le bon *MessageInterpreter*, comme mentionné précédemment. Pour expédier un message, un agent l'insère dans la file d'attente des messages du *MessageSender*, lequel s'occupe de l'expédier. Afin d'effectuer une gestion efficace de la boucle de communication des messages par les agents, les classes *MessageReceiver*, *MessageSender* et *MessageHandler* forment des processus légers autonomes (Thread) fonctionnant en parallèle. Chacun des agents peut donc recevoir, traiter et expédier des messages de façon simultanée, tel qu'illustré à la Figure 5.4.

Différentes classes ont été dérivées des classes *MessageReceiver* et *MessageSender* afin de répondre aux besoins de tous les agents. Ainsi, tous les agents, à l'exception du CBA, utilisent un expéditeur de messages basé sur les *sockets*, *SocketMessageSender*, afin d'expédier leurs messages au CBA pour fin d'aiguillage. Pour les besoins du CBA, qui expédie les messages en les déposant directement dans les boîtes aux lettres des agents, la classe *CBAMessageSender* a été dérivée. De même, tous

les agents utilisent un récepteur de messages basé sur les *sockets*, *SocketMessageReceiver*. Chacun de ces récepteurs de messages effectue une écoute continue sur le *socket* associé à l'agent lors de sa création.

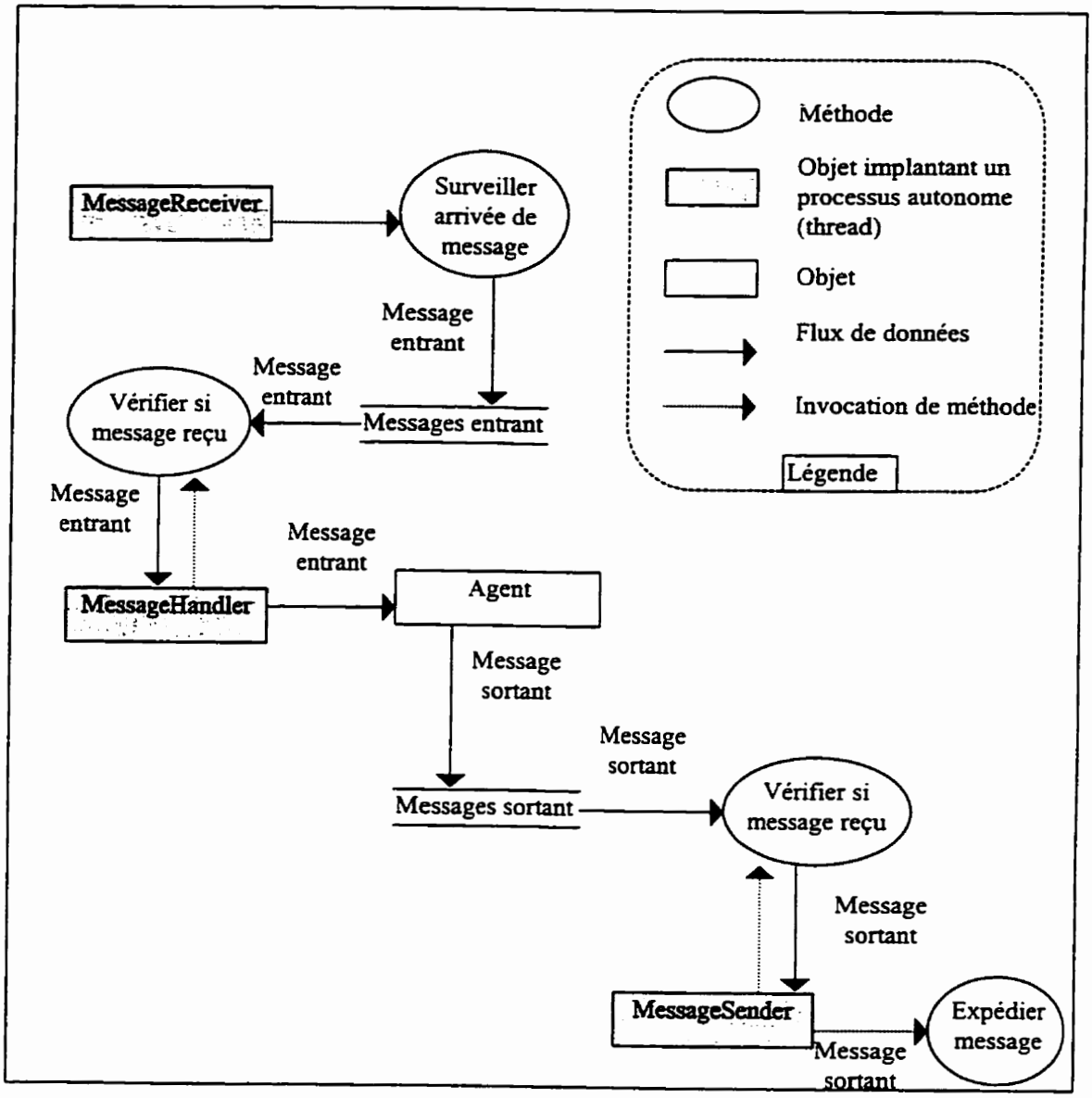


Figure 5.4. Réception, traitement et envoi de messages par les agents et leurs objets associés.

Pour fins de tests, ainsi que pour offrir plus de souplesse aux utilisateurs des librairies d'objets fournies dans ISAME, des versions locales, c'est-à-dire sans *socket* et

basées sur la communication directe entre les boîtes aux lettres et les centres de communication, ont été développées. Ces deux classes, *LocalMessageReceiver* et *LocalMessageSender*, ne peuvent être utilisées que dans le contexte où les objets du centre de communication et le commis de la boîte aux lettres résident sur la même machine, de façon à permettre l'échange de messages entre les objets par l'appel direct des méthodes de l'un et de l'autre. L'utilisation de ces versions locales permettrait d'éviter la sur-utilisation de *sockets* par l'ensemble des agents. La hiérarchie des classes permettant le flux des messages dans chacun des agents est présentée à la Figure 5.5.

Le second groupe de classes supportant l'échange de messages entre les agents est celui des boîtes aux lettres, ou *MailBox*, et des commis, ou *MessageClerk*. C'est par l'entremise de ces objets que le CBA aiguille les messages vers les agents. Rappelons que tous les messages échangés entre les agents de ISAME transitent par le CBA. Ceci évite à chacun des agents de connaître l'adresse, voire même l'existence, de chacun des autres agents avec lequel il doit entrer en contact. Cependant, ceci impose plus de travail au CBA. En effet, il doit gérer la file de messages de chacun des agents et l'expédition de ces messages. Afin d'éviter que le CBA n'ait à gérer lui-même ces processus, cette gestion est déléguée à une paire d'objets, soit *MailBox* et *MessageClerk*, le premier faisant office de dépôt de messages, et le deuxième, le commis, s'occupant des tâches de gestion proprement dites : filtrage des messages selon les critères des activités de ramasse miettes, gestion de la boîte aux lettres (inscription et retrait des messages), envoi des messages à l'agent destinataire. Toutes les instances d'objets *MailBox* et

MessageClerk créées résident sur le serveur de l'environnement ISAME avec les agents de soutien et les agents requêtes. Tout comme les récepteurs et expéditeurs de messages associés à l'agent, le *MessageClerk* est un processus léger autonome, assurant une redirection continue des messages. À l'exception du CBA, chacun des agents actifs dans ISAME a un *MessageClerk* travaillant pour lui. Deux classes sont dérivées de *MessageClerk* soit *SocketMessageClerk*, la version *socket* permettant une connexion avec le *SocketMessageReceiver* de l'agent, et *LocalMessageClerk*, version à utiliser lorsque l'agent utilise un *LocalMessageReceiver*.

Les classes pour faciliter le travail du CBA dans l'envoi de messages sont présentées à la Figure 5.6.

4. Activités de ramasse miettes :

Les activités de ramasse miettes, telles que décrites à la section 4.4, nécessitent divers processus et structures de données. Toutes ces structures de données ont été mises en place mais n'ont été que partiellement testées. Un nombre restreint de processus de ramasse miettes sont actifs à l'heure actuelle. De ces structures et processus, mentionnons :

- L'utilisation des variables d'état pour chacun des agents sont mises en application. L'ensemble des états valides et l'état courant sont conservés pour chacun des agents externes et les agents de QA dans les agents registraires. Chaque commis effectuent les validations nécessaires sur l'état de son agent associé avant d'insérer un nouveau message dans la boîte aux lettres. Cependant,

cet état n'est pas mis à jour et garde la valeur *éveillé ou AWAKE* jusqu'à ce que l'agent soit expulsé de l'environnement. Les systèmes de vigie n'ont donc pas été réalisés dans les registraires.

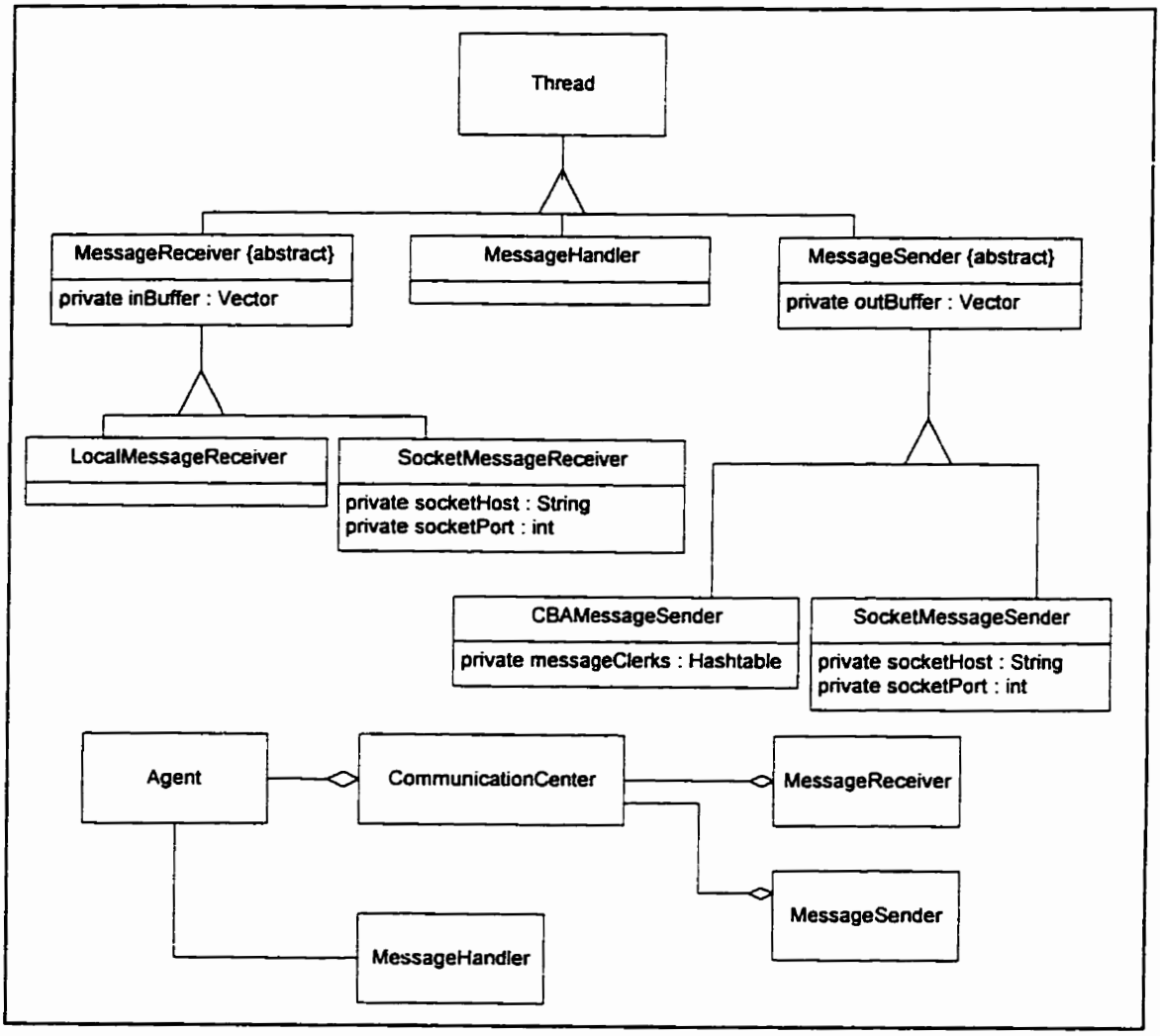


Figure 5.5. Hiérarchie et association des classes pour la réception et l'envoi de messages à l'intérieur de l'agent.

- Les mécanismes d'expulsion automatique et de nettoyage de messages lorsqu'un agent se désiste auprès de l'environnement fonctionnent. Sur réception d'un message *unregister*, chaque registraire élimine les informations qu'il

possède sur l'agent quittant, déclenche les activités d'élimination du commis et de la boîte aux lettres, puis expédie un message *tell* au CBA informant ce dernier que l'agent quittant a maintenant le statut *désuet* ou *DEAD*. Sur réception de ce message, le CBA le diffuse à tous les agents de RA et les agents de QA actifs dans l'environnement afin que ces derniers détruisent les messages qu'ils ont reçus de cet agent et qui sont présentement en traitement. Cependant, aucun traitement d'élimination des messages en provenance de l'agent quittant et déposés dans les boîtes aux lettres n'est effectué par les commis.

- Les processus de contrôle pour la durée de vie des messages en attente dans les boîtes aux lettres ainsi que pour la durée d'attente active d'un retour d'information par les interpréteurs de messages des agents sont tous actifs. Ainsi, les commis effectuent régulièrement une vérification sur les messages de la boîte aux lettres et éliminent les messages périmés. De plus, chaque interpréteur de messages utilise au moins deux conditions pour déterminer s'il continue à attendre un message en réponse à une requête. La première condition est la réception de cette réponse, et la deuxième est l'atteinte d'un délai maximum imposé par l'agent lors de la création de l'interpréteur.

5.1.2 Environnement logiciel et matériel

L'ensemble des classes de l'environnement ISAME, de l'agent utilisateur et des interfaces créées pour fins de tests ont été réalisées à l'aide des outils logiciels suivants :

- systèmes d'exploitation : **Windows 95.**

- langage de programmation : **Java.**
- outils de développements : Borland C++ 5.1, Symantec Visual Café 1.0, Java Development Kit (JDK) 1.0.1 et 1.1 de Sun Microsystems, librairies d'objets de JavaAgent.

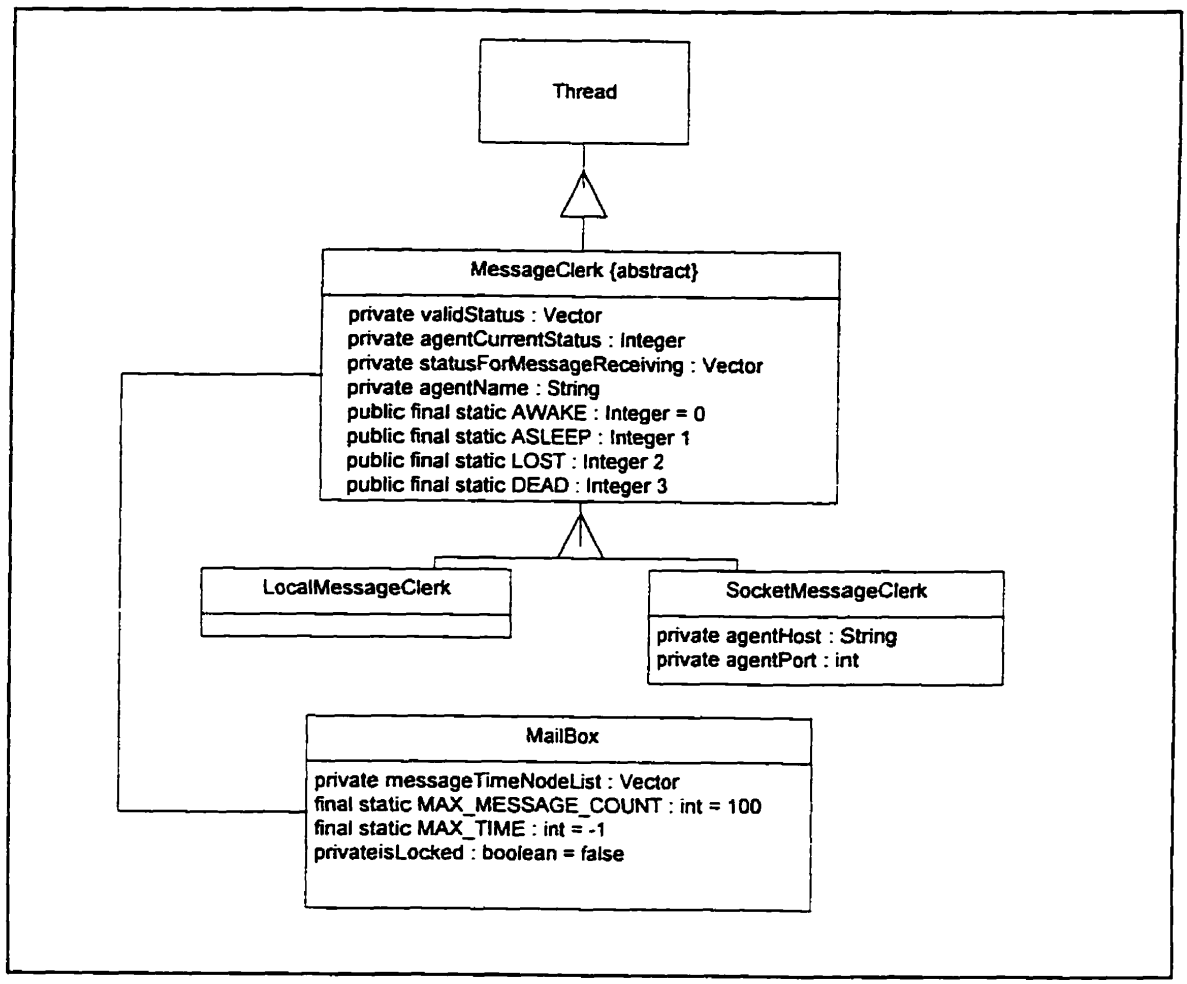


Figure 5.6. Hiérarchie et association des classes pour l'aiguillage des messages à partir de l'environnement ISAME.

L'environnement ISAME ainsi que ses agents ont été développés sur des machines de type PC reliées par un réseau de type Microsoft Windows NT. Le coeur de l'environnement ISAME (les agents de soutien, les agents requêtes, les boîtes aux lettres

et les commis) réside sur l'une de ces machines, le serveur de l'environnement ISAME. L'autre machine, qui agit comme client de l'environnement ISAME, est le lieu de résidence de l'agent utilisateur créé pour effectuer les tests. Tous les agents distribués sur le serveur et sur le ou les clients communiquent par des liens TCP/IP via *sockets*, le nombre des *sockets* pouvant être variables sur chacune des machines. Ce scénario est typique de l'implantation d'un environnement ISAME, puisque le cœur de l'environnement résidera toujours sur un serveur centralisé, et les agents utilisateurs et information résideront sur des machines clients.

Selon les standards internationaux, l'adressage des sockets TCP/IP est basé sur 16 bits et est réparti comme suit : les adresses 0 à 1023 sont réservées aux services connus de façon générale sur l'Internet (exemple : 80 est le port pour le WWW); les adresses 1024 à 4999 sont des adresses libres dont la machine se sert pour attribuer des ports éphémères aux processus clients; finalement, les adresses 5000 et plus sont utilisées comme adresses de serveur pour les nouveaux services (Steven, 1994). ISAME respecte ces normes. Il attribue des adresses entre 1024 et 4999 aux processus clients, c'est-à-dire pour les objets *SocketMessageClerk* et *SocketMessageSender*, et utilise des adresses dont les numéros sont de 5000 et plus pour les processus serveurs, c'est-à-dire pour les objets *SocketMessageReceiver*. Les caractéristiques du serveur utilisé pour le développement et les tests de l'architecture ISAME sont les suivantes :

- processeur : Pentium 166 Mz.
- mémoire vive : 16 Mb.

- carte réseau : 3Com EtherLinkXL COMBO 10Mb Ethernet Adapter.
- adresse physique : 206.167.88.156.
- adresses *sockets* réservées pour ISAME :
  - CBA, soit le point d'entrée de l'environnement ISAME : 5001.
  - RAUA, RAQA et RAIA : les adresses de 5002 à 5004.
- pour les fins de tests, 10 adresses *sockets* ont été réservées pour les agents ∈ QA : de 5005 à 5014. Ce nombre d'adresses pourrait être augmenté selon la capacité du serveur utilisé.

Les caractéristiques de la machine client utilisée sont :

- processeur : Pentium 90 Mz.
- mémoire vive : 16 Mb
- carte réseau : 3Com EtherLink III Bus Master PCI Ethernet Adapter.
- adresse physique : 206.167.88.146.
- adresse *socket* utilisée pour l'agent utilisateur de test : 5001.

5.2 Mise en oeuvre

Les objets présentement actifs dans ISAME permettent de compléter un certain nombre de scénarios d'interaction entre les agents : enregistrement et désistement d'agents utilisateur et information, prise en charge d'une requête utilisateur par l'environnement et, par conséquent, toute interaction inhérente à cette prise en charge. Les sous-sections qui suivent font un suivi détaillé des activités déclenchées par

certaines de ces interactions en présentant un scénario type et les résultats obtenus à la suite d'une session réelle dans l'environnement ISAME.

5.2.1 Élaboration d'un scénario

La mise en situation est la suivante : l'environnement ISAME est actif sur la machine serveur (les agents CBA, RAUA, RAIA et RAQA sont donc en attente de messages) et un nouvel agent utilisateur désire tirer profit des services offerts par ISAME pour effectuer une recherche d'information. Pour ce faire, l'agent utilisateur, que nous nommerons UA_X, devra compléter les étapes suivantes :

1. enregistrement auprès de l'environnement ISAME;
2. envoi de la requête utilisateur à l'environnement ISAME;
3. éventuellement, désistement auprès de l'environnement ISAME.

Bien que chacune de ces étapes puisse s'exprimer en une seule ligne, elles se décomposent toutes en une série d'activités et d'interactions entre les agents ISAME afin que soit complété l'ensemble des tâches et sous-tâches identifiées dans les Figures 3.2 à 3.4. Les Figures 5.7 à 5.11 illustrent, en termes d'échange de messages et de traitement par chacun des agents, une partie de la décomposition de chacune de ces étapes. Le contenu des messages échangés est fourni par la suite.

