Titre: Title:	Modeling a multi-agent system for retrieving information from distributed sources
Auteurs: Authors:	Sophie-Julie Pelletier, Samuel Pierre, & Hai Hoc Hoang
Date:	2003
Type:	Article de revue / Article
Référence: Citation:	Pelletier, S.-J., Pierre, S., & Hoang, H. H. (2003). Modeling a multi-agent system for retrieving information from distributed sources. Journal of Computing and Information Technology, 11(1), 15-39. https://doi.org/10.2498/cit.2003.01.02

Titre de la revue: Journal Title:	Journal of Computing and Information Technology (vol. 11, no. 1)
Maison d'édition: Publisher:	University of Zagreb Faculty of Electrical Engineering and Computing
URL officiel: Official URL:	https://doi.org/10.2498/cit.2003.01.02
Mention légale: Legal notice:	

Titre: Title:	Modeling a multi-agent system for retrieving information from distributed sources
Auteurs: Authors:	Sophie-Julie Pelletier, Samuel Pierre et Hai Hoc Hoang
Date:	2003
Type:	Article de revue / Journal article
Référence: Citation:	Pelletier, S.-J., Pierre, S. & Hoang, H. H. (2003). Modeling a multi-agent system for retrieving information from distributed sources. *Journal of Computing and Information Technology*, *11*(1), p. 15-39. doi:10.2498/cit.2003.01.02

Modeling a Multi-Agent System for Retrieving Information from Distributed Sources

Sophie-Julie Pelletier*, Samuel Pierre** and Hai Hoc Hoang**

*Softimage, Montréal, Canada
**Department of Computer Engineering, École Polytechnique de Montréal, Canada

This paper presents a multi-agent system called ISAME and designed for intelligent information retrieval from heterogeneous distributed sources. ISAME constitutes a virtual library that supplies a set of software agents with a simplified access to a set of dynamic information sources available under electronic formats, as well as services for facilitating and optimizing information retrieval. The system also uses TCP/IP communication protocols, and proposes a series of garbage-collection mechanisms to avoid the preservation and propagation of information among agents, or messages that become inaccessible or outdated as well as the use of the resources that become undesirable. The originality of this system rests on the fact that it simplifies the information retrieval from distributed heterogeneous sources by making these sources transparent to the user.

Keywords: information retrieval, distributed sources, heterogeneous networks, multi-agent architecture, virtual library.

1. Introduction

Access to information is efficient when one knows where to find the information he or she is looking for. However, this task becomes complex when the information source is unknown. The complexity considerably increases when one does not know the type of information useful for his or her purpose and has only limited knowledge of the information sources put at his or her disposal without mentioning the fact that the arrangement and use of these sources can considerably vary from one to another. Thus, it becomes necessary either to ease the task of information retrieval, or to provide the users with some automation mechanisms.

In the elaboration of search engines, two main goals are generally pursued. The first goal is the provision to users of a vast array of information sources. The second goal is to supply these users with services usually provided by a competent librarian. The usefulness of such tools lies in their capacity to adequately meet each user's particular needs.

Over the years, information retrieval processes have witnessed considerable changes. Information sources are diversified and indexed on numerical supports; most often, their contents are only accessible in numerical formats. Now, it is possible to access large information sources, either on CD-ROMS (dictionaries, abstracts of journal papers, proceedings of conferences, encyclopedia,. . .) or on institutional networks (data from enterprises, university libraries) or through the Internet (web sites, ftp sites,. . .). Moreover, distinction among electronic storage formats and different types of information media displays have practically disappeared. Thus, thanks to multimedia application software, texts, photographs, and videos can be displayed on desktop. In effect, one can speak of democratization of information.

However, although such democratization has many advantages, it also has many complexities. For example, the number of information sources is ceaselessly growing, and each source has its search engines, and supplies results in distinct formats. Thus, during a research process, the user faces numerous questions such as:

- What are the most relevant sources of information at the beginning of the research?
- How to express the research theme for a particular information source? How to refine research?
- What are the dependability and quality of the information accessed from a given source?
- If two sources give contradictory information, how to identify the most credible source?
- At what pace does the information content change?