5.2.2 Détail des messages échangés par les agents

Tous les messages échangés dans le cadre des scénarios précédents sont présentés en détail dans cette section. Lorsqu'il s'agit d'un message expédié au CBA pour routage vers un agent autre que le CBA, le routage du message s'effectue en trois étapes :

1. envoi du message à l'environnement : Le *SocketMessageSender* de l'expéditeur fait parvenir le message au *SocketMessageReceiver* du CBA.
2. filtrage du message par le CBA : À la réception du message, le CBA effectue les validations syntaxiques et sémantiques KQML sur le message et lit la valeur associée au paramètre *receiver* afin de déterminer si ce destinataire possède une boîte aux lettres dans ISAME sur le serveur de l'environnement.
3. envoi du message au destinataire ou message d'erreur à l'expéditeur :

Cas 1 : Le destinataire ne possède pas de boîte aux lettres dans ISAME, ou encore des erreurs syntaxiques ou sémantiques sont trouvées, le CBA lit alors la valeur associée au paramètre *reply-with* du message. Si cette valeur existe, le CBA renvoi un message d'erreur à l'expéditeur avec un code d'erreur associé expliquant la raison du refus d'aiguillage. Présentement, si le paramètre *reply-with* est absent du message, le message est détruit. Dans le futur, le CBA pourrait expédier un message de type *Sorry* au destinataire avec comme valeur associée au paramètre *comment* le message n'ayant pu être expédié et un code d'erreur spécifiant l'impossibilité d'expédié le message.

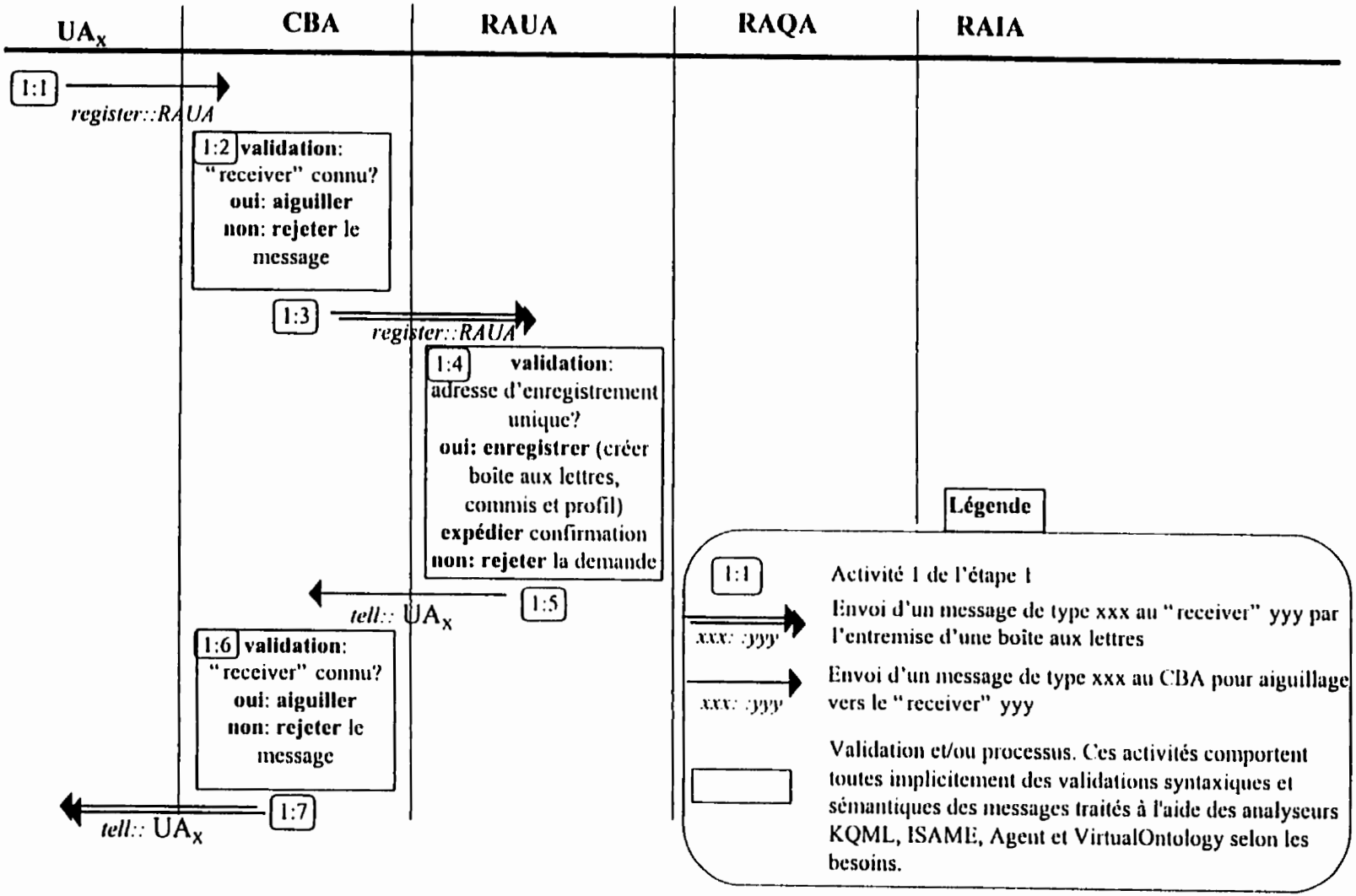


Figure 5.7. Étape 1 : Enregistrement du UA_x auprès de ISAME.

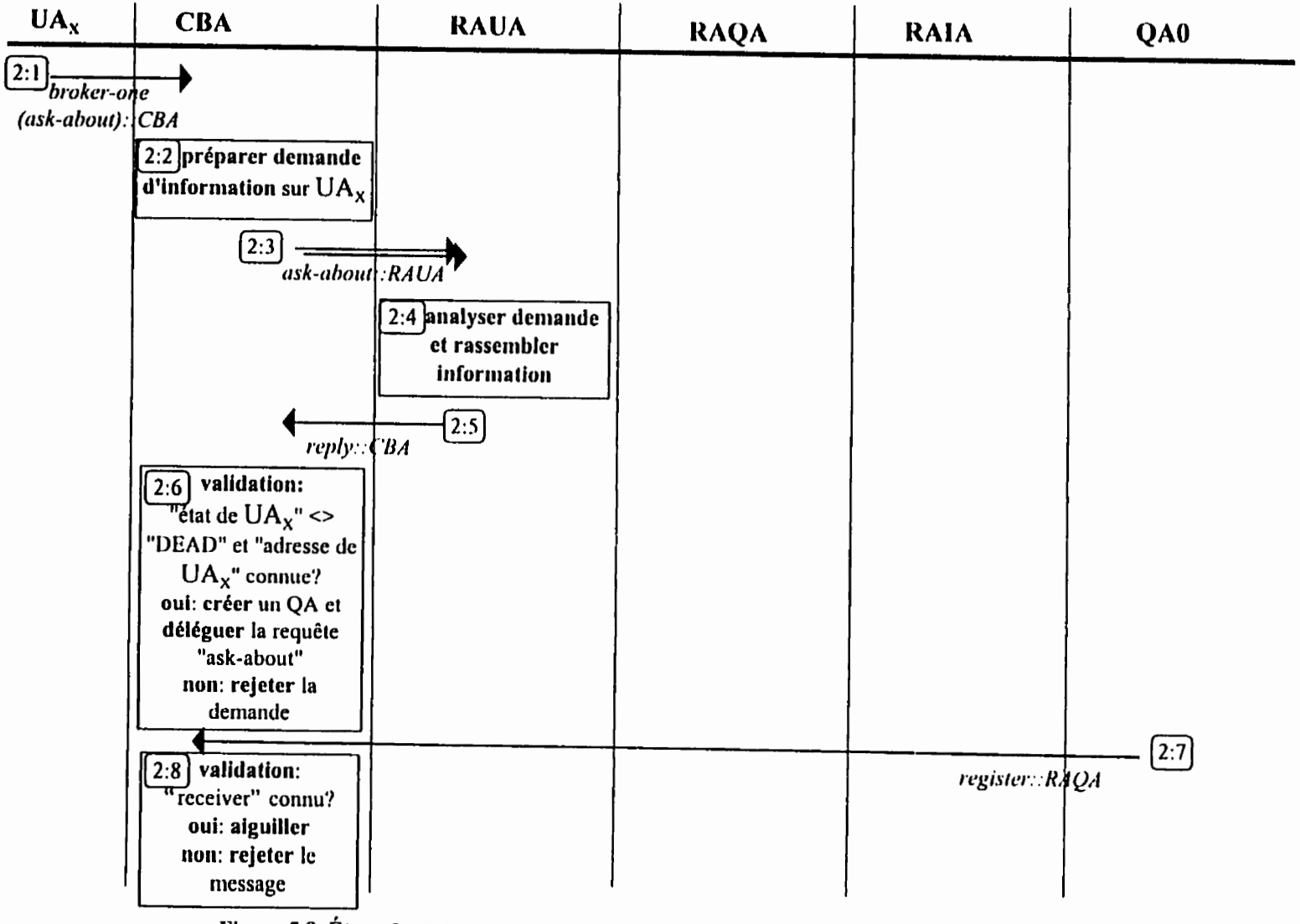


Figure 5.8. Étape 2 : Prise en charge de la requête utilisateur par l'environnement ISAME.

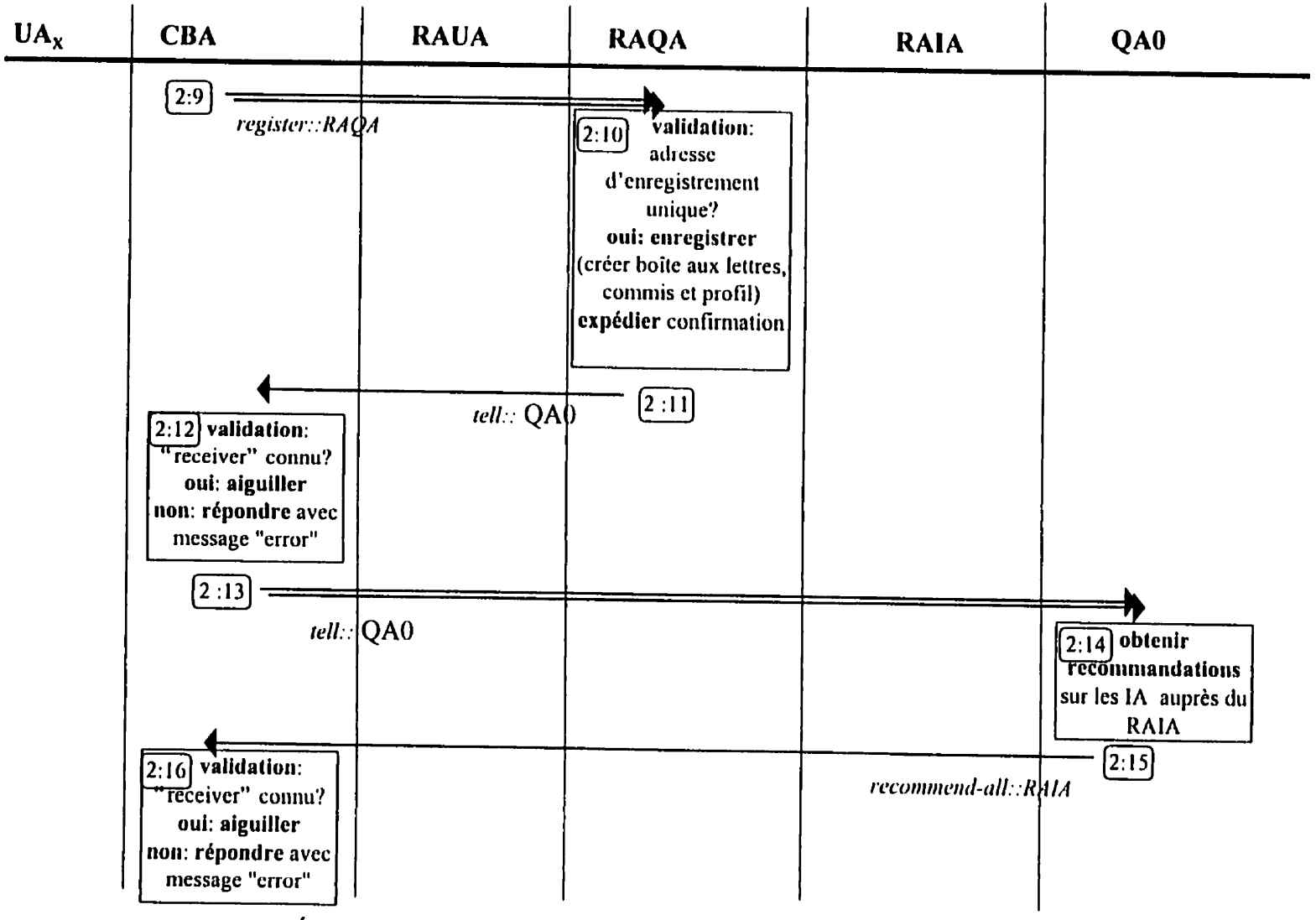


Figure 5.9. Étape 2 (suite) : Prise en charge de la requête utilisateur par l'environnement ISAME.

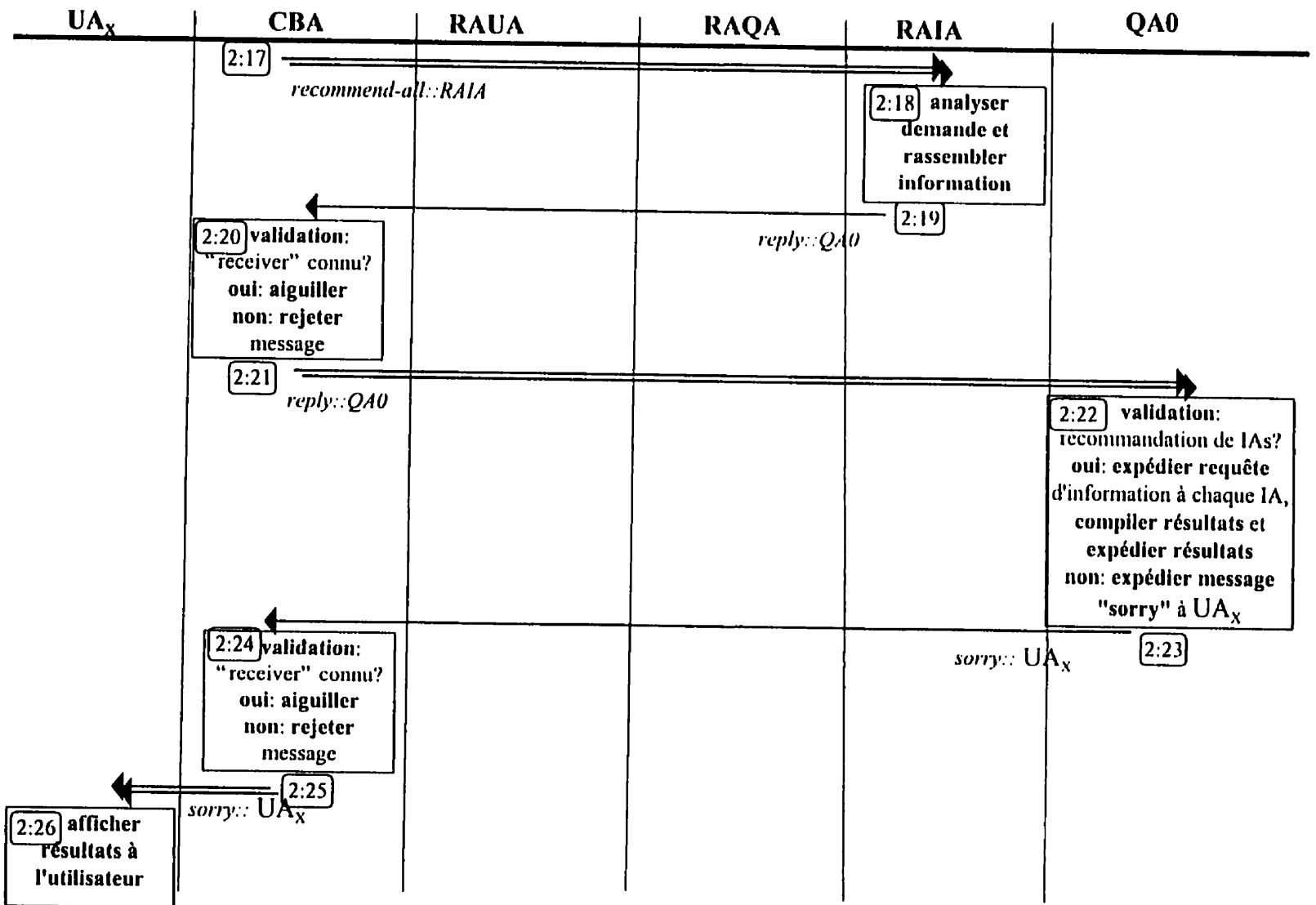


Figure 5.10. Étape 2 (suite) : Prise en charge de la requête utilisateur par l'environnement ISAME.

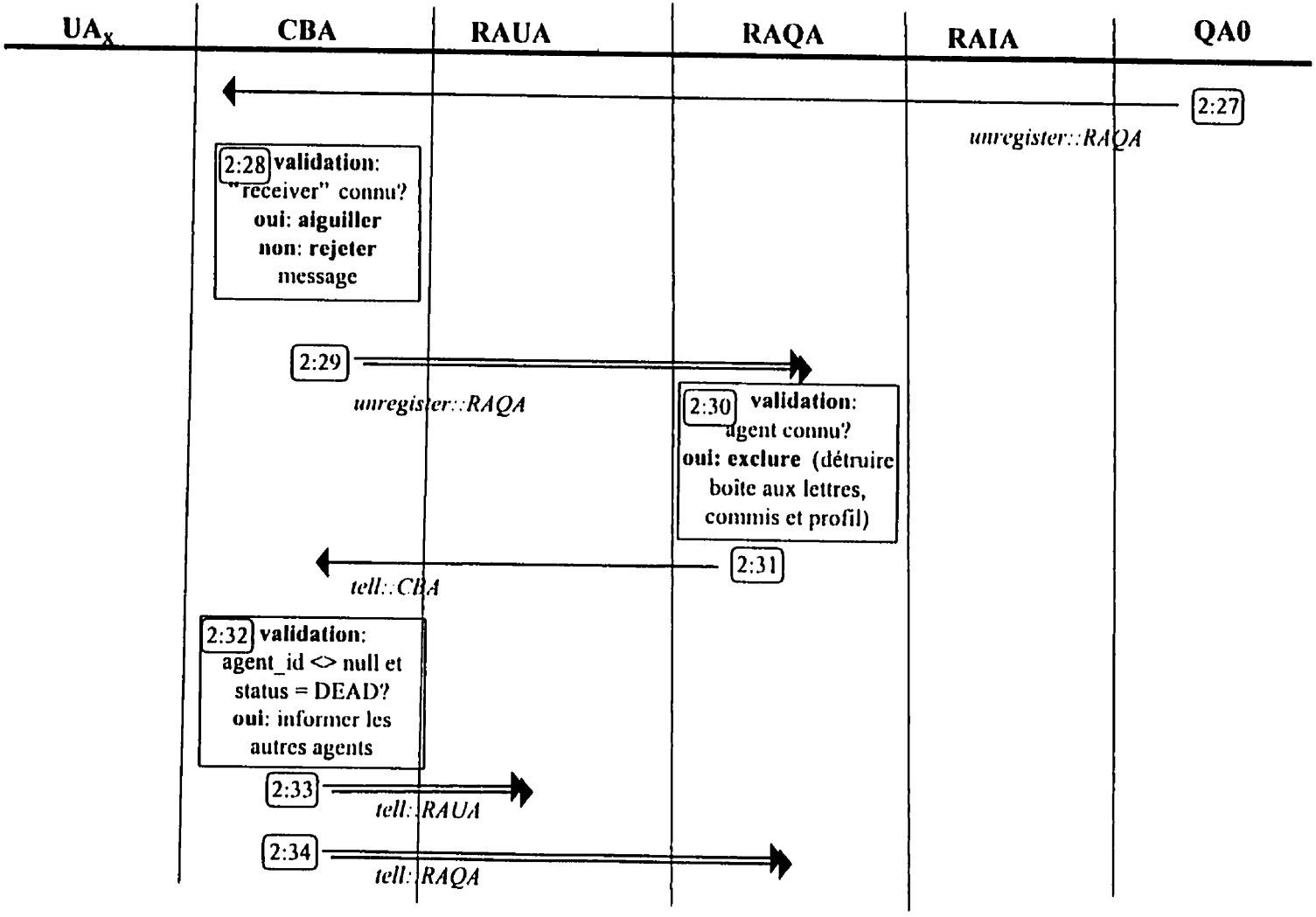


Figure 5.11. Étape 2 (suite) : Prise en charge de la requête utilisateur par l'environnement ISAME.

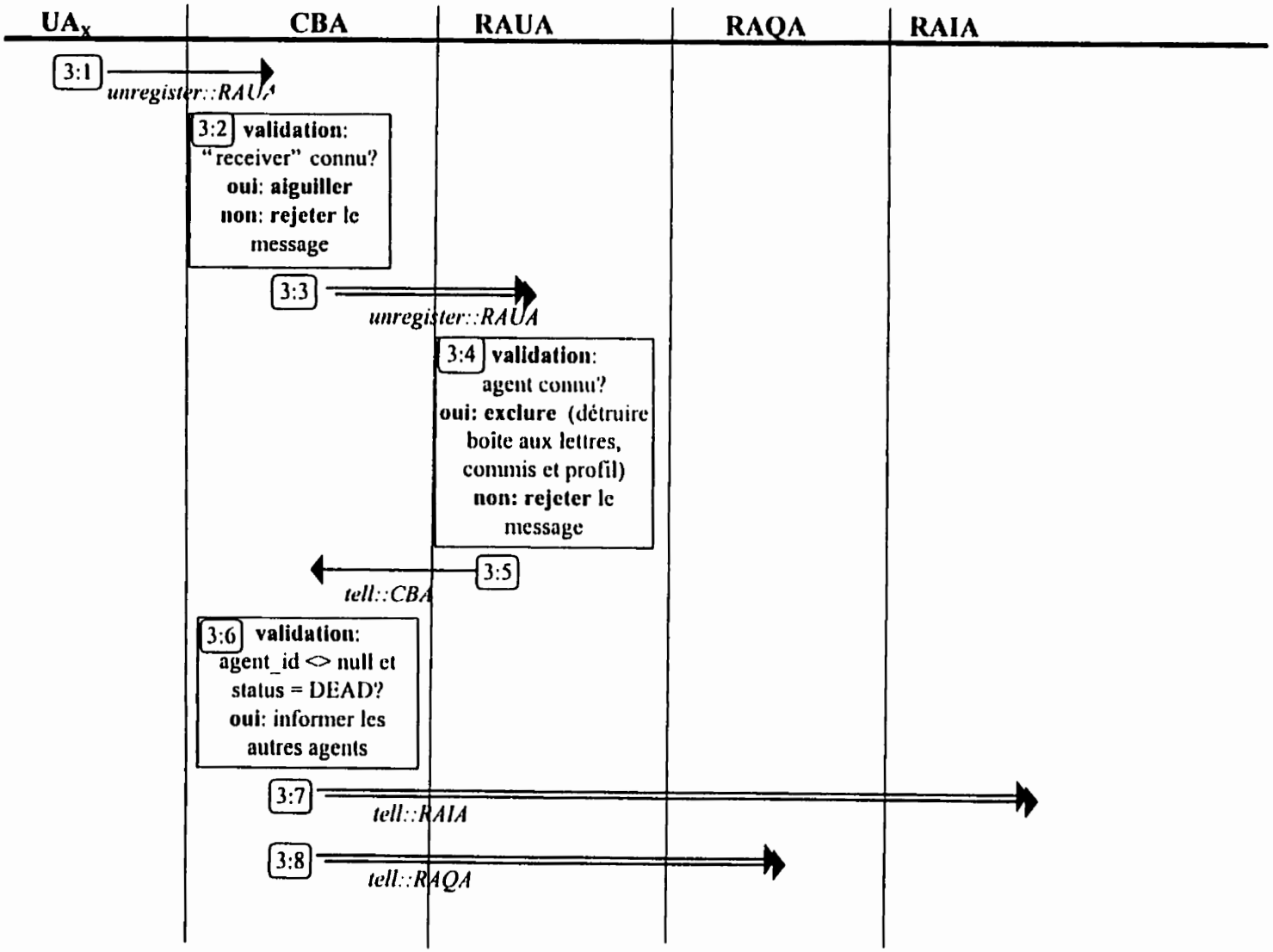


Figure 5.12. Étape 3 : Désistement de l'agent utilisateur UA_x auprès de l'environnement ISAME.