These questions reveal an increased transfer of technical burdens from information professional to users. Therefore, in order to facilitate and render information retrieval more efficiently, it is imperative to equip users with new tools. Given the fact that a large number of information sources are now available on electronic supports, computers are increasingly used for information retrieval. Further, the information found is usually handled in a computerized environment. Consequently, it is necessary for the tools that are available on the users' desk to be integrated to both the user's work environment and numerical sources of information. To be efficient, such tools must be able to assess the context of an information request. The questions to be answered are the following: Who is the user? For what purpose is the research being undertaken? What are the relevance and quality of the retrieved information [10], [11] ?

This paper proposes a multi-agent architecture for information retrieval from distributed heterogeneous sources. This architecture leads to a client/server system using TCP/IP as communication protocol, *Windows 95* as operating system, and *Java* as programming language. Section 2 characterizes the main challenges related to the issue and analyses some relevant works related to the problem treated. Section 3 describes the proposed multi-agent architecture for information retrieval, while Section 4 presents the implementation details of this architecture. Section 5 analyses implementation results.

2. Challenges and Related Work

The creation of either a tool or an intelligent information retrieval assistant brings about huge problems related to the notion of intelligence and the associated concept of the assistant. Humans succeed in accomplishing many tasks, solely with the help of their cognitive system. It appears that there are no known artificial intelligence techniques that synthesize this cognitive system. Moreover, there is no universal or optimal research tool for all these information sources. In this section, in order to approach the problems of communications among agents, we elaborate some specifications of an information retrieval assistant and state the basic principles on which the multi-agent environment lies.

2.1. Description of a Research Assistant

Based on heuristic analysis of tasks traditionally achieved by human research assistants such as librarians, a list of characteristics usually met by an "ideal" research tool was established [5], [14]. In effect, a librarian or a research assistant specialized in information retrieval has, among other qualifications, the following aptitudes:

Knowledge of information sources: A human agent knows about the information sources to which he or she has access, information format, and update frequencies. She or he ceaselessly explores the new possibilities offered by these sources.

Knowledge of the client: A human research assistant also knows his or her client's (or user's) profile, i.e. the preferences and needs of the latter. This profile is updated according to the interactions between the research assistant and the information client.

Adequate presentation of results: A research assistant knows how to present results of the research according to practical format for the client (selection of the most relevant documents only, bibliographical index only, abstracts only, and order of relevance). This format can vary depending of the request context. The quantity of presented results is also being adjusted according to this context.

Suggestion for research refinement: A research assistant is able to help the information

client define the information request through successive refinement mechanisms. Explanation of research process: A research assistant is capable of providing the client with the explanations needed to complete the process of information retrieval. In doing so, the research assistant trains his or her client to be more efficient in retrieving information, and thus avoids making the client totally dependent on the assistance services.

Autonomy and pro-activity: The research assistant is on the look out for new information deemed interesting for the client. The research assistant accumulates this information, and timely presents it to the client.

Within the framework of computerized implementation, the execution of each of the identified assistant aptitudes constitutes a complex problem. Given the rapid changes in the realm of automated information retrieval, it is inefficient to integrate all these tools into a unique computer entity. Since integrated tools are increasingly complex to model and maintain, it is necessary to devise a modular approach to this problem.

Diversity and heterogeneity of information sources, as well as of associated research techniques, constitute only one aspect of the knowledge necessary for an intelligent assistant. Other, totally distinct tasks, but complementary to the searching of information resources, also require specific implementation techniques. Thus, establishing a user's profile requires precise and distinct knowledge of exploring information sources. Similarly, filtering research results which adequately fulfill users' needs, is a different task from providing an explanation of the process used for retrieving information. This brief analysis leads to the following assessment: the intelligence of a given assistant cannot be designed in a monolithic fashion [2], [12].

2.2. Basic Principles of the Multi-Agent Architecture

A more modular architecture that allows for the distribution of the assistant's intelligence seems more appropriate to the context described previously. Such a modular architecture requires that each module — part of the assistant — intelligently achieves the task to which it has been dedicated [8], [19]. In order to orient these modules' individual activities towards the expected global behavior, the system requires that they are equipped with collaboration and communications mechanisms [18]. This is a matter of defining a multi-agent architecture allowing for the creation of an assistant according to the following criteria:

- independence of the intelligent assistant in relation to the modes of implementation of used information sources;
- independence of the assistant in relation to users who want to call upon these information sources and in relation to the types of request related to these sources;
- independence of the same assistant towards the availability of known sources.