Cas 2 : Le destinataire possède une boîte aux lettres. Le CBA transfert alors le message au *SocketMessageClerk* associé à la boîte aux lettres sur le serveur de l'environnement. Le *SocketMessageClerk* vérifie alors si l'état courant de l'agent lui permet de recevoir des messages, puis dépose le message dans la boîte aux lettres. Aussitôt qu'il le peut, le *SocketMessageClerk* expédie le message, ainsi que tous ceux en attente, au *SocketMessageReceiver* du destinataire situé sur la machine hôte de l'agent, soit le serveur de l'environnement dans le cas des agents de soutien et des agents requêtes, soit sur une machine client dans le cas des agents utilisateurs et information. Sur réception du message, le destinataire effectuera ses propres validations syntaxiques et sémantiques, déterminera s'il est en mesure de traiter ce type de message, déterminera s'il s'agit d'une réponse à un message expédié plus tôt (auquel cas le message est donné au *MessageInterpreter* en attente d'une réponse) ou s'il s'agit d'une requête de traitement (auquel cas l'agent démarrera un *MessageInterpreter*). Ce *MessageInterpreter* demeure actif pour la durée de traitement du message.

Lorsque ce message s'adresse au CBA, et ceci s'applique pour tous les messages dont le CBA est explicitement mentionné comme destinataire ou que le message ne comporte pas de paramètre *receiver*, le traitement est similaire au traitement effectué par tous les agents sur réception d'un message leur étant adressé. Le CBA vérifie donc si ce type de message est supporté, vérifie s'il s'agit d'une réponse ou d'une requête de traitement, et dirige le message vers le *MessageInterpreter* en attente ou instancie un

nouveau *MessageInterpreter* pour le traitement du message (ici le CBA créerait un *CBAMessageInterpreter*).

Dans chacun des messages présentés dans le scénario étudié ici, l'utilisation de l'italique dénote la chaîne de caractères qui, une fois vérifiée par un analyseur *ISAMEMessage*, devient un message de type ISAME-L (dénoté ISAME dans l'implantation présentée ici). Lors de son arrivée chez un agent, chacun de ces messages subit une validation syntaxique et sémantique par un analyseur *KQMLMessage*. De plus, toute chaîne de caractères devenant un message ISAME-L est vérifiée par un analyseur *ISAMEMessage* en conjonction avec un analyseur de type *AgentOntology* ou *VirtualLibraryOntology* selon le cas. Ainsi, chacun des messages subit de 1 à 3 niveaux de validation.

Exemple :

Message : (ask-about :sender CBA :receiver RAUA :language ISAME :ontology AGENT :reply-with 1 :content *(query :content (:agent_id 206.167.88.146:5001 :information [status address agentID]))*)

1. Une première validation syntaxique s'effectue par un analyseur *KQMLMessage* pour l'ensemble de ce message, de même que pour la sémantique des mots clés associés au message *ask-about* (ces mots clés sont tous ceux précédés de ":" mais n'apparaissant pas en italique).
2. Une deuxième validation syntaxique est complétée par un analyseur *ISAMEMessage* sur la partie présentée en italique, de même qu'une validation

sémantique pour les mots clés associés au message *query* (ici, le seul mot clé utilisé est *content*).

3. Une validation sémantique est complétée sur le contenu associé au paramètre *content* du message *query* par l'analyseur sémantique de l'ontologie identifiée par le paramètre *ontology* du message KQML, *AgentOntology* dans l'exemple précédent. Cet analyseur s'assure que les mots clés utilisés, c'est-à-dire *agent_id* et *information*, font partie de l'ontologie, et que toutes les valeurs inscrites dans le vecteur associé au mot *information* font aussi partie de l'ontologie.

Il est à noter que l'ordre des paramètres associés aux messages KQML présentés dans les Figures 5.13 à 5.18 peut différer de celui des messages tels que présentés ci-dessous. L'ordre des paramètres dans les messages qui suivent est basé sur celui habituellement donné dans la documentation KQML. Celui des Figures suit la logique d'extraction séquentielle des clés dans les tables de hachage (classe *Hashtable*) de Java, lesquelles sont utilisées pour conserver les paires paramètres-valeurs des messages KQML manipulés par les agents.

Les paragraphes qui suivent donnent les détails d'explication et de contenu de chacun des messages du scénario étudié.

Étape 1 : Enregistrement du UA_x auprès de ISAME.

- **Activités 1 et 3 :**

But :

Envoi d'un message *register* de UA_X au CBA pour aiguillage vers le RAUA. Ce message est expédié par le *SocketMessageSender* de UA_X vers le *socket* réservé pour le CBA, soit 206.167.88.156:5001. Par ce message, le UA_X désire s'enregistrer auprès du RAUA sous l'adresse "206.167.88.146:5001", 206.167.88.146 étant l'adresse de la machine hôte de UA_X (le client) et 5001 le *socket* auquel il peut être rejoint.

Message :

(register :sender 206.167.88.146:5001 :receiver RAUA :name 206.167.88.146:5001)

- **Activités 5 et 7 :**

But :

Envoi d'un message *tell* du RAUA au UA_X. Après avoir vérifié que l'adresse utilisée par le UA_X est unique pour l'ensemble des UA_X, le RAUA crée un profil local sur le UA_X dans lequel il conserve toutes les informations connues sur le UA_X (nom, adresse, énoncés performatifs de base pour les UA, état courant, états possibles) et crée une boîte aux lettres et un commis pour les messages reçus par le CBA pour le UA_X. Le RAUA informe ensuite l'agent UA_X qu'il est maintenant enregistré dans l'environnement ISAME en lui indiquent que son état dans l'environnement est *AWAKE*.

Message :

(tell :sender RAUA :receiver 206.167.88.146:5001 :language ISAME :ontology AGENT :content (*update :content (:agent_ID 206.167.88.146:5001 :status AWAKE)*))

Étape 2 : Prise en charge de la requête utilisateur par l'environnement ISAME.

- **Activité 1 :**

But :

Envoi d'un message *broker-one* de UA_X au CBA. Par ce message, le UA_X demande au CBA de trouver un agent de QA pouvant traiter la requête d'information décrite dans le message *ask-about* associé au paramètre *content*.

Message :

(broker-one :sender 206.167.88.146:5001 :receiver CBA :language KQML :ontology Agent :reply-with 1 :content (ask-about :sender 206.167.88.146:5001 :receiver CBA :reply-with 1 :language ISAME :ontology VirtualLibrary :content *(query :content (:format* [HTML] *:content [multi-agent architecture and KQML] :information [source]))*)

- **Activité 3 :**

But :

Envoi d'un message *ask-about* du CBA au RAUA. Par ce message, le CBA demande à l'agent RAUA de lui retourner les informations *status*, *address* et *agent_ID* associées à l'agent UA_x connu sous le nom "206.167.88.146:5001". Ceci permettra à l'agent CBA de déterminer si l'agent UA_X est actif dans l'environnement ISAME avant de démarrer le processus de recherche d'information.

Message :

(ask-about :sender CBA :receiver RAUA :language ISAME :ontology AGENT :reply-with 1 :content *(query :content (:agent_id 206.167.88.146:5001 :information [status address]))*)

- **Activité 5 :**

But :

Envoi d'un message *reply* du RAUA au CBA. Ce message comprend les résultats trouvés par le RAUA dans le profil qu'il conserve sur le UA_X pour la requête *ask-about* précédemment expédiée par le CBA. Lorsqu'il reçoit ce message, le CBA le redirige vers le *CBAMessageInterpreter* dont le numéro d'identification est 1 puisqu'il s'agit de l'interpréteur en attente du résultat. L'état du UA_X permettant le traitement de la requête d'information, le *CBAMessageInterpreter* en charge du message demande à l'environnement ISAME de créer un QA pouvant se charger de la requête. L'agent QA0 est alors créé. Le nom de l'agent requête est généré automatiquement par l'environnement ISAME qui s'occupe aussi de lui attribuer un *socket* à partir de la région d'adresses réservées pour les agents de QA La requête est alors déléguée au QA0 nouvellement créé.

Message :

(reply :sender RAUA :receiver CBA :language ISAME :ontology AGENT :in-reply-to 1 :content *(update :content ([:agentID 206.167.88.146:5001 :status AWAKE :address 206.167.88.146:5001]))*)

- **Activité 7 et 9 :**

But :

Envoi d'un message *register* du QA0 au CBA pour aiguillage vers le RAQA. À sa création, le QA0 s'enregistre automatiquement auprès de l'environnement ISAME.

Message :

(register :sender 206.167.88.156:5005 :receiver RAQA :name QA0)

- **Activités 11 et 13 :**

But :

Envoi d'un message *tell* du RAQA au QA0. Après avoir vérifié que l'adresse utilisée par le QA0 est unique pour l'ensemble des QA, le RAQA crée un profil local sur le QA0 dans lequel il conserve toutes les informations connues sur le QA0 (nom, adresse, énoncés performatifs de base pour les QA, état courant, états possibles) et crée une boîte aux lettres et un commis pour les messages reçus par le CBA pour le QA0. Le RAQA informe ensuite l'agent QA0 qu'il est maintenant enregistré dans l'environnement ISAME en lui indiquent que son état dans l'environnement est *AWAKE*.

Message :

(tell :sender RAQA :receiver 206.167.88.156:5005 :language ISAME :ontology AGENT :content (*update :content (:agent_ID 206.167.88.156:5005 :status AWAKE)*))

- **Activités 15 et 17 :**

But :

Envoi d'un message *recommend-all* du QA0 au RAIA. Par ce message, le QA0 débute le traitement de la requête d'information du UA_x en demandant au RAIA de lui suggérer des sources d'information dont le profil correspond aux informations indiquées dans le paramètre *content* du message ISAME-L. Dans ce cas, le QA0 demande des IA_y dont l'état courant est éveillé (AWAKE) et qui peuvent fournir des résultats en format *html*. Pour chacun des agents suggérés, le QA0 désire recevoir les informations suivantes : *agent_id*, *address*, *format*, *percentage_rating* et *relative_ranking*.

Message :

(recommend-all :sender 206.167.88.156:5005 :receiver RAIA reply-with 1 :language ISAME :ontology AGENT :content *(query :content (:format html :status AWAKE :information [agent_id address format percentage_rating relative_ranking]))*)

- **Activités 19 et 21 :**

But :

Envoi d'un message *reply* du RAIA à QA0. Ce message de réponse fournit au QA0 des suggestions de IA_y pour le profil donné. Dans ce cas, aucun IA_y n'étant actif dans l'environnement, le paramètre *content* du message ISAME-L a une valeur associée vide ("( )"). À la réception de ce message, le RAIA identifie les IA_y dont la description correspond à la requête du QA0 et comparant les éléments de la requête avec les informations contenus dans le profil de chacun des IA_y.

Message :

(reply :sender RAIA :receiver 206.167.88.156:5005 :language ISAME :ontology Agent :in-reply-to 1 :content *(update :content())*)

Si le RAIA avait été en mesure de suggérer deux IA_y, soit IA1 et IA2, le format du message aurait été le suivant (les informations indiquées ici sont fictives et ne sont données qu'à titre d'exemple) :

(reply :sender RAIA :receiver 206.167.88.156:5005 :language ISAME :ontology Agent :in-reply-to 1 :content *(update :content([:agent_id IA1 :address 206:167:88:160:5037 :format [html url ascii] :percentage_rating 98 :relative_ranking 1] [:agent_id IA2 :address 206:167:88:143:5007 :format [html] :percentage_rating 95 :relative_ranking 3]))*)

- **Activités 23 et 25 :**

But :

Envoi d'un message *sorry* de QA0 à UA_x. À la réception du message *reply* du RAIA, le QA0 constate qu'aucun IA_y n'est présentement en mesure de fournir des résultats à la requête du UA_x. Le QA0 informe donc le UA_x qu'aucun résultat ne pourra être fourni.

Message :

(sorry :sender QA0 :receiver 206.167.88.146:5015 :in-reply-to 1 :comment "No suitable information agent available at the present")

Dans le cas où des IA_y auraient été disponibles, le QA0 aurait expédié un message *ask-about* à chacun des IA_y identifiés, et aurait associé au paramètre *content* la requête d'information reçue du UA_x. Il aurait ensuite attendu les résultats de chacun des IA_y, compilé les résultats en un seul message et expédié un message *reply* au UA_x avec comme contenu l'information compilée.

- **Activités 27 et 29 :**

But :

Envoi d'un message *unregister* de QA0 au CBA pour aiguillage vers le RAQA. Par ce message, le QA0, qui a maintenant complété la tâche pour laquelle il avait été créé, désire se désister auprès de ISAME. Sur réception de ce message, le RAQA détruira le profil qu'il maintient sur le QA0, détruira sa boîte aux lettres et son commis et informera le reste de l'environnement, par l'entremise du CBA, du désistement du QA0.

Message :

(unregister :sender 206.167.88.156:5005 :receiver RAQA :name QA0)

- **Activité 31 :**

But :

Envoi d'un message *tell* du RAQA au CBA. Par ce message, le RAQA informe le CBA que l'agent enregistré sous l'adresse 206.167.88.156:5005 vient d'être exclu de l'environnement. Sur réception de ce message, le CBA en analysera le contenu et, puisqu'il s'agit d'un message l'informant de l'exclusion d'un agent, en informera les autres agents registraires.

Message :

(tell :sender RAQA :receiver CBA :language ISAME :ontology AGENT :content *(update :content (:agent_id 206.167.88.156:5005 :status DEAD))*)

Dans le futur, le CBA devrait faire parvenir ce message à tous les agents ayant une boîte aux lettres dans l'environnement afin que tout message provenant de cet agent ainsi que tout message présentement traité soient détruits.

- **Activités 33 et 34 :**

But :

Envoi d'un message *tell* du CBA aux RAUA et RAIA. Par ce message, le CBA informe les deux agents de soutien qu'un agent vient d'être exclu de l'environnement. À la réception de ce message, les registraires cessent de traiter toute requête provenant de l'agent exclu.

Messages :

(tell :sender CBA :receiver RAUA :language ISAME :ontology AGENT :*content (update :content (:agent_id 206.167.88.156:5005 :status DEAD))*)

(tell :sender CBA :receiver RAQA :language ISAME :ontology AGENT :content *(update :content (:agent_id 206.167.88.156:5005 :status DEAD))*)

Étape 3 : Désistement du UA_X auprès de ISAME.

- **Activités 1 et 3 :**

But :

Envoi d'un message *unregister* de UA_X au CBA pour aiguillage vers le RAUA. Par ce message, le UA_X désire se désister auprès de ISAME. Sur réception de ce message, le RAUA détruira le profil qu'il maintient sur le UA_X, sa boîte aux lettres et son commis et informera le reste de l'environnement, par l'entremise du CBA, du désistement du UA_X.

Message :

(unregister :sender 206.167.88.146:5001 :receiver RAUA :name 206.167.88.146:5001)

- **Activité 5 :**

But :

Envoi d'un message *tell* de RAUA au CBA. Par ce message, le RAUA informe le CBA que l'agent enregistré sous l'adresse 206.167.88.146:5001 vient d'être exclu de l'environnement. Sur réception ce message, le CBA en analysera le contenu et, puisqu'il s'agit d'un message informant de l'exclusion d'un agent, en informera les autres agents registraires.

Message :

(tell :sender RAUA :receiver CBA :language ISAME :ontology AGENT :content (update :content (:agent_id 206.167.88.146:5001 :status DEAD)))

Dans le futur, tout comme dans le cas précédent, le CBA devrait faire parvenir ce message à tous les agents ayant une boîte aux lettres dans l'environnement afin que tout

message provenant de cet agent ainsi que tout message présentement traité soient détruits.

- **Activités 7 et 8 :**

But :

Envoi d'un message *tell* du CBA aux RAQA et RAIA. Par ce message, le CBA informe les 2 agents de soutien qu'un agent vient d'être exclu de l'environnement. À la réception de ce message, les registraires cessent de traiter toute requête provenant de l'agent exclu.

Messages :

(tell :sender CBA :receiver RAIA :language ISAME :ontology AGENT :content *(update :content (:agent_id 206.167.88.146:5001 :status DEAD))*)

(tell :sender CBA :receiver RAQA :language ISAME :ontology AGENT :content *(update :content (:agent_id 206.167.88.146:5001 :status DEAD))*)

5.2.3 Résultats obtenus par la mise en oeuvre de l'architecture

Les trois étapes du scénario ont été exécutées dans l'environnement ISAME présentement disponible. Les Figures 5.13 à 5.19 démontrent que tous les messages identifiés précédemment sont échangés entre les agents impliqués.

5.3 Évaluation du modèle

L'évaluation de l'architecture proposée pour ISAME reflète les objectifs fixés en introduction ainsi que les hypothèses adoptées à la section 5.1. Elle est basée sur les

résultats obtenus à partir de l'implantation, sur le potentiel escompté des éléments non implantés à ce jour, ainsi que sur la comparaison avec le projet UMDL, sans doute à ce jour le plus ambitieux projet en recherche intelligente d'information par système multi-agents. Une brève description du projet UMDL est fournie à l'Annexe E. La documentation disponible sur ce projet peut être trouvée à l'adresse URL suivante :

(http : //telemachus.engin.umich.edu:80/documents/nov_demo/main/main.html).

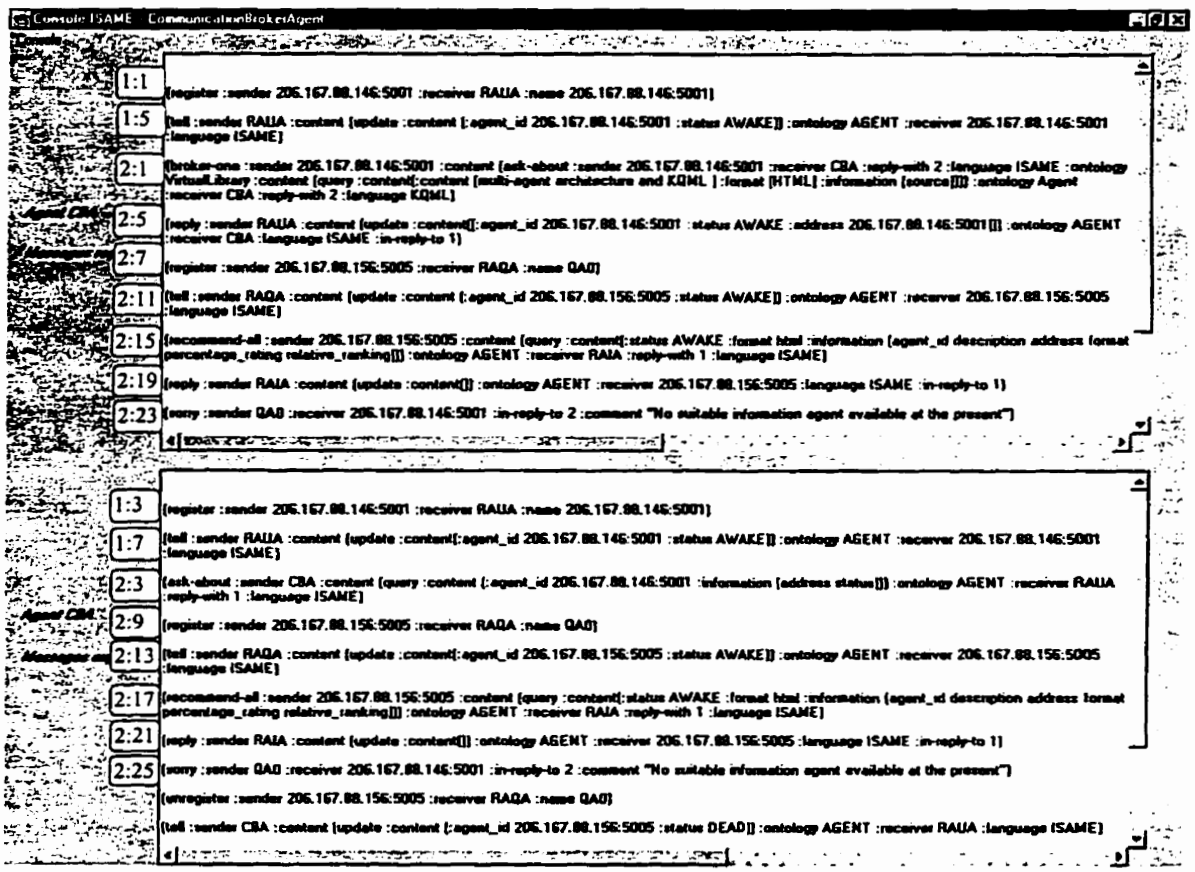


Figure 5.13. Messages échangés par le CommunicationBrokerAgent.

ISAME comme point d'accès unique

L'environnement ISAME offre bel et bien un point d'accès unique à un ensemble de services et contenus de bibliothèque virtuelle. Cependant, bien que des services de

filtrage de sources d'information et de résultats de recherche fassent partie des services de soutien de l'environnement ISAME, la qualité et la quantité des contenus d'information demeureront toujours fortement dépendantes de la qualité et de la quantité des contenus fournis par les agents information inscrits auprès d'ISAME. Ceci est dû au fait que les sources d'information sont externes à l'environnement ISAME et donc non contrôlées par l'environnement lui-même. Ce problème n'est pas unique à ISAME, mais est plutôt inhérent aux architectures multi-agents dont les composantes sont dynamiques.

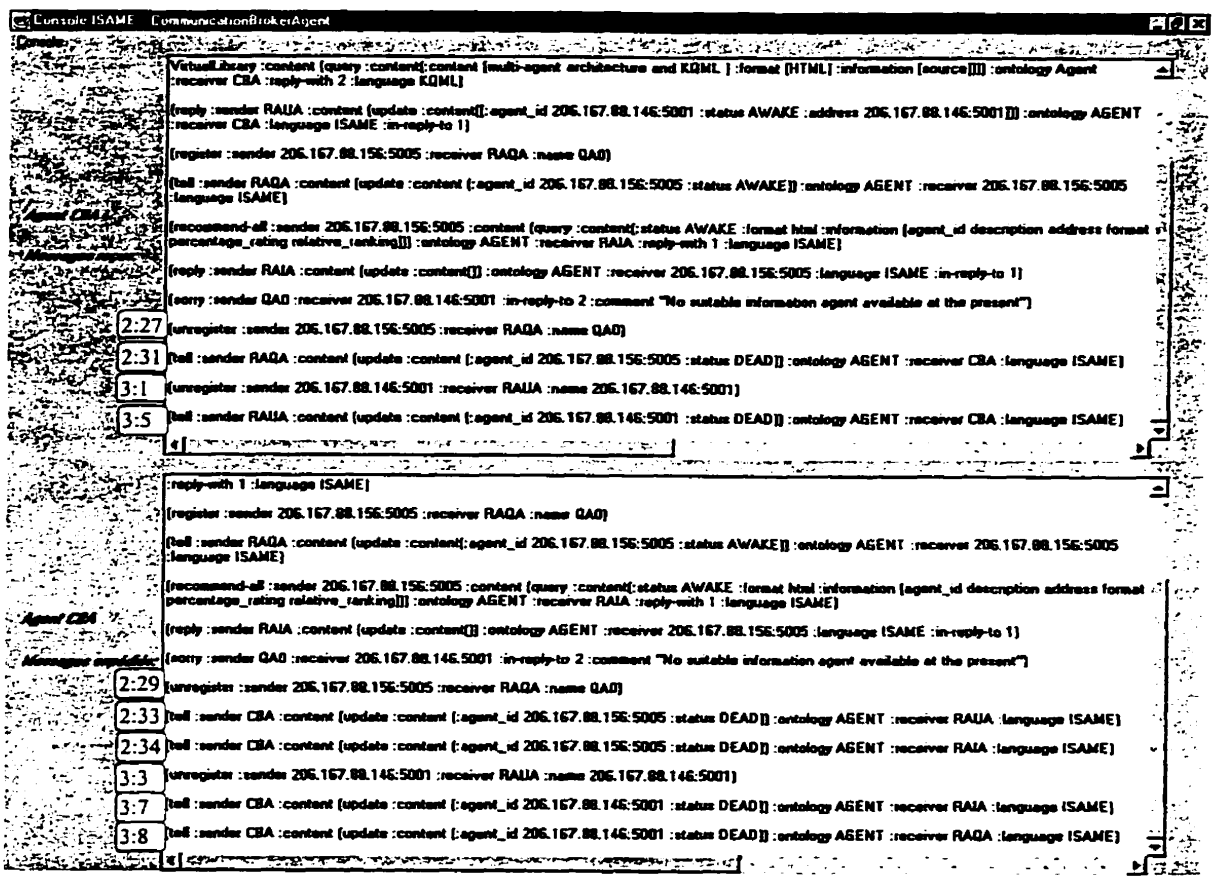


Figure 5.14. Messages échangés par le CommunicationBrokerAgent (suite).

Néanmoins, bien qu'aucun agent information n'ait été implanté jusqu'à ce jour dans ISAME, il est espéré que ce dynamisme permettra à ISAME de tirer profit des avantages

reliés aux systèmes multi-agents et mentionnés à la section 2.3.4.1 : recherche distribuée de solution, création dynamique des équipes de travail, adaptation des équipes en cours de processus, faible coût de réutilisation des agents, et augmentation de la performance avec le temps. Tout est mis en place dans ISAME afin de minimiser l'impact des problèmes inhérents à la recherche distribuée d'information en fournissant aux agents une gamme d'outils : langages et outils permettant de rendre homogène la communication, interpréteurs de langages et d'ontologies pour éviter l'ambiguïté, et intégration simplifiée des nouveaux agents à l'environnement. La validité de tels outils devra cependant être mise à l'épreuve par l'implantation d'agents utilisateur et information.

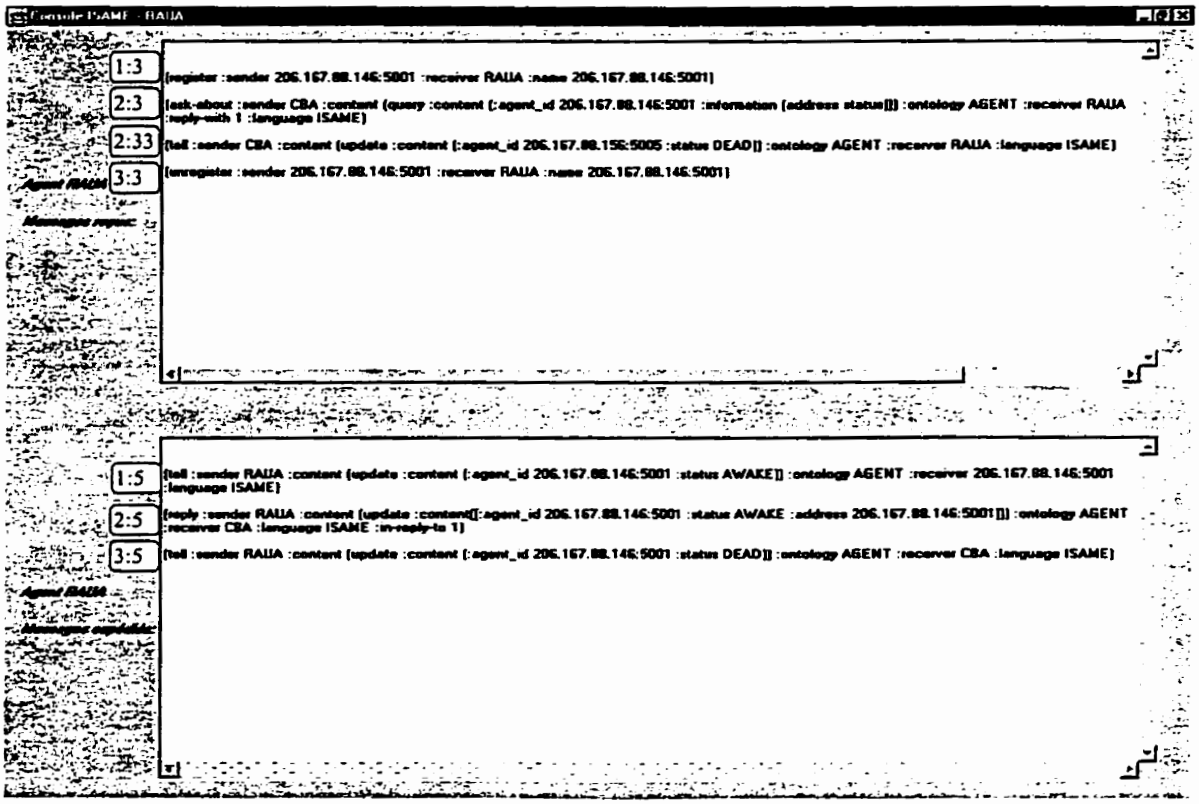


Figure 5.15. Messages échangés par le RAUA.

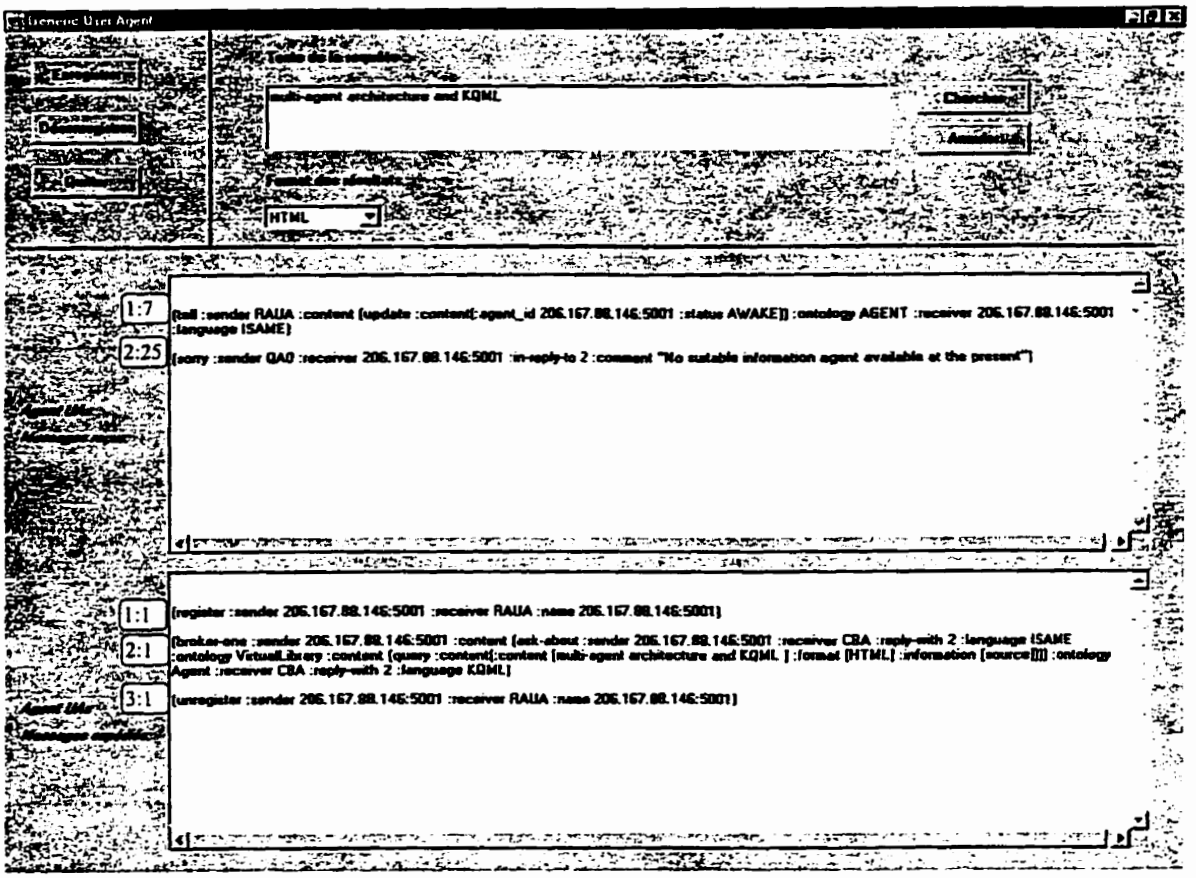


Figure 5.16. Messages échangés par le UAx.

Ces caractéristiques ne sont pas uniques à ISAME, mais plutôt à l'ensemble des architectures multi-agents offrant une architecture ouverte pour la recherche intelligente d'information ou pour d'autres buts. C'est le cas notamment des projets UMDL et JavaAgent qui sont basés sur des systèmes centralisés d'enregistrement d'agents et de facilitation pour l'échange de services entre agents. JavaAgent utilise comme point d'accès central à l'environnement multi-agents l'adresse d'un agent de soutien nommé le ANS (Agent Name Server) qui s'occupe d'enregistrer les agents d'une façon comparable à ce qui est accompli par les agents registraires de ISAME. UMDL utilise aussi un agent

central, soit le Registry Agent, pour donner l'accès aux services et contenus offerts par l'environnement, bien que cet agent soit perçu comme centralisé logiquement.

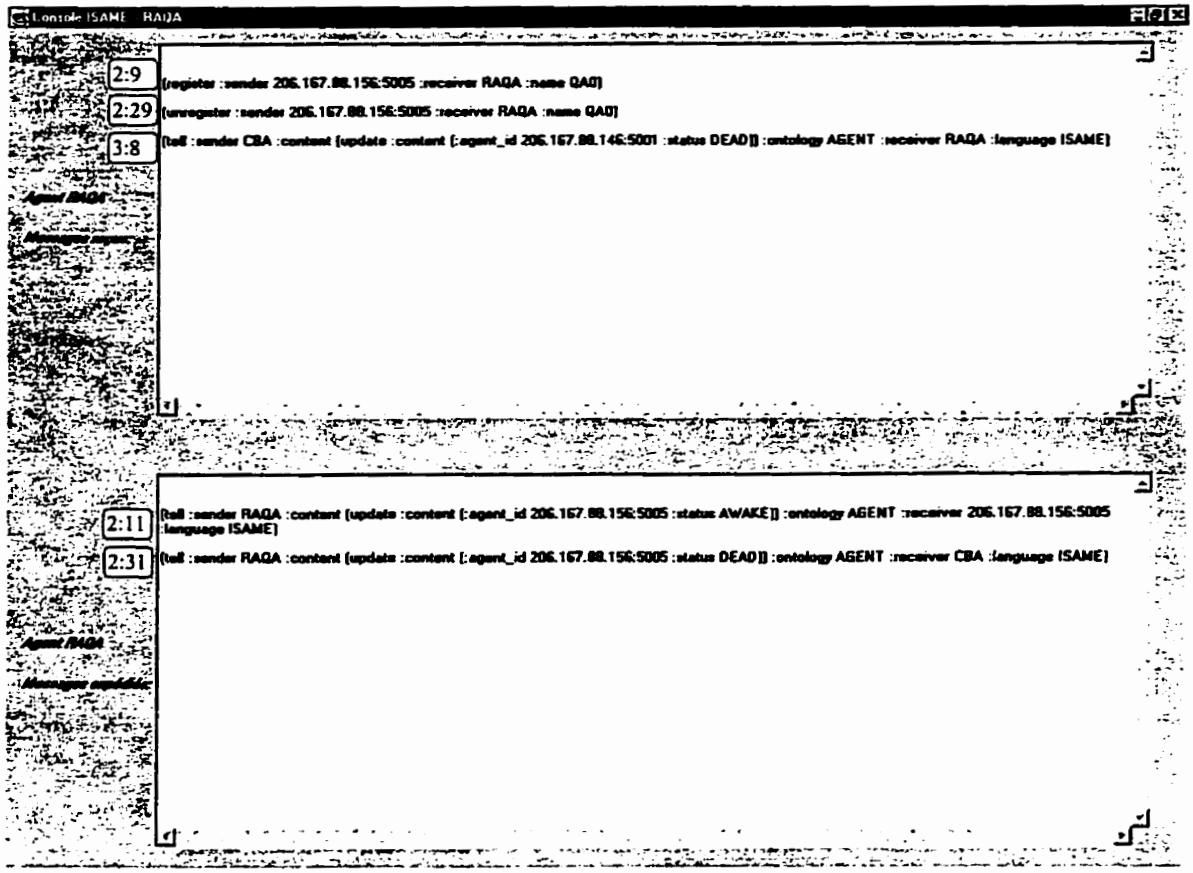


Figure 5.17. Message échangés par le RAQA.

À la manière de ISAME, UMDL utilise plusieurs agents registraires qui conservent des informations sur les différentes catégories d'agents : utilisateurs, information et les agents médiateurs (les agents responsables de planifier les requêtes, appelés *Query Planners*, et les registraires eux-mêmes). Si dans ISAME les agents UA_x et IA_y doivent envoyer leur message d'enregistrement explicitement auprès du bon registraire (soit RAUA ou RAIA), l'environnement UMDL ne semble pas faire cette distinction explicite. La documentation actuellement disponible sur UMDL ne permet

cependant pas de déterminer comment cette distinction est effectuée afin que les agents soient enregistrés auprès du bon registraire.

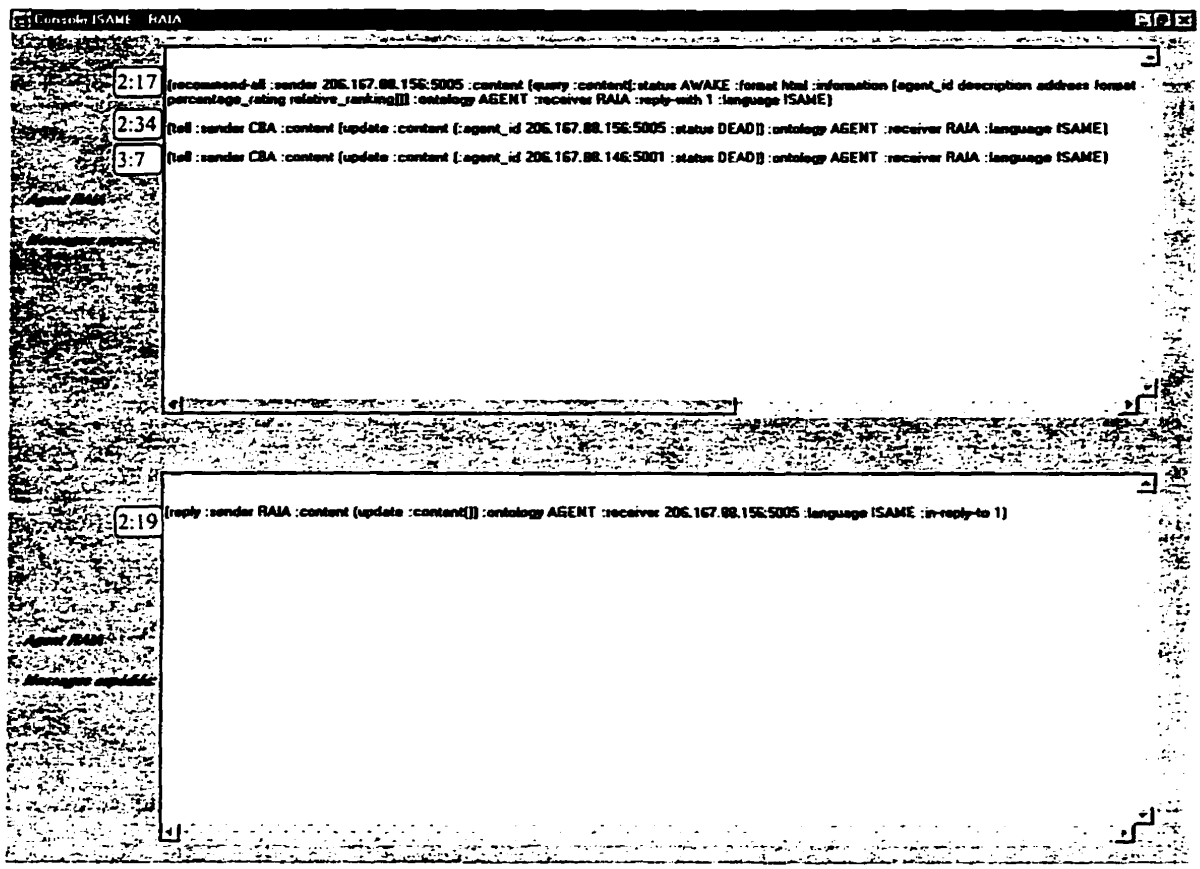


Figure 5.18. Messages échangés par le RAIA.

Indépendance des agents utilisateurs et information

Les résultats obtenus par l'implantation partielle de ISAME soutiennent l'argument d'indépendance des agents utilisateurs face aux agents information dans ISAME. En effet, grâce au travail des agents de soutien, des agents requêtes, du système de communication interne et des ontologies communes, les agents de UA et les agents de IA demeureront totalement indépendants les uns des autres, tant dans leur mode d'implantation que dans leur disponibilité. Ainsi, une requête postée par un UA_x auprès

de ISAME est totalement déléguée à un agent de QA jusqu'au retour des résultats. Lorsqu'ils seront implantés, cette requête bénéficiera de l'ensemble des services offerts par l'environnement (profils sur les agents de UA et les agents de IA, manipulation des requêtes persistantes, choix optimisé des sources d'information, filtrage et livraison des résultats). Cette caractéristique est aussi l'une des grandes forces de l'architecture de UMDL. Tout comme dans ISAME, cet isolement est effectué par l'entremise d'une série d'agents de soutien prenant en charge les besoins des agents utilisateurs.

Bien que cet isolement total des UA_x face au travail de sélection des sources d'information, de collecte et de filtrage de résultats et de connaissance des agents information simplifie grandement le travail des concepteurs d'agents utilisateurs, ce même isolement pourrait devenir une contrainte inacceptable pour les concepteurs d'agents utilisateurs désirant exercer plus de contrôle sur ces étapes de la requête à partir de l'agent utilisateur. Toutefois, bien que cet élément n'ait pas été développé ici, on peut remarquer que l'implantation actuelle de l'architecture ne peut empêcher un agent utilisateur d'entrer en communication avec des sources d'information par l'entremise des agents de soutien. Pour ce faire, l'agent utilisateur peut demander au RAIA de lui suggérer des sources d'information, puis de communiquer sa requête directement aux sources d'information. La seule contrainte imposée au UA_x serait alors de s'enregistrer auprès de l'environnement de façon à obtenir les services d'une boîte aux lettres dont le CBA peut se servir pour réacheminer les résultats fournis par les IA_y directement sollicités.

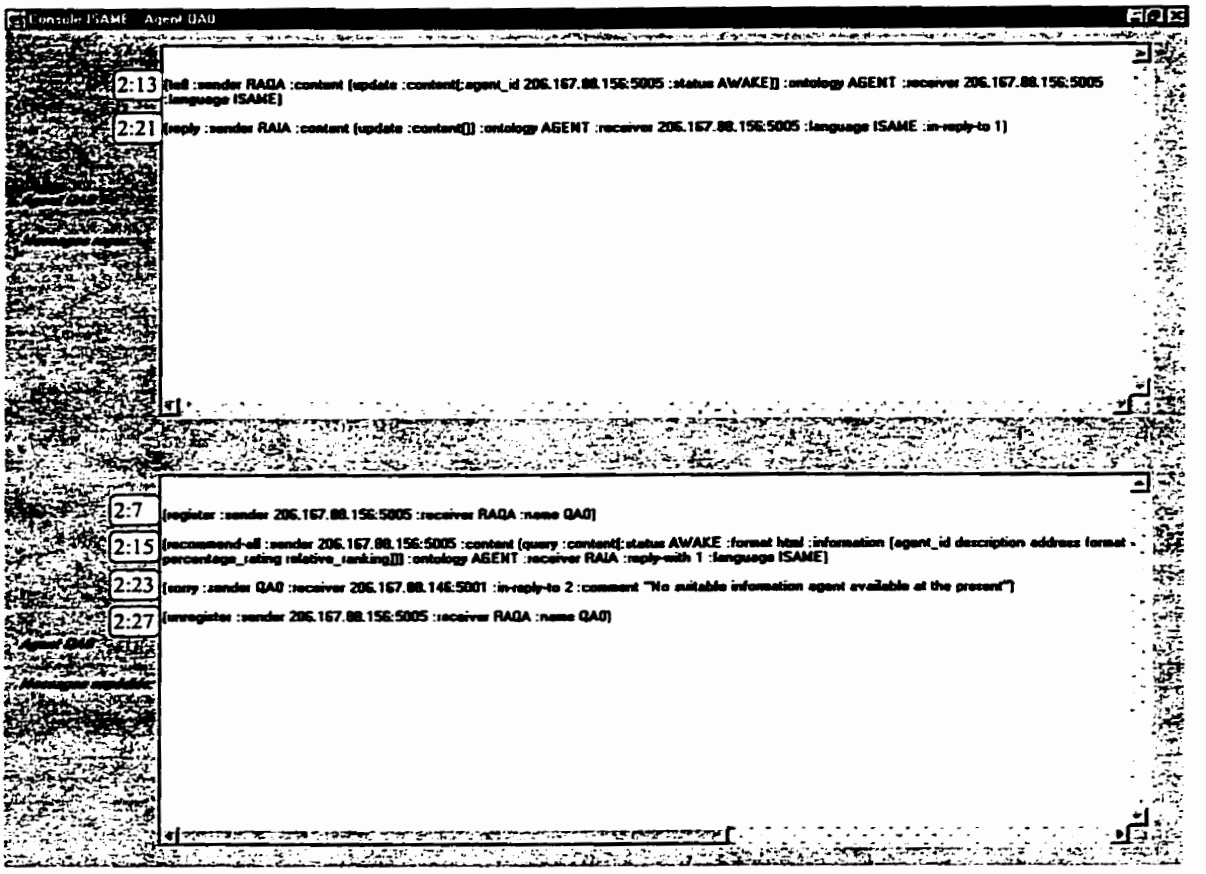


Figure 5.19. Messages échangés par le QA0.