The whole idea is to allow for the integration of tools already available, as well as to add others as they become available. For the assistant to play his or her role, the following items should be considered: the user's profile, intelligent techniques for selecting information sources and the ability to explain the reasoning used to accomplish the research. It should be noted that the user is defined as either a human or a software entity. For the sake of simplicity, hereafter, we use the use the pronoun "it" instead of "she" or "he".

The proposed architecture is called ISAME; it is largely inspired by architectural notions defined by Lander and Lesser [13] and UMDL project group [5]. ISAME uses the KQML language developed by Finin et al. [6]. Agents can be seen as black boxes. They incorporate knowledge, languages and mechanisms used in the exchange with agents external to the ISAME architecture.

2.3. Communication Among Agents

In order to collaborate, heterogeneous agents must access the techniques through which they can efficiently communicate [7]. For such communications to be efficient, the meaning of a message must be the same for all agents. Therefore, a language common to all agents must be developed. Implementation of this language should not be an integral part of any agent. The language is simply a communication protocol, or, in other words, an interface among the agents

in interactions. It enables agents to base their messages on common syntax and semantic.

Even though communication in multi-agent systems could be carried out in a procedural fashion, that is through an exchange of autonomous executables, the majority of research in inter-agents communication is done on the communication using declarative messages. In such a case, the agent emits messages whose syntax and semantics are precise and often based on two components: a label indicating the type of message and the content, that is, the message itself.

Communication languages used by autonomous agents are called inter-agents communication languages. At the world stage, the KQML (Knowledge Query and Manipulation Language) [4], [6], [15] is among the languages that are undergoing development. KQML is being increasingly adopted as a standard, and has been already used throughout the world in a dozen of multi-agent projects, including the two large-scale ones: ARCHON in England [9] and UMDL in the United States [5].

KQML allows for the exchange of messages because its syntax, semantic and message envelope are standardized. But, KQML does not impose restrictions on the way of expressing the content of information conveyed by the message. One of KQML's disadvantages lies in the fact that it is not sufficiently appropriated for the expression of complex messages such as request for databases and mathematical expressions to be evaluated by the message receiver. The KQML research team has proposed a more appropriate language in order to express the value of the parameter *content* of a message, the KIF (Knowledge Interchange Format) language [7]. KIF is based on the syntax similar to LISP language.

Whatever syntax is chosen, the words used to express knowledge (message content) and the values must be derived from a vocabulary common to the agents called "ontology". Such ontology provides the agents with a frame of reference allowing them to share and interpret information. A type of ontology can be developed for each of the knowledge realms within a group of agents involved in generic information retrieval, stock markets, search for whether forecast, electronic commerce, etc. In short, two agents exchanging KQML messages, must both know the language and ontology allowing for the expression of the values related to the parameter *content* of exchanged performative statements.

3. Multi-Agent Information Research Architecture

In this paper, the proposed architecture aims to establish a support environment for collaboration and coordination among autonomous specialized agents. Such an environment is called ISAME. It must be persistent in a way that agents can, if needed, display service requests or results, or services that have already been provided. Persistence refers to the fact that ISAME must be in service on a continuous basis, that is, without interruption and for an indefinite period of time. In the case where ISAME is put in waiting for a given period of time, its state must be saved, then restored when it becomes active. In this way, ISAME's long-term and short-term memories (agents, messages in transit, etc.) could be preserved. The set of tasks assigned to the assistant does not fall on a single software entity. Rather, these tasks fall under the responsibility of a team of specialized agents, a set of software entities.

3.1. Basic Principles and Constraints of the Architecture

The ISAME architecture is based on the following five basic principles:

- an open architecture with minimum constraints to the agents;
- four integrated classes of agents;
- a taken-for-granted class of agents;
- communications through KQML type of declarative messages and a language expressing information content and ontology that is specific to ISAME, the ISAME-L and a *VirtualLibrary* ontology and an agent.

Although ISAME is mainly designed for information retrieval, it can be easily extended to support several other types of activities such as electronic commerce or network management. These activities can be carried out, on the condition that they are based on a multi-agent approach. For the sake of architectural openness

and for the purpose of providing the designers of agents with maximum latitude in the ISAME environment, the architecture imposes on the agents the following minimum set of requirements:

- understanding of KQML and ISAME-L languages as well as of the Agent and/or Virtual library ontology;
- publication of KQML types of performative statements;
- publication of the physical (IP) address at which the agent can be reached;
- reading and writing of messages according to mailbox mechanisms defined by ISAME in an autonomous fashion (i.e., without a request from ISAME).