Indépendance des agents externes face aux agents internes

Lors du traitement d'une requête utilisateur, les agents de UA et les agents de IA n'ont pas à se soucier de la complexité et de la diversité des services offerts par ISAME. Cette remarque tient tout autant pour les agents de UMDL. Cette indépendance provient en fait de la standardisation des langages et ontologies de communication. En effet, tant que les agents externes utilisent des énoncés performatifs valides et annoncés par les agents de soutien connus et qu'ils respectent la syntaxe et la sémantique des langages et ontologies connus dans l'environnement, la qualité des interactions entre les agents

externes et internes demeure assurée, assurant du même coup le traitement en bonne et due forme de la requête de l'utilisateur.

Le scénario présenté dans la section 5.2 met en relief le nombre de tâches complétées dans ISAME pour mener à terme la requête d'information de l'utilisateur et la simplicité de l'interaction nécessaire entre ce UA_x et l'environnement. Les seules contraintes imposées au UA_x sont de connaître l'adresse d'entrée en contact avec le CBA ainsi que le nom du RAUA pour fin d'enregistrement dans l'environnement, et de supporter les langages et ontologies pour la communication de requêtes d'information. Bien que complexe, cette dernière contrainte est grandement simplifiée par la disponibilité de classes spécifiques et génériques fournies par l'implantation actuelle et pouvant servir de base à la mise en oeuvre d'agents utilisateurs spécifiques.

Dynamisme des agents externes

Tant dans ISAME que dans UMDL, les agents externes sont libres de s'inscrire et de se désister auprès de l'environnement selon leurs besoins. Dans ISAME, les concepteurs des agents externes doivent cependant tenir compte du fait suivant : lors d'un désistement, les activités de ramasse miettes s'assureront qu'aucune trace de cet agent ne réside dans l'environnement, ce qui signifie que le profil de l'agent, construit peut-être au fil de nombreuses interactions avec l'environnement, devient lui aussi irrécupérable. Si l'agent désire se réinscrire, le profil devra être reconstruit. UMDL évite ce problème en archivant le profil des agents qui quittent ISAME dans une base de données persistante. Ce profil pourrait être récupéré dans le cas où l'agent s'enregistre à nouveau. Ceci est mis

en place par l'agent responsable des activités de ramasse miettes (*Garbage Collection Agent*). Cependant, ce profil ne peut être réutilisé que dans la mesure où l'agent se réinscrit sous le même nom.

Finalement, mentionnons que ce dynamisme des agents utilisateur et information n'est pas toujours supporté dans les projets de recherche intelligente d'information. Un très petit nombre de projets de recherche d'information examinés, dont plusieurs sont décrits en Annexe E, présente une architecture ouverte permettant l'intégration de nouvelles sources d'information ou de services. Plusieurs de ces projets, bien que basés sur une approche multi-agents, consistent en des boîtes fermées offrant une interface de requête fixe à l'utilisateur, c'est-à-dire un agent utilisateur imposé. De plus, seuls les concepteurs du système sont en mesure d'ajouter ou de retirer des sources d'information à l'environnement, contrôlant ainsi totalement les contenus offerts aux utilisateurs. Parmi ces systèmes, mentionnons Autonomy et MetaSearch. La documentation complète peut être trouvée aux adresses URL suivantes :

Autonomy : (http://www.agentware.com/)

MetaSearch : (http://www.netseek.com/metaseek/homepage.html).

Optimisation de la communication par les boîtes aux lettres

Une évaluation comparative des mécanismes de communication ISAME avec les méthodes du tableau noir et de communication par messages a déjà été complétée à la section 3.2.6. Le scénario présenté dans ce chapitre démontre que, grâce au service de boîte aux lettres, aucun des agents externes de ISAME n'est forcé d'être branché en

permanence pour s'assurer un retour d'information de la part des agents sollicités. De même, les agents désirant fournir une réponse ne sont pas dépendants de la disponibilité de leur interlocuteur, cette tâche utilitaire étant prise en charge par le système de commis et boîte aux lettres fourni par ISAME. Cependant, si un agent reste inactif trop longtemps dans ISAME, les messages mis en attente seront éventuellement détruits sans être livrés.

La communication entre les agents de UMDL est basé sur la norme CORBA (Common Object Request Broker Architecture) (Mowbray et Zahavi, 1995) permettant la communication entre objets distribués. Les agents délèguent donc l'échange de messages à un ORB (Object Request Broker), service rendu par le CBA dans ISAME. Si l'échange de message par l'entremise du CBA et des boîtes aux lettres augmente le nombre total de message échangés dans ISAME et nécessite une bonne coordination entre les différents intervenants, la communication par CORBA amène cependant d'autres contraintes : elle impose la mise en place d'un ou de plusieurs ORB (*Object Request Broker*, soit l'objet qui achemine les messages d'un objet vers un autre), et l'enregistrement des agents ou de leurs objets de communication auprès de ce ORB. Ce travail peut être grandement simplifié dans le cas où le serveur de UMDL prend en charge l'ensemble de ces contraintes. La documentation du projet ne fournit cependant aucun détail à ce sujet. De plus, cette documentation ne permet pas de déterminer si les agents font directement appel aux méthodes des autres agents pour le traitement des messages, ou s'ils échangent les chaînes de caractères représentant les messages KQML

en faisant appel à une méthode générique implantée dans les objets. Finalement, rien n'est spécifié quant aux actions prises par le ORB dans le cas où les messages ne peuvent être livrés.

Optimisation de l'environnement par ramasse miettes

L'impact réel des activités de ramasse miettes dans ISAME peut difficilement faire l'objet d'une évaluation substantielle à partir de l'implantation actuelle du modèle. Un premier coup d'oeil aux structures de données et processus mis en place jusqu'à ce jour démontre que les activités de ramasse miettes permettent d'éliminer les résidus reliés à un agent lors de son expulsion de l'environnement. Cependant, deux éléments clés demeurent à vérifier :

- Une grande partie de ces activités étant implantés par le biais de processus légers autonomes, ces processus pourraient éventuellement ralentir le travail des agents résidant sur le serveur ainsi que les autres processus roulant en parallèle, dont les commis responsables de la livraison des messages. Lorsque toutes les activités de ramasse miettes seront mises en place et qu'un nombre suffisant d'agents seront actifs dans l'environnement, il deviendra essentiel d'effectuer des tests de performance permettant de mieux équilibrer l'utilisation des ressources sur le serveur ISAME entre les activités secondaires, dont celles de ramasse miettes, et les activités prioritaires, soit l'échange et le traitement des messages.

- Le temps alloué pour chacune des étapes de vigie avant l'expulsion d'un agent inactif devra être étudié avec grande attention. Idéalement, ce laps de temps devrait être établi dynamiquement à partir du profil de l'agent. Ainsi, plus un agent a « bonne réputation » dans l'environnement, plus le laps de temps devrait être long, cette réputation étant basée sur des éléments tels que la fréquence avec laquelle l'agent interagit dans l'environnement et sa fidélité à l'environnement. De plus, des mécanismes complémentaires aux boîtes aux lettres pourraient être prévus de façon à éviter la destruction complète de résultats obtenus pour un agent utilisateur. À titre d'exemple, une livraison automatique des résultats résiduels lors de l'expulsion pourraient être livrés par courrier électronique à l'utilisateur, l'informant du même coup que son agent, ainsi que ses requêtes persistantes, ont été expulsés de l'environnement, permettant ainsi une fin plus « civilisée » à l'interaction entre un utilisateur et ISAME.

L'architecture UMDL centralise les activités de ramasse miettes dans un agent dédié à ces activités, le *Garbage Collection Agent*. Cet agent n'effectue cependant qu'une partie des tâches de ramasse miettes prévues dans ISAME, soit expulsion d'un agent inaccessible depuis trop longtemps par le désistement automatique de cet agent. Une autre activité de ramasse miettes est implantée afin d'éviter que les agents s'étant engagés à fournir des informations sur une base continue à d'autres agents travaillent inutilement lorsque les agents clients deviennent inaccessibles. Si d'autres activités de nettoyage (tel

que l'envoi explicite d'un avertissement à tous les autres agents pour qu'ils cessent de traiter les demandes de l'agent expulsé) sont prévues, la documentation n'en fait cependant pas état. Il est donc impossible de déterminer ce qui survient avec chacun des messages reliés à cet agent et qui sont toujours en transit ou en traitement dans l'environnement. Il n'est donc pas certain que les activités de ramasse miettes de UMDL soient aussi complètes que celles prévues dans ISAME. D'autre part, il serait souhaitable que le modèle de ISAME soit augmenté, à la manière de UMDL, de façon à sauvegarder un profil des agents expulsés. Ceci permettrait aux agents utilisateurs et information de recommencer soit avec un ancien profil ou à neuf.

Distribution des tâches entre les agents de soutien

L'évaluation de la qualité des services offerts par les agents requêtes et de soutien, ainsi que la distribution des tâches entre l'ensemble de ces agents, ne sera significative que lorsque l'ensemble des agents prévus dans le modèle ainsi qu'un nombre représentatif d'agents utilisateurs et information seront présents dans l'environnement. L'hypothèse voulant que toutes les tâches de service reliées à la recherche d'information ont été pleinement distribuées parmi les agents internes et ce de façon optimale devra être évaluée à nouveau lors de travaux futurs. Néanmoins, la comparaison de ISAME avec l'architecture préparée dans le cadre du projet UMDL permet de supposer que l'architecture de ISAME répond bien aux besoins inhérents à la recherche d'information par système multi-agents. Il est cependant trop tôt pour déterminer avec certitude si l'un des modèles de distribution des tâches est supérieur à l'autre. Une comparaison sommaire

entre les activités complétées par les agents internes d'ISAME et celles supportées par les agents internes de UMDL démontre que la couverture des tâches offerte par l'un et l'autre sont intimement reliées : suivi des agents utilisateurs et information avec profils, prise en charge des requêtes des utilisateurs par les agents internes, sélection des sources pertinentes (par le SFA dans ISAME et par le *Task Planner* dans UMDL) et filtrage des résultats.

Bien qu'il soit comparable au projet UMDL dans ses objectifs et les méthodologies qui lui sont sous-jacentes, le projet ISAME demeure pour l'instant beaucoup plus modeste que le projet UMDL. En effet, le projet UMDL atteint présentement une envergure beaucoup plus importante, étant donné le travail des dizaines de spécialistes du domaine dont il bénéficie. En fait, les modifications apportées à ISAME dans le futur pourront grandement bénéficier des idées mises de l'avant par l'équipe de UMDL.

CHAPITRE 6

CONCLUSION

Ce chapitre propose une synthèse des travaux complétés dans le cadre de ce mémoire ainsi qu'une présentation des travaux demeurant à être complétés afin d'enrichir l'environnement ISAME actuel.

6.1 Synthèse des travaux

Les travaux présentés dans ce mémoire portent sur l'élaboration et l'implantation partielle d'une architecture multi-agents, nommé ISAME, pour la recherche intelligente d'information parmi des sources d'information hétérogènes et distribuées. Cette architecture constitue en fait une bibliothèque virtuelle offrant, à des agents représentant des utilisateurs, un accès simplifié à un ensemble dynamique de sources d'information disponibles sous format électronique ainsi qu'à des services visant à faciliter et optimiser cette recherche d'information.

Comme en fait foi ce mémoire, l'élaboration de l'environnement ISAME s'est effectué en plusieurs étapes, c'est-à-dire analyse du concept de bibliothèque virtuelle, évaluation des systèmes multi-agents comme solution possible aux besoins d'implantation du concept de bibliothèque virtuelle, élaboration d'une architecture multi-agents supportant l'ensemble des besoins et services reliés aux bibliothèques virtuelles,

sélection d'un sous-ensemble d'éléments pour implantation immédiate, programmation et test de ce sous-ensemble, puis analyse de l'architecture.

La grande force de l'architecture ISAME est de simplifier la recherche d'information dans des sources d'information hétérogènes et distribuées en les rendant transparentes à l'utilisateur. En effet, grâce au travail des agents intelligents, les utilisateurs n'ont pas à se soucier de la méthode d'implantation, de la localisation, de la disponibilité et du langage d'interaction propre à chacune des sources. Ceci demeure cependant à être validé de façon plus concrète lors de l'implantation d'agents utilisateurs et information.

De plus, ISAME simplifie le travail de l'ensemble des agents désirant interagir dans son environnement. Ainsi, ISAME se charge d'aiguiller tous les messages échangés entre les agents inscrits auprès de l'environnement. Ceci évite à chacun des agents inscrits de tenir à jour des informations sur les agents enregistrés et sur l'adresse les messages doivent leur être expédiés. De plus, cette livraison centralisée des messages par ISAME tire profit d'un système de boîte aux lettres, ce qui, d'une part, évite aux agents expéditeurs de perdre du temps inutilement à expédier un message à un agent temporairement déconnecté et, d'autre part, évite à un agent destinataire d'être à l'écoute de façon permanente afin d'éviter la perte de messages.

ISAME offre aussi les bénéfices d'une architecture multi-agents ouverte, puisqu'elle permet en tout temps à de nouveaux agents utilisateurs et information de s'inscrire et de quitter l'environnement, ce qui en fait donc un environnement où les ressources sont modifiées dynamiquement selon la disponibilité de chacun. Finalement, ISAME met en

place une série d'activités de ramasse miettes qui visent à optimiser l'environnement et l'utilisation des ressources pour le bénéfice des agents actifs.

Par ailleurs, certains des éléments actuels de ISAME devront être reconsidérés dans le futur afin d'en améliorer l'architecture et les processus. En premier lieu, il faudra réviser la méthode d'expulsion des agents utilisateurs afin d'éviter toute perte catastrophique d'information recueillie pour l'usager. De plus, il serait souhaitable que ISAME non seulement intègre un modèle plus élaboré de profil d'agents, mais qu'il conserve les profils des agents exclus durant une certaine période, de façon à en permettre la récupération lorsqu'un agent se réinscrit dans des délais raisonnables. Lorsque des tests pourront être effectués avec un certain nombre d'agents information, il sera sans doute nécessaire de réviser l'ontologie *VirtualLibrary* afin d'offrir une plus grande richesse dans l'expression des requêtes d'information par l'intégration de concepts tels que ceux utilisés par SQL. Enfin, on pourrait revoir l'implantation des agents CBA et agents registraires de façon à permettre une distribution de ces agents sur différents serveurs et, à la limite, permettre l'interconnexion de différents environnements ISAME, offrant ainsi aux agents l'accès à plus de ressources matérielles et logicielles.

6.2 Travaux futurs

Bien que l'architecture de l'environnement ISAME ait été partiellement concrétisée dans ce mémoire, plusieurs travaux demeurent à être complétés afin de faire de ISAME un environnement complet pour la recherche intelligente d'information. Dans un premier temps, certains des objets implantés jusqu'à ce jour devront être complétés et

testés en profondeur. C'est le cas notamment des structures de données et processus prévus pour les activités de ramasse miettes, des énoncés performatifs demeurant à être implantés dans chacun des types d'agents, ainsi que la gestion des requêtes persistantes par les agents requêtes. Dans un deuxième temps, les objets prévus à l'architecture originale mais toujours absents dans l'environnement actuel devront être mis en place. Il faudra donc développer les agents SFA, RFA, ainsi que quelques agents information et utilisateurs de façon à fournir un ensemble complet d'outils pour la recherche intelligente d'information.

Finalement, il serait souhaitable que plusieurs éléments non abordés dans le cadre de ce mémoire mais pouvant grandement contribuer à l'amélioration de l'architecture ISAME soient explorés : exploration plus approfondie des principes d'économie de marché dans la sélection des agents information, élaboration d'ontologies plus sophistiquées s'inspirant de celles développées dans le cadre de projets comme UMDL et permettant entre autres les recherches dans les bases de données de type relationnelles ou objets, augmentation de la sécurité dans les échanges de messages entre agents, soutien aux agents utilisateurs pour la reformulation de requête auprès des agents requêtes, et exploration de différents modes de livraison de messages et résultats dont le courrier électronique.

Une partie de ces travaux sera complétée dans le cadre du projet SAME (Système d'aide multi-experts) développé par une équipe de chercheurs de la Télé-Université du Québec. SAME est un environnement de soutien au télé-apprentissage. Il propose de

fournir l'assistance nécessaire à tout intervenant (étudiant, tuteur, animateur, etc.) engagé à un titre ou à un autre dans un processus d'apprentissage (Pierre et Hotte, 1996). Cet environnement met à la disposition des intervenants plusieurs outils spécialisés appelés « fonctionnalités » : accueil dans le système (F-Accueil), dépannage informatique (F-Depanne), facilitateur pour le travail de groupe (F-Groupe), gestion de conflits entre participants à un groupe de travail (F-Conflit), etc. L'une de ces fonctionnalités, appelée F-Informe, constitue un outil d'accès guidé à diverses sources d'information : ressources documentaires, information institutionnelle de la Télé-Université, glossaires, offre de cours, etc. ISAME servira de base à l'élaboration de F-Informe, permettant ainsi la mise en oeuvre de certains des éléments manquants dans l'architecture. Finalement, F-Informe fournira un environnement de test réel pour ISAME.

BIBLIOGRAPHIE

ANNET, J. et DUNCAN, K.D. (1967). Task Analysis and Training Design. Occupational Psychology, 41, 211-221.

ARCAND, J.-F. et PELLETIER, S.-J. (1996). Cognition Based Multi-Agent Architecture. In M. Wooldrige, P. Müller & M. Tambe (Eds.), Intelligent Agents Volume II - Proceedings of the 1995 Workshop on Agents Theories, Languages and Architectures (ATAL '95), Lecture Notes in Artificial Intelligence, Springer-Verlag, 267-282.

BENSLIMANE, A., FEKI, T. et GOBLET, X. (1993). Conception de systèmes multi-agents : application aux tuteurs intelligents. Actes du colloque international en informatique cognitive des organisations, Montréal.

BICCHIERI, C., EPHRATI, E., ANTONELLI, A. (1995). Games Servers Play : A Procedural Approach. Workshop on Agents Theories, Languages and Architectures, IJCAI '95, août, Montréal, Québec, 282-291.

BIRMINGHAM, W. P. (1995). An Agent-Based Architecture for Digital Libraries, D-Lib Magazine, juillet.

BROOKS, R. A. (1991). Intelligence Without Reason. Proceedings of IJCAI 91, 569-595.

CHAIB-DRAA, B. (1993). Le contrôle et les communications dans la résolution distribuée de problèmes. ICO, janvier, 6-15.

CARVER, N. et LESSER, V. (1994). Evolution of Blackboard Control Architectures. Expert Systems With Applications, 7, 1-30.

COHEN, P. R. et LEVESQUE, H. J. (1995). Communicative Actions for Artificial Agents. Proceedings of the First International Conference on Multi-Agent Systems (ICMAS '95), juin, San Francisco, 65-72.

DAVIS, A. (1994). General Magic's Agents : a More Flexible EDI. Datamation, 40(16), 15 août, 51-52.

DAI, H. et al. (1993). A Distributed Real-Time Knowledge-Based System and its Implementation Using Object-Oriented Techniques. IEEE, 23-30.

Digital Library Project Research Proposal. Disponible à l'adresse URL suivante : http://http2.sils.umich.edu/UMDL/HomePage.html

EDMONDS, E. A. et al. (1994). Support for Collaborative Design : Agent and Emergence. Communication of the ACM, 37(7), juillet 1994, 41 - 47.

FININ, T. and al. (1993). Specification of the KQML Agent-Communication Language. Disponible à l'adresse URL suivante: http://www.cs.umbc.edu/kqml/mail/kqml

FININ, T. et al. (1993). KQML - A Language and Protocol for Knowledge and Information Exchange. Technical Report CS-94-02, Computer Science Department, University of Maryland, UMBC, États-Unis.

FROST, H. R. (1995). Librairies d'objets JavaAgent. Stanford University. Disponible à l'adresse URL suivante: http://cdr.stanford.edu/ABE/JavaAgent.html.

GENESERETH, M. R. et FIKES, R. E. (1992). Knowledge Interchange Format Version 3.0 Reference Manual. Report Logic-92-1, Stanford University, Logic Group, États-Unis.

GENESERETH, M. R. et KETCHPEL, S. P. (1994). Software Agents. Communications of the ACM, 37(7), juillet, 48 - 53.

JENNINGS, N.R., CORERA, J.M. et LARESGOITI, I. (1995). Developing Industrial Multi-Agent Sytems. Proceedings of the First International. Conference on Multi-Agent Systems (ICMAS '95), juin, San Francisco, 423-430.

KING, James A. (1995). Intelligent Retrieval. AI Expert, 10(1), janvier, 15-17.

KING, James A. (1995). Intelligent Agents : Bringing Good Things to Life. AI Expert, février, 17-19.

KNOBLOCK, Craig A. et ARENS Yigal. (1994). An Architecture for Information Retrieval. AAAI Symposium on Software Agents, mars, Stanford University, 49-56.

LANDER, S. et LESSER, V. (1994). Sharing Meta-Information to Guide Cooperative Search Among Heterogeneous Reusable Agents. Technical Report 94-48, University of Massachusetts, Amherst, Etats-Unis.

LANDER, S. et LESSER, V. (1992). Customizing Distributed Search Among Agents with Heterogeneous Knowledge, Proceedings of the First International Conference on Information and Knowledge Management, novembre, Baltimore, MD.

MAES, Pattie (1994). Agents that Reduce Work and Information Overload. Communications of the ACM, 37(7), juillet, 31-42.

MARX, J. et ROPPEL, S. (1992). WING : An Intelligent Multimodal Interface for a Materials Information System. Proceedings of the 14th Information Retrieval Colloquium. avril, University of Lancaster, 67-78.

MAYFIELD, J., LABROU, Y. et FININ, T. (1995). Evaluation of KQML as an Agent Communication Language. Workshop on Agents Theories, Languages and Architectures, IJCAI '95, août, Montréal, Québec, 282-291.