If an agent respects these constraints, then it can be registered on ISAME, and thus it can enter communications with other agents. However, it is not guaranteed that an agent will find other agents to communicate with! Actually, because ISAME is only a site that facilitates the meeting among agents, it has no control over those agents that are registered on the environment. Thus, it is possible that no research agent is registered when a user-agent places an information request. However, if an agent (who is able to undertake information research) is registered at ISAME, such an agent would likely be able to respond to the request. In order to increase the chances of response to an information request, the user can define his or her request as being persistent. This request will remain valid until the user withdraws it. The results are then supplied to the user according to its own response frequency specifications: every day, several times a day, when a result is found, when a new source of information responds to the request, etc.

Component	***Symbols***	***Main Functions***
User Agent	$UA_{\mathbf{x}}$	Representing a user x by expressing its information request.
Information Agent	$IA_{\mathbf{y}}$	Representing an information source and answering users' requests.
Query Agent	$QA_{\mathbf{xn}}$	Follow-up a request n from the user x.
Support Agents	SA	Set of agents providing internal service in the ISAME environment. Categories of support agents: CBA, RFA, SFA and RA.
Communication Broker Agent	CBA	Transmiting messages into the agents.
Source Filtering Agent	SFA	Determining which subset of agents should be called to answer a request.
Result Filtering Agent	RFA	Filtering results by the $IA_{\mathbf{y}}$.
Registry Agent	RA	Keeping the track of active agents in the ISAME environment. List of registries: RA_{UA}, RA_{IA}, RA_{QA}.
UA Registry Agent	RA_{UA}	Keeping the track of $UA_{\mathbf{x}}$ type agents.
IA Registry Agent	RA_{IA}	Keeping the track of $IA_{\mathbf{y}}$ type agents.
$QA_{\mathbf{xn}}$ Registry Agent	RA_{QA}	Keeping the track of $QA_{\mathbf{xn}}$ type agents.
Communication Area	CA	Putting a mailbox at each agent's disposal.
Clerks	—	Rooting messages related to the accumulated messages in the mailboxes.
Mailboxes	—	Accumulating messages addressed to one agent.

Table 1. ISAME's Key Elements.