MOWBRAY, Thomas J. et ZAHAVI, Ron. 1995. The Essential CORBA: Systems Integration Using Distributed Objects. Object Management Group Publishers.

MULLEN, T., Wellman, M. P. (1995). Some Issues in the Design of Market-Oriented Agents. Workshop on Agents Theories, Languages and Architectures, IJCAI '95, août, Montréal, Québec, 282-291.

NII, H. Penny. (1993). Blackboard Systems at the Architecture Level. Expert systems With Applications, 7, 43-54.

PELLETIER S.-J., ARCAND J.-F. et TREVAIL, R. (1995). Using a Hierarchical Task Analysis to Define the Components of an Information Retrieval System Interface. International Ergonomics Association, 16-20 octobre, Rio de Janeiro, Brazil, 339-344.

PELLETIER, S.-J., ARCAND, J.-F. et VELISSARIOS, J. (1996). STEALTH: A Personal Digital Assistant for Information Filtering. The Practical Application of Intelligent Agents and Multi-Agent Technology, avril, Londres, 455-474.

PIERRE, S. et HOTTE, R. (1996). Vers un modèle intégré de support au télé-apprentissage coopératif. Annales de Télécommunications, 51(5-6), 272-287.

POWERS, Andrea. (1994). Hoover Acts as Your Online Scouts. PC World, 12 no 8, août, 9.

ROSENCHEIN, J. S. et ZLOTKIN, G. (1994). Designing Conventions for Automated Negociation. AI Magazine, automne, 29 - 46.

RUMBAUGH, J. et al. (1991). Object-Oriented Modeling and Design. Prentice Hall.

SAVETZ, K. (1996). Here Come the Knowbots! Disponible à l'adresse URL suivante: http://redwood.northcoast.com/savetz/articles/knowbots.html.

SHOHAM, Y. (1992). Multi-Agent Research in the Knobotics Group. Artificial Social Systems, Cristiano Castelfranchi and Eric Werner (eds). Proceedings of the 4th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW '92, Italie.

SHOHAM, Y. (1993). Agent-Oriented Programming. Artificial Intelligence, Elsevier Science Publishers, 51 - 92.

SINGH, M. P. (1991) Towards a Formal Theory of Communication for Multiagent Systems. Proceedings of the 12th International Conference on Artificial Intelligence (IJCAI - 91), 69-74.

SINGH, M. P. (1993). Declarative Representations of Multiagent Systems. IEEE Transactions on Knowledge and Data Engineering, 5(5), octobre, 721-740.

SINGH, M. P. (1994). Multiagent Systems : A Theoretical Framework for Intentions, Know-How, and Communications. Springer-Verlag Publishers.

SUDKAMP, T. A. (1991). Languages and Machines. Addison-Wesley Publishing Company.

URZELAI, K. et GARIJO, F. J. (1992). MAKILA : A Tool for the Development of Cooperative Societies. Artificial Social Systems, Cristiano Castelfranchi and Eric Werner (eds). Proceedings of the 4th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW '92, Italy. 311-323.

VALAUSKAS, E. J. (1994). AppleSearch : How Smart Is Apple's Intelligent Agent? Online, juillet, 52-64.

(1995) Verity agents on the Internet. The Seybold Report on Desktop Publishing, 9(6), février, 34.

(1994) Verity's InfoAgents. Which Computer?, 17(7), juillet, 36.

VOSSOS, G. et al. (1993). Developing Co-operating Legal Knowledge Based Systems. Proceedings of the Third Congress of the Italian Association for Artificial Intelligence, AI*IA '93, octobre, Torino, Italy, 160-165.

WOOLDRIDGE, M. (1992). The Logical Modelling of Computational Multi-Agent Systems. Thèse de doctorat, Department of Computation, University of Manchester, Etats-Unis, août.

WOODRIDGE, M. and JENNINGS, N. R. (1994). Intelligent Agents : Theory and Practice. Knowledge Engineering Review, 10(2), 1995.

ANNEXE A

PRINCIPES DE BASE DU LANGAGE KQML

Le langage KQML définit en quelque sorte l'enveloppe dans laquelle les agents encapsulent les messages qu'ils s'échangent, le protocole de communication. Chacun des énoncés KQML est échangé sous la forme d'une chaîne de caractères ASCII, cette chaîne suivant une syntaxe précise. Trois niveaux de communication sont associés à un énoncé KQML : message, communication et contenu (Finin, 1993). Le niveau message forme le cœur du message, celui qui identifie le protocole utilisé pour la livraison du message ainsi que l'énoncé performatif. Le niveau communication comprend toutes les composantes de bas niveaux de la communication, tels les identificateurs des expéditeurs et destinataires. Finalement, le niveau contenu représente le contenu proprement dit du message. Ce contenu n'est pas traité directement par le protocole KQML, sauf lorsqu'il s'agit de déterminer sa longueur. Les deux sections qui suivent décrivent tour à tour le langage KQML et le format de chacun des énoncés performatifs retenus dans le cadre de ISAME.

1. Le langage KQML

Chaque énoncé KQML débute par l'identificateur de l'énoncé performatif, suivi d'une série de paramètres indexés par des mots clés, chaque mot clé débutant par « : » et

étant suivi d'une valeur associée. L'ordre des paramètres n'est pas important. Le format BNF d'un énoncé KQML s'énonce comme suit :

<performative> ::= (<word> {<whitespace> :<word> <whitespace> <expression>}*)

<expression> ::= <word> | <quotation> | <string> |

(<word> {<whitespace> <expression>}*)

<word> ::= <character> <character*>

<character> ::= <alphabetic> | <numeric> | <special>

<special> ::= < | > | = | + | - | * | / | & | ^ | ~ | _ | @ | $ | % | : | . | ! | ?

<quotation> ::= '<expression> | `<comma-expression>

<comma-expression :: = <word> | <quotation> | <string> | ,<comma-expression>

(<word>

{<whitespace> <comma-expression>}*)

<string> ::= "<stringchar>*" | #/<digit> <digit>*"<ascii>*

<stringchar> ::= \<ascii> | <acii>-\-<double-quote>

Bien que l'ensemble des énoncés performatifs KQML soit extensible, les auteurs ont identifié et défini de façon formelle une trentaine d'énoncés, dont la liste peut être trouvée dans les documents de définition de KQML disponibles sur le Web. Si que les agents puissent choisir quels énoncés ils désirent supporter, ils doivent cependant les implanter de façon standard. Ces énoncés ont été classifiés par les auteurs selon différentes catégories : *énoncés de base* (tell, deny, untell), *énoncés de systèmes de gestion de bases de données* (insert, delete, delete-one, delete-all), *énoncés de requête de*

base (evaluate, reply, ask-if, ask-about, ask-one, sorry), *énoncés de réponse de base* (error, sorry), *énoncés de requête multi-réponses* (stream-about, stream-all, eos), *énoncés directifs de base* (achieve, unachieve), *énoncés générateurs* (standby, ready, next, rest, discard, generator), *énoncés de publication de capacité* (advertise), *énoncés de notification* (subscribe, monitor), *énoncés de réseau* (register, unregister, forward, pipe, break, transport-address), et *énoncés de facilitation* (broker-one, broker-all, recommend-one, recommend-all, recruit-one, recruit-all).

Onze mots réservés servent à indexer les paramètres associés à un énoncé performatif. Ce sont *content*, *force*, *from*, *in-reply-to*, *language*, *name*, *ontology*, *receiver*, *reply-with*, *sender*, *to* et sont définis comme suit :

- content — L'information à laquelle le message s'applique.
- force — Indique si l'expéditeur se réserve le droit d'annuler le message à un moment ultérieur.
- from — L'expéditeur virtuel du message.
- in-reply-to — Identifie l'étiquette associée au message pour laquelle une réponse est envoyée.
- language — Identifie le langage dans lequel l'information du paramètre *content* est exprimé ; si le paramètre n'apparaît pas, ce paramètre prend comme valeur par défaut *KQML*.
- name — Le nom sous lequel un agent est enregistré.

- ontology Identifie l'ontologie (un ensemble de mots et leurs définitions) utilisée pour exprimer l'information du paramètre *content*.
- receiver Le destinataire réel du message.
- reply-with L'étiquette à utiliser lors d'une éventuelle réponse au message expédié.
- sender L'expéditeur réel du message.
- to Le destinataire virtuel du message.

À titre d'illustration, voici comment se compose les énoncés performatifs tell, advertise et transport-address :

(**advertise** :content <performative>
:language KQML
:ontology <word>
:force <word>
:sender <word>
:receiver <word>)

(**tell** :content <expression>
:language <word>
:ontology <word>
:in-reply-to <expression >
:force <word>
:sender <word>

:receiver <word>)

(**transport-address** :name <word>

:content <word>

:language <word>

:ontology <word>)

Certains énoncés performatifs incluent comme contenu du paramètre « :content » un autre énoncé performatif destiné à un agent tiers. Cette composition d'énoncés visent principalement à faciliter les services de facilitation (« brokerage ») et de notification. Ainsi, l'énoncé « broker-one » comprend dans le paramètre « :content » l'énoncé performatif devant être traité par l'agent recruté par le destinataire au nom de l'expéditeur. Cette composition d'énoncés explique aussi le besoin d'instantiation du paramètre « :language » à la valeur KQML. Voici un exemple d'imbrication d'énoncés performatifs :

(**broker-one** :content (ask-about :content <expression>

:language <word>

:ontology <word>

:reply-with <expression>

:sender <word>)

:language KQML

:ontology <word>

:sender <word>

:receiver <word>)

2. Énoncés performatifs retenus pour ISAME

- **advertise**

But : L'expéditeur (« :sender ») informe un destinataire (« :receiver ») des messages performatifs qu'il peut manipuler.

Format : *(advertise :content* <performative> *:language* KQML *:ontology* <word> *:force* <word> *:sender* <word> *:receiver* <word>*)*

- **ask-about**

But : L'expéditeur demande au destinataire d'expédier toute information contenue dans sa base de connaissances correspondant à la demande incluse dans le paramètre « :content ». Les résultats sont fournis sous forme d'une collection dans un seul énoncé. Le langage de transmission de l'information doit donc permettre l'expression d'un concept de collection (liste, ensemble, ...).

Format : *(ask-about :content* <expression> *:language* <word> *:ontology* <word> *:reply-with* <expression> *:sender* <word> *:receiver* <word>*)*

- **broker-one**

But : L'expéditeur demande au destinataire de faire parvenir l'énoncé performatif contenu dans le paramètre « :content » à un seul agent capable de le traiter. Le destinataire doit se charger d'acheminer les résultats à l'expéditeur.

Format : (broker-one :content <expression> *:language* KQML *:ontology* <word> *:reply-with* <expression> *:sender* <word> *:receiver* <word>*)*

- **deny**

But : L'expéditeur informe le destinataire qu'un message expédié plus tôt n'est plus valide.

Format : *(deny :content* <performative> *:language* KQML *:ontology* <word> *:in-reply-to* <expression> *:force <word> :sender* <word> *:receiver* <word>*)*

- **eos**

But : L'énoncé eos (« End of Stream ») indique que la séquence de résultats attendus comme résultats à un message demandant plusieurs résultats (exemple : stream-about) est complétée.

Format : *(eos :in-reply-to* <expression> *:sender* <word> *:receiver* <word>*)*

- **error**

But : L'expéditeur informe le destinataire qu'un message expédié plus tôt, identifié par le paramètre « :in-reply-to » n'a pu être compris ou que le message performatif auquel ce message référait n'est pas valide. Le paramètre « :code » comprend un code numérique classifiant l'erreur. Le paramètre « :comment » inclus une chaîne de caractères décrivant pourquoi l'énoncé est considéré non valide.

Format : *(error :in-reply-to* <expression> *:sender* <word> *:receiver* <word> *:comment* <string> *:code* <integer>*)*

- **evaluate**

But : L'expéditeur demande au destinataire de simplifier le contenu du paramètre « :content ».

Format : *(evaluate :content* <expression> *:language* <word> *:ontology* <word> *:reply-with* <expression> *:sender* <word> *:receiver* <word>*)*

- **recommend-all**

But : L'expéditeur demande au destinataire de recommander les agents pouvant répondre à la demande contenue dans le paramètre « :content ».

Format : *(recommend-all :content* <expression> *:language* KQML *:ontology* <word> *:reply-with* <expression> *:sender* <word> *:receiver* <word>*)*

- **register**

But : L'expéditeur informe le destinataire qu'il peut livrer des énoncés performatifs à l'agent connu sous le nom inscrit dans le paramètre « :name », incluant le cas où l'expéditeur inscrit son propre nom.

Format : *(register :name* <word> *:sender* <word> *:receiver* <word>*)*

- **reply**

But : L'expéditeur retourne une réponse au destinataire concernant un message reçu du destinataire et identifié par la paramètre « :in-reply-to ».

Format : *(reply :content* <expression> *:language* <word> *:ontology* <word> *:in-reply-to* <expression> *:force* <word> *:sender* <word> *:receiver* <word>*)*

- **sorry**

But : L'expéditeur informe le destinataire que le message identifié par le paramètre « :in-reply-to » a été compris, mais qu'aucune réponse ne peut être fournie. Le

paramètre « :comment » inclus une chaîne de caractères décrivant pourquoi l'énoncé ne peut être traité.

Format : (*sorry :in-reply-to* <expression> *:sender* <word> *:receiver* <word> *:comment* <string>*)*

- **stream-about**

But : L'expéditeur demande au destinataire d'expédier toute information contenue dans sa base de connaissances correspondant à la demande incluse dans le paramètre « :content ». Les résultats sont fournis sous forme d'une série d'énoncés dont les contenus forment une collection de résultats.

Format : *(stream-about :content* <expression> *:language* <word> *:ontology* <word> *:reply-with* <expression> *:force* <word> *:sender* <word> *:receiver* <word>*)*

- **subscribe**

But : L'expéditeur souhaite que le destinataire l'informe de tout changement à la réponse qui correspond à l'énoncé performatif contenu dans le paramètre « :content ». Des exemples d'énoncés pouvant être inclus dans le paramètre « :content » sont *ask-about et stream-about*.

Format : *(subscribe :content* < performative > *:language* KQML *:ontology* <word> *:reply-with* <expression> *:force* <word > *:sender* <word> *:receiver* <word>*)*

- **tell**

But : L'expéditeur désire transmettre l'information contenue dans le paramètre « :content » au destinataire.

Format : *(tell :content* <expression> *:language* <word> *:ontology* <word> *:in-reply-to* <expression> *:force* <word> *:sender* <word> *:receiver* <word>*)*

- **transport-address**

But : L'expéditeur connu sous le nom identifié dans le paramètre « :name » désire changer son adresse physique à celle inscrite dans le paramètre « :content ». Cet énoncé peut être inclus dans d'autres énoncés dont *tell.*

Format : *(transport-address :name* <word> *:content* <expression> *:language* <word> *:ontology* <word>*)*

- **unregister**

But : L'expéditeur connu sous le nom inscrit dans le paramètre « :name » désire annuler son inscription auprès du destinataire.

Format : *(unregister :name* <word> *:sender* <word> *:receiver* <word>*)*

ANNEXE B

LANGAGE ISAME-L

La syntaxe générale du langage ISAME-L est inspirée de celle de KQML. En effet, ce langage est basé sur une liste de mots réservés auxquels sont associés des paramètres de différents types. Le langage ISAME-L supporte les notions d'ensembles nécessaires aux besoins de l'énoncé « ask-about ». Le format BNF du langage ISAME-L est le suivant :

<expression> ::= (<performative> <espace> :content (<singleton>| <ensemble>))

<singleton> ::= <mot-valeur> {<espace> <mot-valeur>}*

<ensemble> ::= [<singleton>] [<singleton>]*

<mot-valeur> ::= {:<mot> <espace> {<valeur> | <vecteur-valeurs>}}

<valeur> ::= <numérique> | <mot> |<phrase>

<vecteur-valeurs> ::= [{<valeur> }{<espace> <valeur>}*]

<mot> ::= <lettre> {<caractère>}*

<phrase> ::= "<ascii> <ascii>*"

<caractère> ::= <lettre> | <lettre-spéciale> | <chiffre>

<lettre> ::= a |... | z | A | ... | Z

<lettre-spéciale> ::= < | > | = | + | - | * | / | & | ^ | ¨ | _ | @ | $ | % | : | . | ! | ?

<numérique> ::= <chiffre><chiffre>* {<.> <chiffre> <chiffre>*}

<chiffre> ::= 0 | 1 | ... | 9

<performative> ::= query | update

<adresse_tcp_ip > ::= <chiffre><chiffre><chiffre>.<chiffre><chiffre><chiffre>.
<chiffre><chiffre>.<chiffre><chiffre><chiffre>:
<chiffre><chiffre><chiffre><chiffre>
(ex.: 206.167.88.156:5001)

ANNEXE C

ONTOLOGIE BIBLIO-VIRTUELLE

L'ontologie Biblio-virtuelle vise à supporter l'échange d'information entre agents dans le but de répondre à des requêtes d'information exprimées par des agents de types UA_x au nom d'utilisateurs. Cette ontologie comprend donc les mots réservés permettant de « typifier » l'information, ainsi que les valeurs permises pour ces mots réservés lorsqu'il y a lieu. Les valeurs des paramètres associés aux mots réservés sont restreintes à celles explicitement mentionnées ci-dessous. L'information que cette ontologie supporte peut être divisée en trois grande catégories : codes d'erreurs, requête d'information et résultat de requête. Les mots réservés et valeurs associées pour chacune des catégorie sont présentés dans le Tableau C.1. La définition des expressions données en format BNF correspondent à celles du langage ISAME-L. Cette ontologie ne supporte pas les expressions de type SQL.

Tableau C.1. Ontologie Biblio-virtuelle.

Catégorie	***Mot réservé***	***Signification***	***Valeurs associées***
query		L'ensemble du contenu de la requête.	**:content** (<singleton> | <ensemble>)
	content	Les mots contenus dans la requête de l'utilisateur.	[<mot> <mot>*] | <phrase>

Tableau C.1. Ontologie Biblio-virtuelle (suite).

Catégorie	*Mot réservé*	*Signification*	*Valeurs associées*
	modifier	Les combinaisons possibles sur les mots de la requête.	<valeur> ou <vecteur-valeurs>: **and or near sentence extend restrict**
	location	Le lieu préféré pour la localisation des mots de la requête dans les résultats.	<valeur> ou <vecteur-valeurs>:: **title sub_title author summary anywhere**
	format	Les formats de résultats acceptés.	<valeur> ou <vecteur-valeurs>: **html ascii sgml**
	relative_time	La durée de traitement permise pour la requête.	<relative-time>
	absolute_time	L'heure absolue de fin de traitement de la requête.	<absolute-time>
	information	Les informations souhaitées pour chacun des résultats retournés.	<valeur> ou <vecteur-valeurs>: **author location url format modifier absolute_rating relative_rating percent_rating source]**
	update_ frequency	La fréquence avec laquelle une requête récurrente doit être traitée.	<numérique> de 6 caractère : (jjmmhhss: jours, minutes, heures, secondes)
update		L'ensemble d'un résultat de requête.	<singleton> | <ensemble>

Tableau C.1. Ontologie Biblio-virtuelle (suite).

Enoncé Performatif	*Mots réservés associés*	*Signification*	*Valeurs associées*
	content	Le contenu du résultat, selon le format identifié par le paramètre format.	<phrase>
	author	L'auteur du résultat trouvé.	<phrase>
	url	L'adresse URL à laquelle le résultat peut être trouvé.	<phrase >
	location	La localisation des mots de la requête dans les résultats.	<valeur> ou <vecteur-valeurs>: **title sub_title author summary anywhere**
	format	Le format de la phrase du paramètre content.	**html | ascii | sgml**
	modifier	Les combinaisons appliquées sur les mots de la requête.	<valeur> ou <vecteur-valeurs>: **and or near sentence extend restrict**
	absolute_rating	La valeur de pertinence donnée à ce résultat par la source, selon son échelle interne.	<numérique>
	relative_rating	La valeur de pertinence donnée à ce résultat par rapport aux autres résultats donnés.	<numérique>

Tableau C.1. Ontologie Biblio-virtuelle (suite).

Énoncé Performatif	*Mots réservés associés*	*Signification*	*Valeurs associées*
	percent_rating	La valeur de pertinence donnée à ce résultat en pourcentage du maximum permis par la source.	<numérique>
	source	La source du résultat, pouvant comprendre l'agent et la source d'information.	<phrase>
error		Un code d'erreur suit.	<phrase> | <numérique>

ANNEXE D

ONTOLOGIE AGENT

L'ontologie Agent vise à supporter l'échange d'information entre agents dans le but d'échanger de l'information sur les agents de l'environnement ISAME. Cette ontologie comprend donc les mots réservés permettant de « typifier » l'information sur les agents, ainsi que les valeurs permises pour ces mots réservés lorsqu'il y a lieu. Les mots réservés et valeurs associées pour l'ontologie Agent sont présentés dans le Tableau D.1. La définition des expressions données en format BNF correspondent à celles du langage ISAME-L.

Tableau D.1. Ontologie Agent.

Énoncé Performatif	***Mots réservés associés***	***Signification***	***Valeurs associées***
query		L'ensemble du contenu de la requête.	**:content**(<singleton> | <ensemble>)
	agent_ID	L'identificateur de l'agent pour lequel l'information est requise. Dans le cas où le mot réservé n'est pas utilisé, la requête s'applique à tous les agents.	<valeur> | [<valeur> <valeur>*]

Tableau D.1. Ontologie Agent (suite).

Énoncé Performatif	*Mots réservés associés*	*Signification*	*Valeurs associées*
	information	L'information souhaitée sur l'agent identifié.	<valeur> ou <vecteur-valeurs>: **description performative status percentage_rating relative_ranking address format**
update		Résultat de requête ou mise à jour d'information.	**:content** (<singleton> | <ensemble>)
	description	La description textuelle de l'agent telle qu'il l'a fournie.	<phrase>
	address	L'adresse à laquelle cet agent peut être rejoint	<adresse_tcp_ip>
	performative	La liste des énoncés performatifs KQML supportés par l'agent.	[<performative> <performative>*]
	status	Le statut de l'agent dans l'environnement ISAME.	**active** | **sleeping** | **lost** | **dead** | **expulsed**
	percentage_ rating	Évaluation de l'efficacité de cet agent pour service rendu, sur une échelle de 1 à 100.	<numérique>

Tableau D.1. Ontologie Agent (suite).

Énoncé Performatif	*Mots réservés associés*	*Signification*	*Valeurs associées*
	relative_ ranking	Évaluation de l'efficacité de cet agent pour service rendu, sur une échelle de 1 à 100.	<numérique>
	format	Formats d'information manipulés par cet agent.	<valeur> ou <vecteur-valeurs>: **html ascii sgml**
error		Un code d'erreur suit.	<phrase> | <numérique>

ANNEXE E

QUELQUES PROJETS D'APPLICATION

Plusieurs projets de bibliothèques virtuelles tentent de mettre à contribution les systèmes multi-agents. La présente section fait un survol de quelques-uns de ces projets. Certains d'entre eux visent à développer des agents information dédiés à une source donnée. D'autres s'attaquent au vaste problème de l'intégration des sources d'information hétérogènes distribuées, intégrant des mécanismes d'apprentissage pour permettre aux systèmes de s'adapter au profil de l'utilisateur.

University of Michigan Digital Library - UMDL (Digital Library Project Research Proposal, 1995)

Le projet UMDL (University of Michigan Digital Library) mis de l'avant par l'université du Michigan constitue sans doute le plus ambitieux et le plus complet des projets appliquant la méthodologie multi-agents aux SRII. Ce projet vise en effet à permettre l'intégration de tout genre de source d'information. Il tente aussi d'incorporer une modélisation complète des services offerts par les bibliothécaires traditionnels : sélection des sources les plus pertinentes, intégration et filtrage des résultats, capacité d'adapter les services aux besoins de l'utilisateur, pro-activité et autonomie des processus de recherche d'information, etc. Le but ultime de ce projet est d'établir une

architecture pouvant supporter la création d'une bibliothèque virtuelle à l'échelle mondiale. Entre autres axes de recherche, les chercheurs du projet UMDL s'intéressent :

1 - À la définition d'outils pour la manipulation de requêtes et la visualisation d'information. Ces outils doivent :

- s'adapter aux besoins de l'utilisateur;
- s'adapter au contenu des documents;
- être munis d'interfaces pouvant s'adapter au niveau d'expérience de l'utilisateur (novice/expert);
- intégrer des processus permettant l'analyse des habitudes d'utilisation des utilisateurs afin d'aider ces derniers à améliorer leur performance;
- fournir de l'aide à l'utilisateur désirant naviguer à travers les documents;
- permettre la visualisation de l'information selon différents points de vue.

2 - Au développement de mécanismes offrant des services de recherche et d'extraction d'information. Ces mécanismes doivent :

- permettre la création d'agents capables d'effectuer de la recherche d'information sur des sources d'information distribuées et capables de prendre des décisions à partir d'informations incomplètes;
- tirer profit des théories développées en économie en les appliquant aux problèmes de distribution de ressources.

3 - Aux structures de données et de documents :

- définir des méthodes pour la structuration et l'extraction de méta-données et de structures de base pour les documents et les médias continus (vidéo, audio, ...);
- tirer profit des standards de définition de documents (ex. : SGML, HTML) afin de raffiner leurs stratégies de recherche;
- développer des méthodes permettant de mémoriser les sessions de recherche d'information complétées et réutiliser ces sessions afin d'améliorer la performance du système.

4 - Au développement d'ontologies pouvant faciliter l'échange d'information entre les agents d'une fédération d'agents distribués sur un réseau informatique.

L'architecture de UMDL comprend trois catégories d'agents : les UIA (User-Interface Agent), les CIA (Collection-Interface Agents) et les médiateurs.

UIA :

Dans UMDL, la personnalisation du système pour un utilisateur s'effectue par l'entremise des UIA. Ces agents regroupent une série de fonctions visant à fournir au système le maximum d'information concernant l'utilisateur. Au fur et à mesure des interactions avec l'utilisateur, UMDL apprend les habitudes de recherche et s'adapte en conséquence. L'un des buts visés par ceci est de permettre à un expert d'un domaine d'entraîner le système à effectuer de la recherche d'information selon des critères donnés, de façon à ce qu'un novice puisse en bénéficier lors de ses

propres recherches. Les UIA permettront aussi à l'utilisateur de reformuler sa requête ou de définir une requête à partir d'une autre déjà connue par le système.

L'agent d'entrevue est une instance spécialisée de UIA. Cet agent implante des services de guidage et d'aide fournis à l'utilisateur. Ces services seront rendus disponibles peu importe le type ou le genre de sources d'information manipulées. Par l'entremise de cet agent, le système pourra obtenir les informations suivantes sur l'utilisateur : sujet d'intérêt, format désiré, type d'utilisation potentielle des documents, niveau d'abstraction de l'information, contraintes de temps, etc.

CIA :

Le but visé par les CIA est de garder les agents d'interface indépendants du contenu des corpus d'information. Chaque CIA est responsable d'un corpus d'information et est chargé de maintenir un lien de communication avec le reste du système. Ce CIA doit être capable de traduire une requête d'un langage commun vers sa représentation interne, de résoudre les incohérences entre les connaissances du monde externe et celles de son corpus, etc. Dans le contexte de UMDL, chaque CIA est développé par le fournisseur du corpus d'information. De cette façon, des contraintes minimales sont imposées au concepteur. Entre autres connaissances, le CIA doit être en mesure d'identifier le contenu de son corpus ainsi que son organisation interne. Il doit être capable de communiquer avec les autres CIA du système, être capable de fournir des méta-connaissances sur lui-même (description de son contenu, stratégies de recherche disponibles et outils de recherche, etc.).

Médiateurs :

Ces agents utilitaires ont la responsabilité de faire le lien entre les UIA et les CIA. Des exemples d'activités complétées par les médiateurs sont : structurer la requête pour les CIA, surveillance du traitement des requêtes, prise de décision de terminaison des requêtes, traduction des requêtes entre agents utilisant des ontologies différentes, décomposition des requêtes en sous-requêtes, intégration et filtrage des résultats, application des principes d'économie à l'allocation des ressources, etc.

L'architecture UMDL intègre à ce jour une variété de sources d'information disponibles à l'université de Michigan ainsi que des outils de recherche sur l'Internet. Le domaine des sources d'information est restreint aux sciences de la terre et de l'espace. L'environnement de test inclut des classes d'élèves du secondaire, leurs professeurs et des scientifiques.

De plus amples informations sur cette architecture peuvent être trouvées à l'adresse http suivante : http://http2.sils.umich.edu/UMDL/HomePage.html

(Vossos, 1993)

Beaucoup plus modeste que l'UMDL, le projet de Vossos vise à fournir aux avocats un assistant numérique pour les aider dans leurs recherches d'information. Ce système comprend plusieurs agents de recherche dont l'intelligence est modélisée selon des techniques de règles ou de cas. Trois agents font partie du système : CaRE (Case

Representation Editor Agent), RuBA (Rule Based deductive Agent) et CaBA (Case Based Agent). La liaison entre le système et l'interface utilisateur se fait par l'entremise de CaRE. Selon Vossos, la modélisation des agents selon ces techniques d'apprentissage est très flexible puisqu'elle permet de copier le processus de raisonnement typique appliqué en droit, la jurisprudence s'apparentant à la technique de cas, les relations de cause à effet à la technique de règles. Quelques agents utilitaires sont ajoutés au système afin de faciliter la communication entre les trois agents de base.

Wing-IIR (Marx, 1992)

La particularité du projet Wing-IIR est d'offrir à l'utilisateur une interface de requête multimodales. Parmi les modalités testées par Marx, on retrouve : langage naturel, manipulation directe, langages formels tels SQL, navigation par hypertexte, recherche par hiérarchie de documents et requête à partir d'exemples. Voici trois des conclusions tirées par Marx face à l'utilisation de la multimodalité:

- la possibilité pour l'utilisateur d'exprimer sa requête en langage naturel lui évite de s'adapter à un nouveau langage propre à l'utilisation du système;
- le modèle hiérarchique de requête n'était utilisé que par les individus à la recherche d'un document spécifique déjà connu;
- la requête à partir d'exemples convient à la reformulation d'une requête en cours, puisqu'elle permet d'utiliser une partie des résultats du cycle de requête précédent comme élément de requête courant.

AppleSearch (Valauskas, 1994)

AppleSearch est un outil de SRII spécifique aux ordinateurs MacIntosh reliés par réseaux AppleShare. Cet agent fonctionne de la façon suivante : l'un des ordinateurs du réseau est choisi comme serveur AppleSearch et devient responsable de l'indexation des textes des utilisateurs reliés au serveur. Une fois créés, ces indexes, basés sur l'occurrence des mots, sont mis à la disposition des agents de recherche (" reporters ") situés sur les ordinateurs clients. Lorsqu'un utilisateur veut faire une recherche de texte, il fait appel à son agent de recherche. En utilisant une interface combinant traitement de langage naturel et opérateurs booléens (ces opérateurs sont facultatifs), l'utilisateur fait part de sa requête à l'agent. Ce dernier part alors sur le réseau à la recherche d'information pouvant correspondre à la requête. Les documents trouvés sont présentés à l'utilisateur en ordre de pertinence. La pertinence des documents est basée sur quatre critères : la fréquence d'occurrence des mots de la requête, la présence de termes comportant un haut degré de précision (ex. : " PowerPC " est plus précis que " ordinateur "), la longueur du document (plus le document est long, plus il est jugé pertinent), et la distance séparant les mots pertinents à l'intérieur du texte.

Topic (*Verity agents on the Internet.; Verity's InfoAgents*)

Topic est un serveur Web possédant un outil de recherche combinant hypertexte et agents information. Topic offre deux types de services aux utilisateurs : recherche d'information à partir de requêtes ponctuelles ou surveillance de sources d'information

pour détecter l'information reliée à un sujet donné. La requête peut être définie en reliant des mots à l'aide d'opérateurs booléens ou en formulant des requêtes plus complexes comprenant mots et phrases décrivant les concepts d'intérêt. L'utilisateur est libre d'apposer des poids différents aux mots et phrases utilisés dans la requête.

Le système inclut différents types d'agents semi-autonomes implantant des fonctionnalités de navigation, recherche, filtrage, intégration et sauvegarde d'information. Ces agents sont classés dans trois grandes catégories :

- les surveillants : surveillent les flux continus d'information;
- les chercheurs : naviguent à travers les sources d'information;
- les analystes : filtrent les informations pour l'utilisateur.

Hoover (Powers, 1994)

La caractéristique principale de Hoover est d'être un SRII spécialisé en travail collaboratif. Comme les autres outils de SRII, il fournit sous une seule interface l'accès à différentes sources d'information multimédia, surveille les corpus, recherche, organise et intègre des informations pour l'utilisateur. Il supporte les requêtes en langage naturel, requêtes à partir d'exemples, filtrage à partir de mots auxquels on peut rattacher des poids, recherche plein texte et opérateurs booléens. Il permet l'accès à des bases en ligne et à l'Internet. Hoover s'intègre dans le cadre d'utilisation de l'outil Lotus Notes. L'information rapportée à l'utilisateur est triée selon la date, l'industrie ou le segment de marché. Hoover est présentement disponible sur le marché.

WebWatcher

WebWatcher, développé par l'Université Carnegie Mellon, est un agent dédié à guider les utilisateurs dans un tour du web. L'utilisateur informe l'agent sur le type d'information qu'il recherche. L'agent accompagne l'utilisateur alors qu'il navigue sur le web et attire son attention sur les hyperliens susceptibles de l'intéresser. Les connaissances de l'agent sont basées sur la rétroaction donnée par l'utilisateur lors des tours précédents.

Site Web :

http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-6/web-agent/www/project-home.html

ANNEXE F

MODÈLE ORIENTÉ OBJET PARTIEL POUR LA PHASE D'IMPLANTATION

Cette annexe présente un modèle objet partiel pour les classes implantées pour ISAME. Les Figures F.1 à F.5 présentent les hiérarchies de classes, les relations d'associations et d'aggrégation qui les relient, ainsi que les variables internes à chacune d'entre elles. Les classes dont les noms sont précédées d'un astérisque (*) proviennent d'une librairie externe.

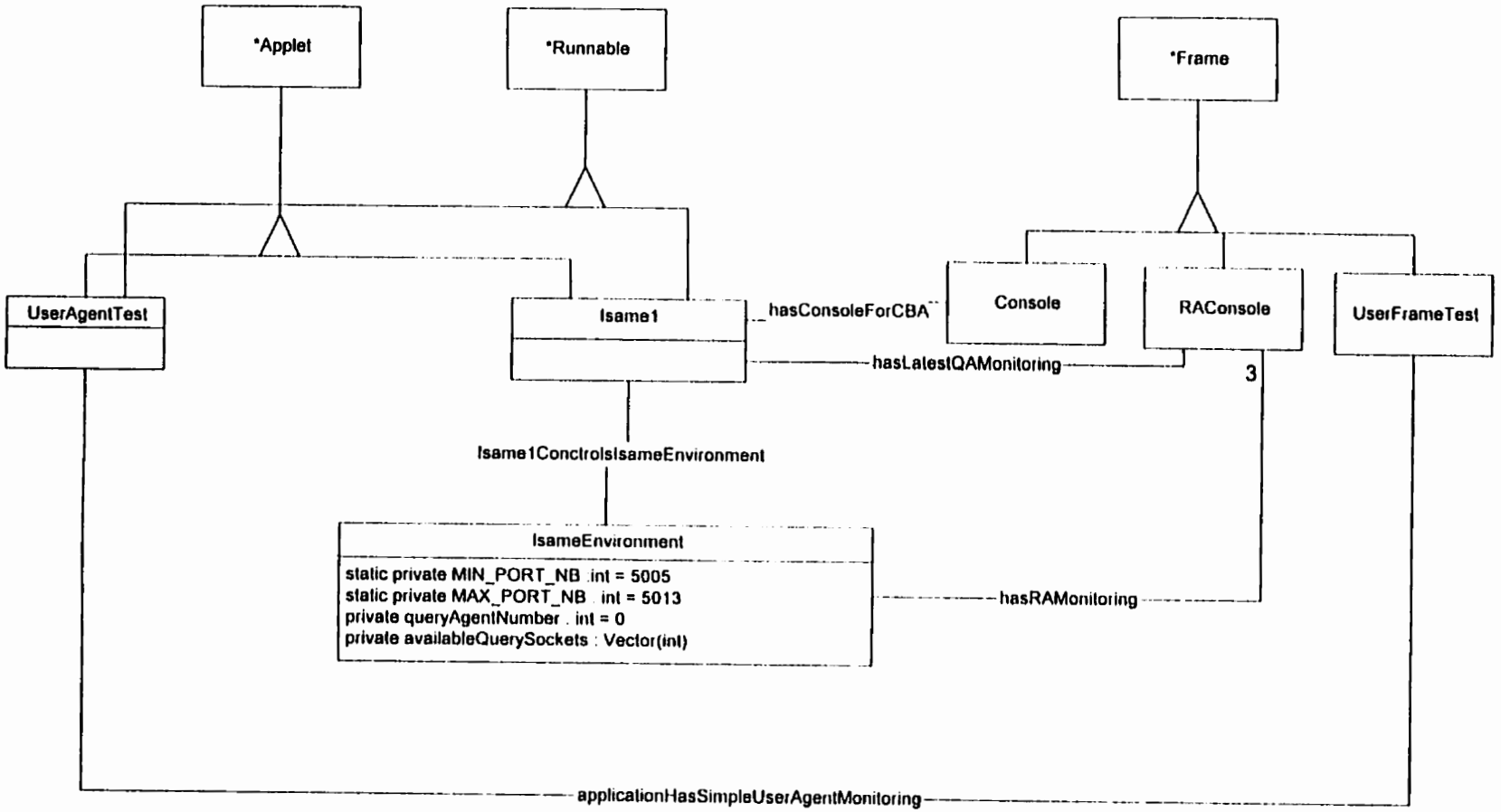


Figure F.1. Modèle objet partiel pour les applications de test de l'environnement Isame.

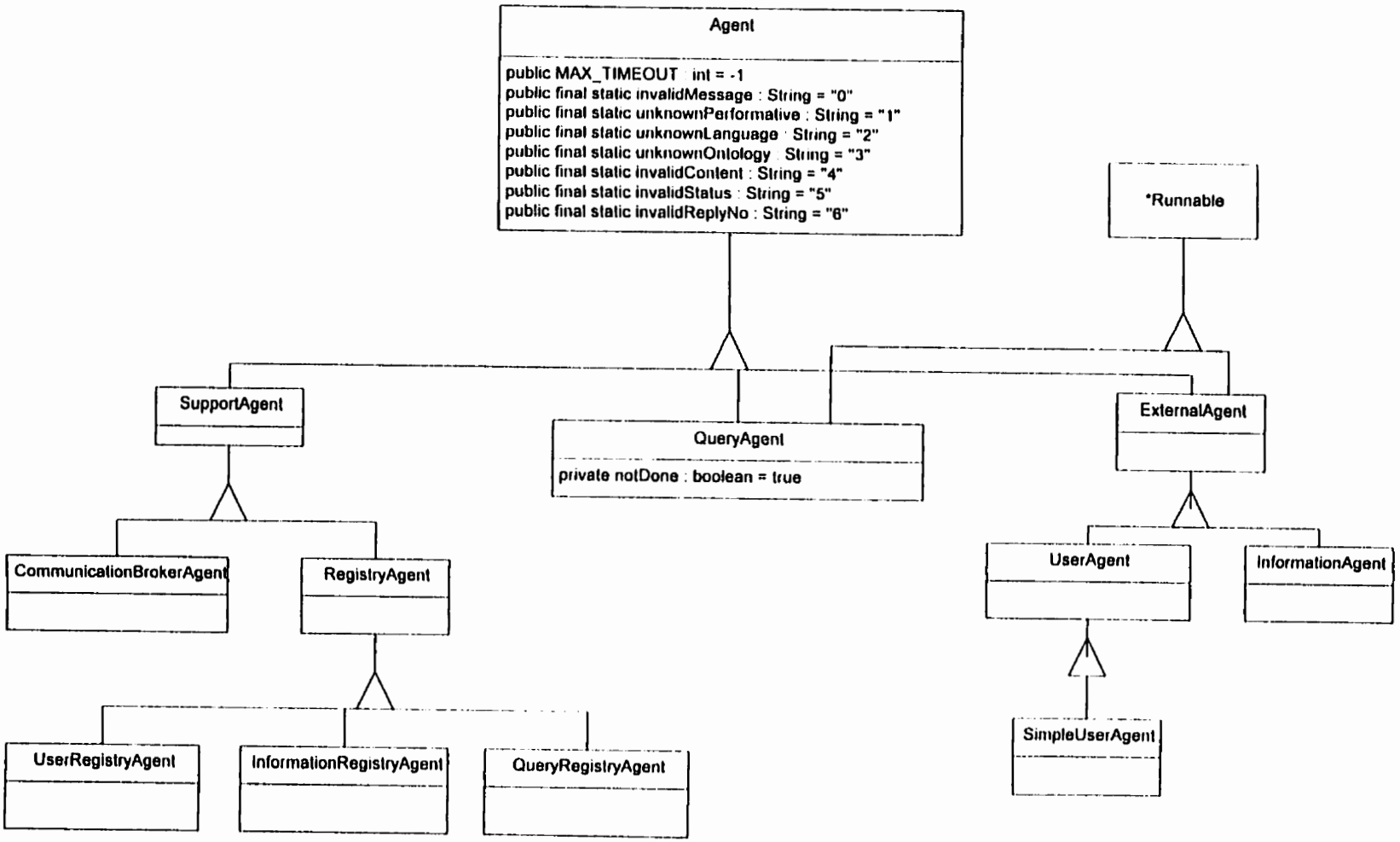


Figure F.2. Hiérarchie des classes d'agents de ISAME.

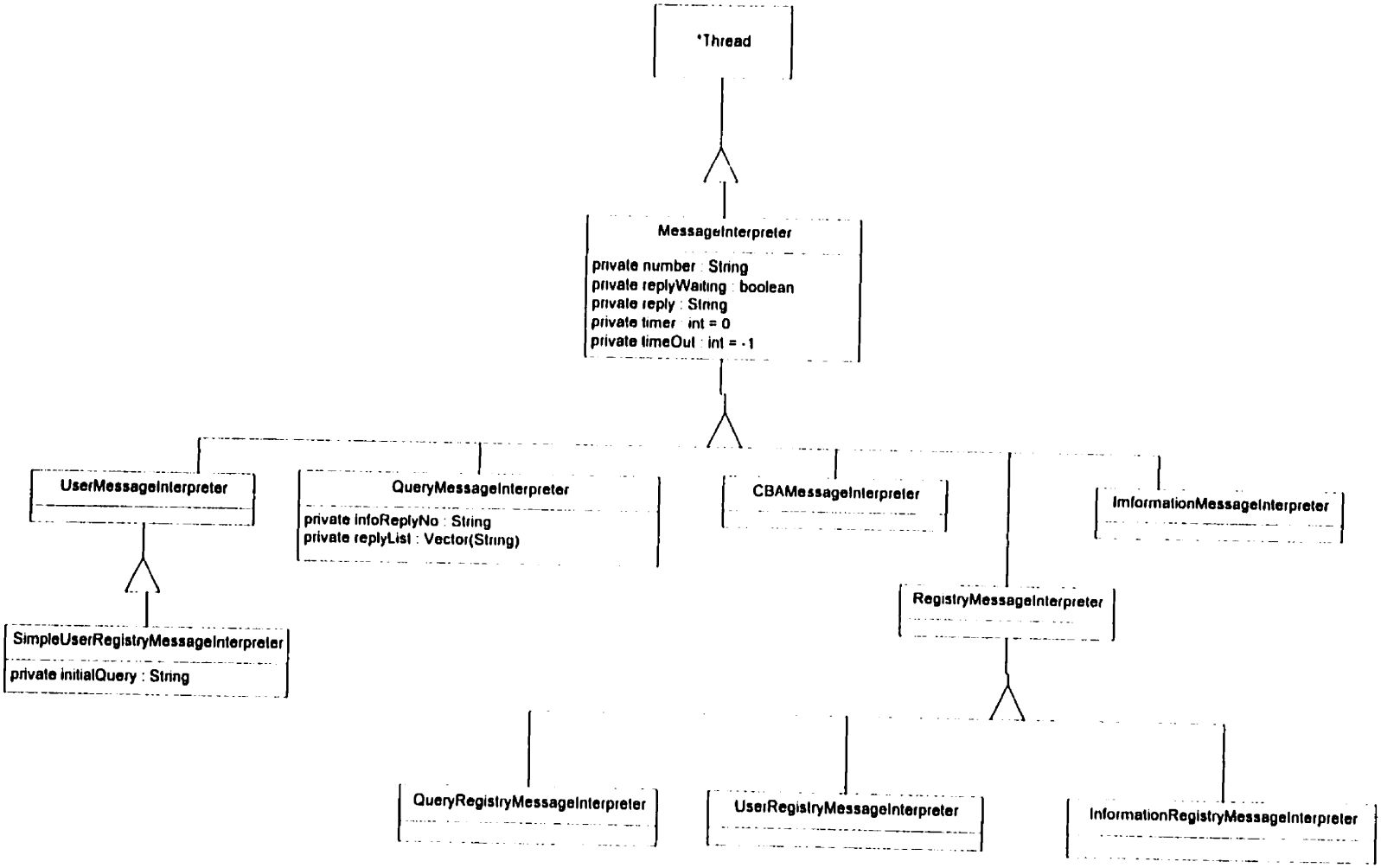


Figure F.3. Hiérarchie des classes d'interpréteurs de messages dans ISAME.

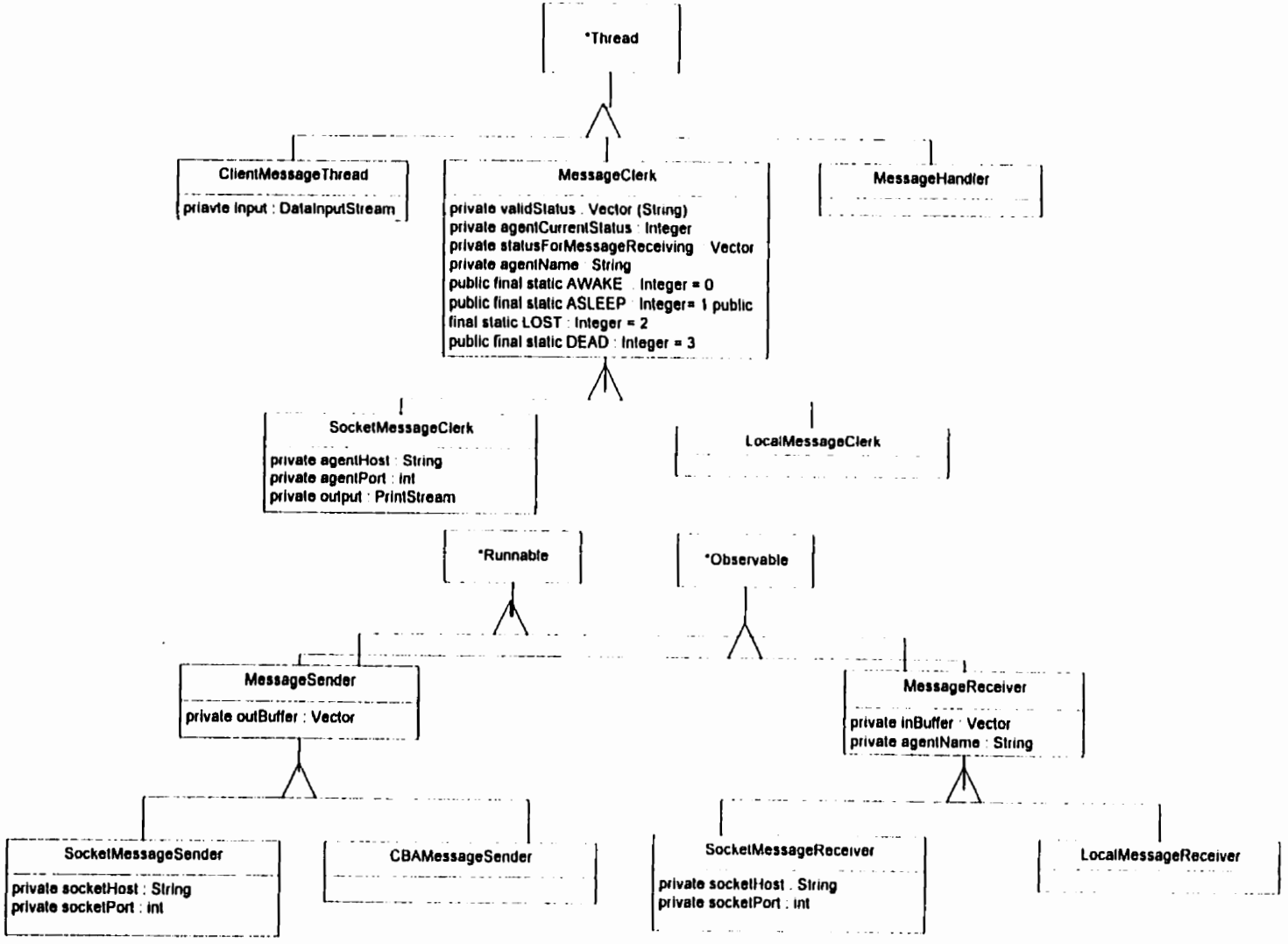


Figure F.4. Hiérarchie des classes pour l'échange de messages dans ISAME.

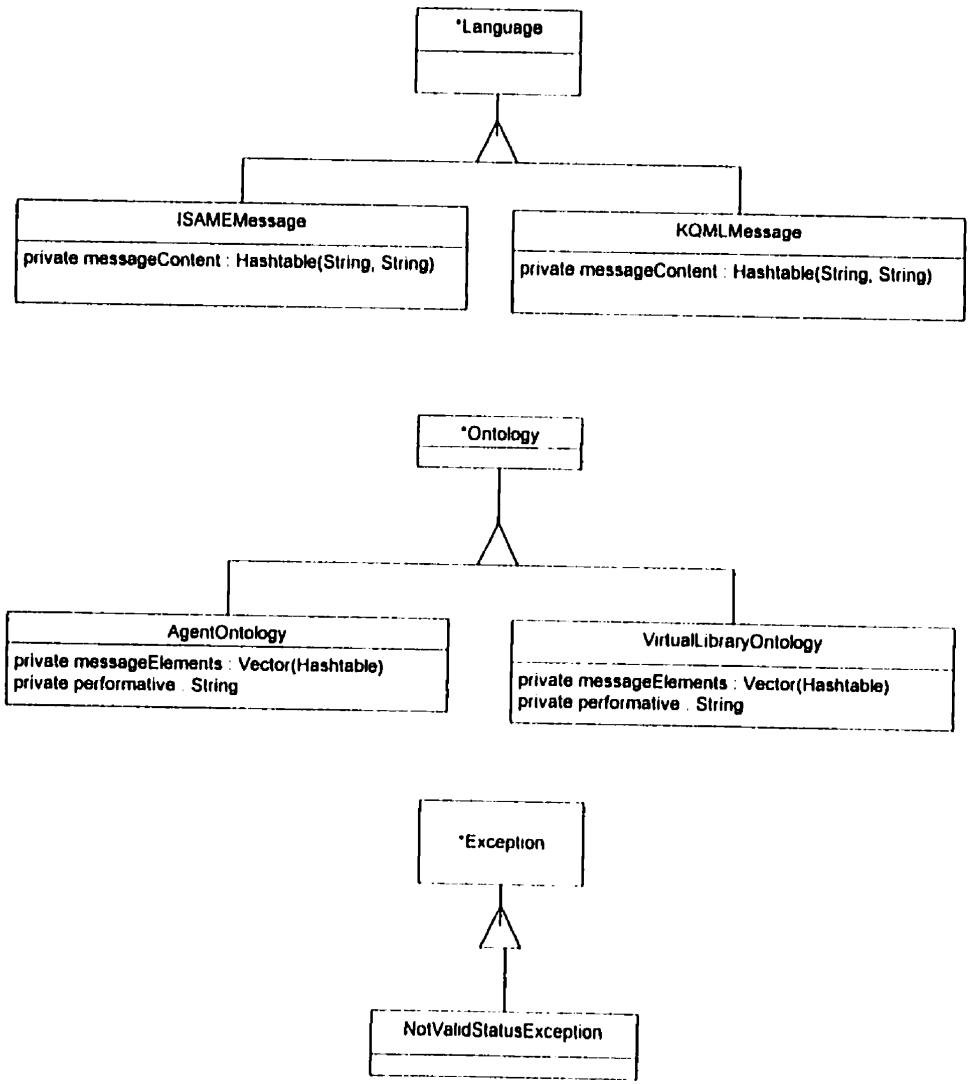


Figure F.5. Hiérarchie des classes de langages, d'ontologies et d'exceptions dans ISAME.

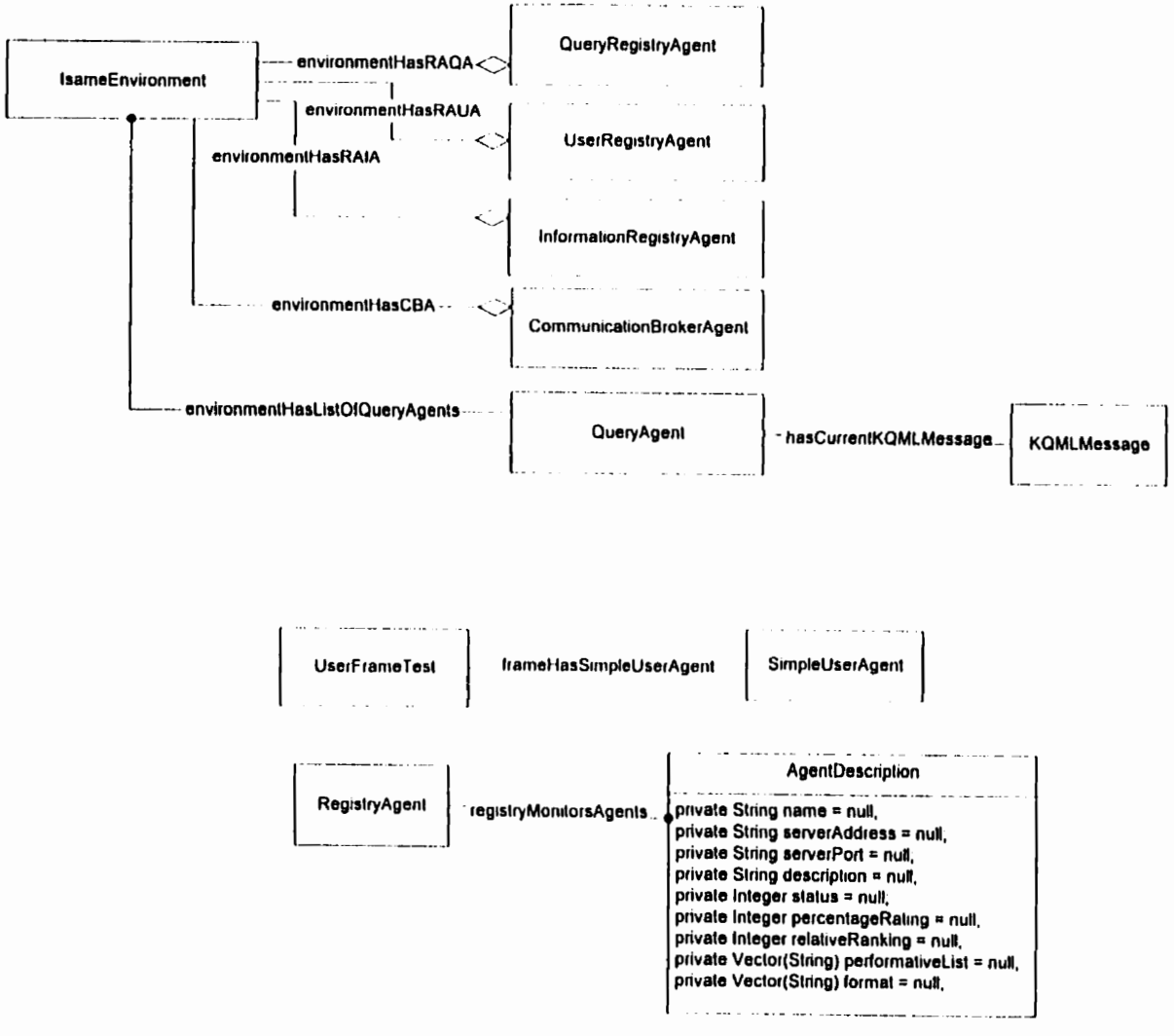


Figure F.6. Principales associations impliquant les agents de ISAME.

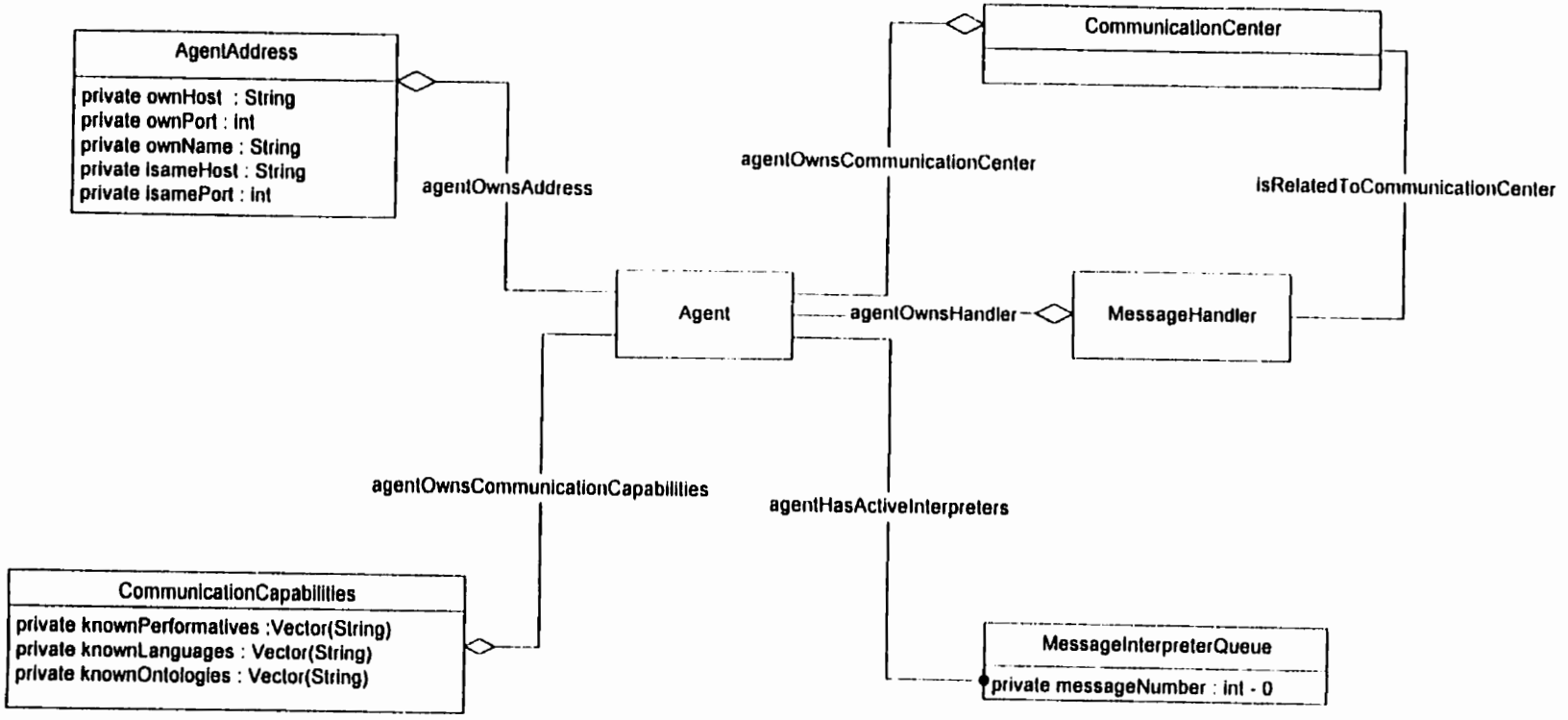


Figure F.7. Principales associations impliquant les agents de ISAME (suite).

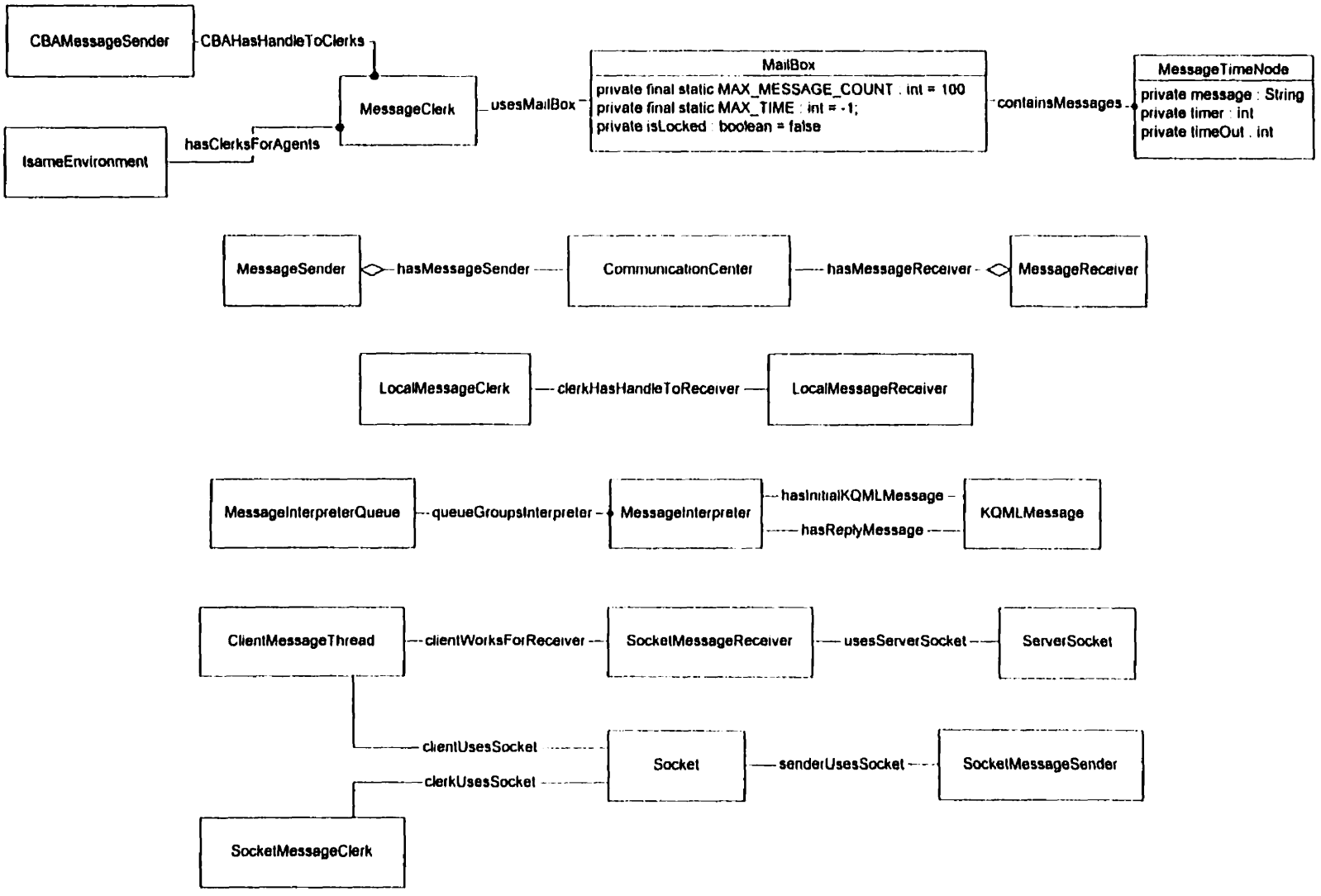
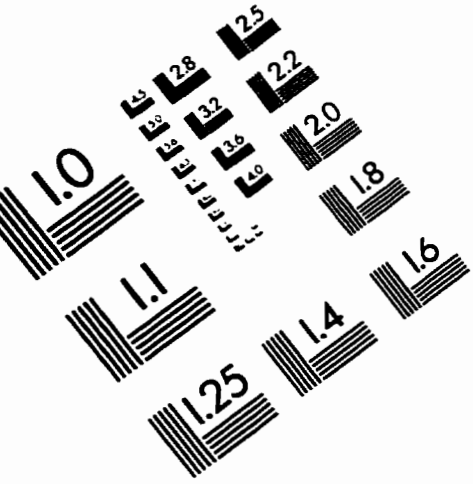
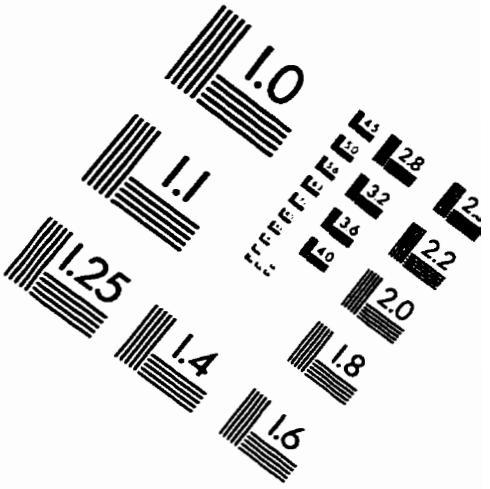
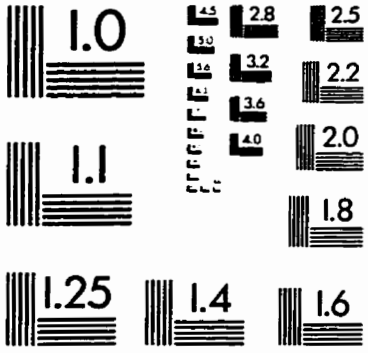


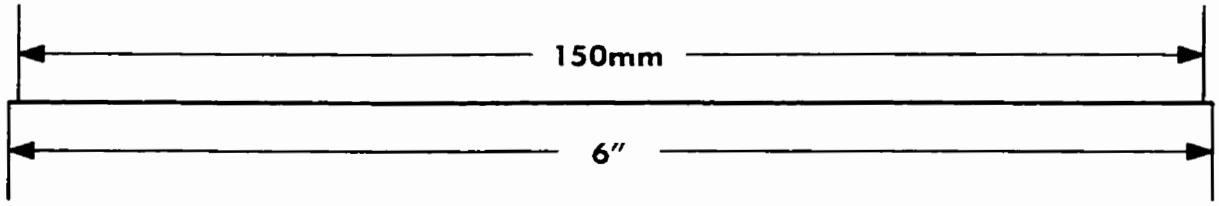
Figure F.8. Principales associations impliquant les classes de communication de ISAME.

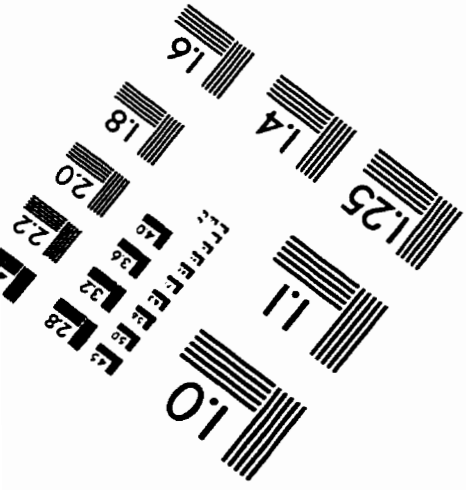
IMAGE EVALUATION
TEST TARGET (QA-3)

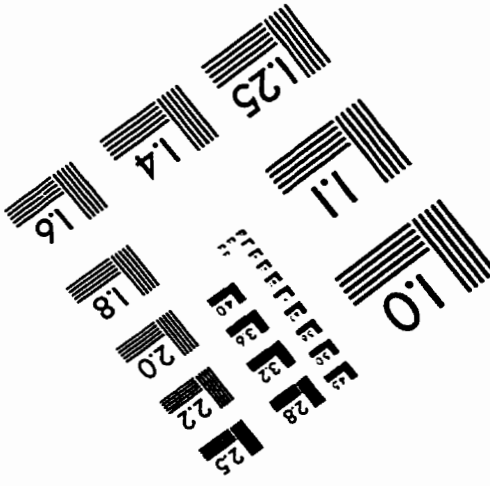












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989