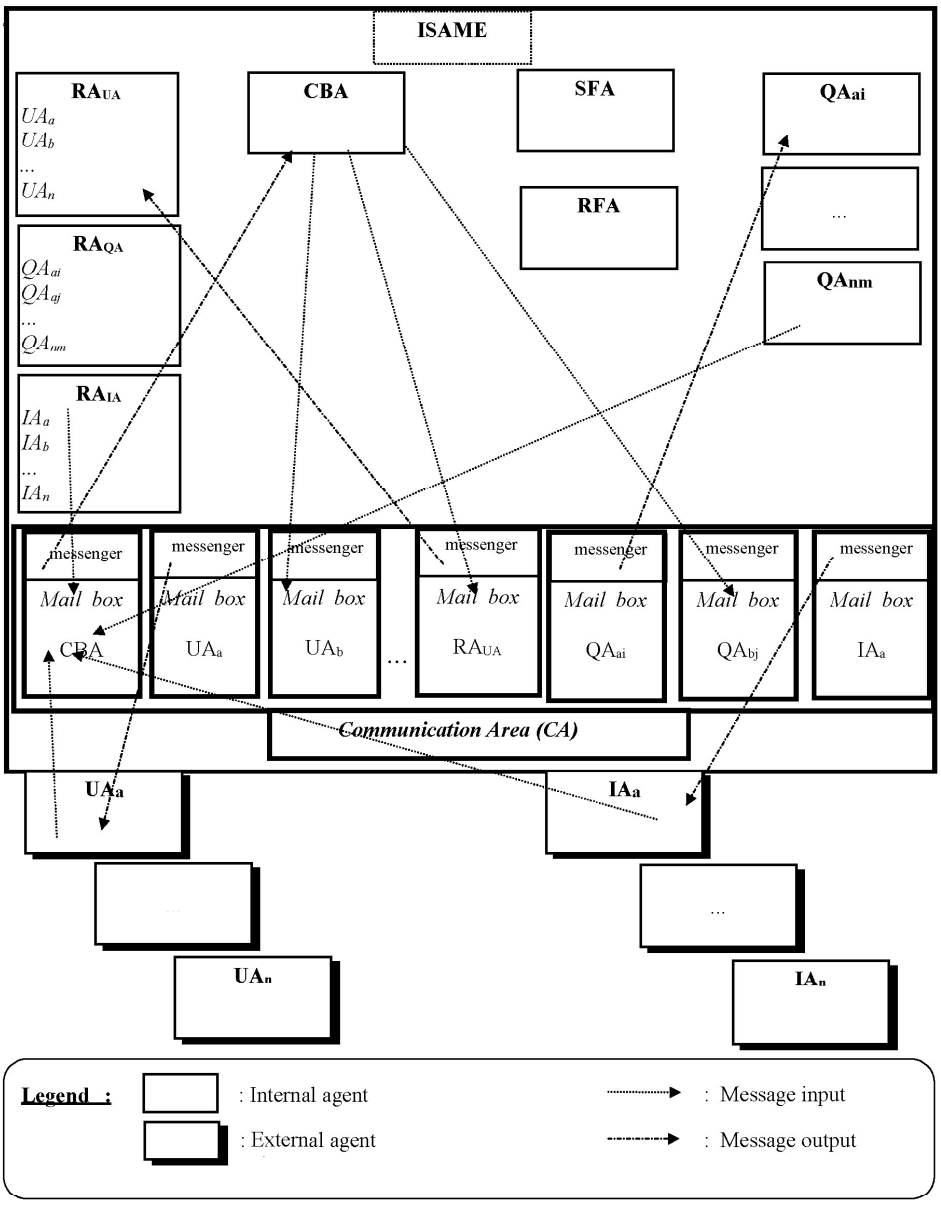

Fig. 1. ISAME Architecture.

3.2. Typologies of ISAME-Supported Agents

As illustrated in Figure 1, ISAME supports four categories of agents: *User Agents, Information Agents, Request Agents* and *Support Agents*. The types of services supplied by these categories make each of them different from the other. Table 1 synthesizes ISAME's main elements, their respective symbols and main functions.

3.2.1. User Agents (UA)

The know-how of agents of UA type is referred to by the symbol UA_x, where x identifies the user to whom the agent is being associated. The role of these agents consists of presenting the request in a multi-agent environment, which in turn calls upon a communication channel between the user and the ISAME environment. There exists one UA_x when a user is registered at ISAME.

A UA_x allows for the independence of the research assistant towards the user interface, because it serves as an intermediary between the two. Moreover, since all UA_x are agents registered at the ISAME environment, but created outside this environment, it is possible for these agents to integrate the level of sophistication required by the pre-processing of information as desired within the context of use of information. Finally, the UA_x themselves carry out the presentation of results and support the users in handling the results.

3.2.2. Information Agents (IA)

In our model, the IAs are referred to by symbol IA_y, where y identifies the information source presented by the agent. All IAs establish a link between the information sources and the ISAME environment. In order to receive requests, each information source must register its IA_y at the ISAME environment.

Like the UA_x, the IA_y are registered at, but created outside the ISAME. They supply the required homogeneous interface and heterogeneous sources of information. Indeed, the IA_y have the required knowledge on the request tools to be used on presented information sources. Information sources become known to ISAME through the registration of their IA_y. For their part, the IA_y secure the independence of ISAME towards the heterogeneity of sources, as well as towards their availability.

3.2.3. Query Agents (QA)

In our model, QA are ISAME's internal agents which look over the interests of users. They are referred to with the symbol QA_{xn} where x represents the user agent which formulated the request and n the specific number of this request. QA_{xn} are created from either punctual or persistent requests coming from users. A request is punctual if it has been processed only once. Such a request disappears from the environment once results have been presented to the user. A persistent request is the one that remains in the environment until a user withdraws it. Such a request is regularly processed once a day, when a new information agent becomes available. In this context, the user is informed of any new result found.

QA_{xn} carry out the requests and hand-over the results to UA_x. QA_{xn} communicate with the *Source Filtering Agent* (SFA) in order to determine the set of IA_y to which requests must be forwarded. In short, the QA_{xn} forward requests to the IA_y, accumulate results, and communicate with the RFA in order to filter and format the results; then, they hand over the result to the requesting UA_x.

3.2.4. Support Agents (SA)

The SAs facilitate coordination and communication among the three categories of agents. The following are SA examples: Communication Broker Agent or **CBA**, Registry Agents or **RA**, Source Filtering Agent or **SFA**, Result Filtering Agent or **RFA**.

Communication Broker Agent (CBA)

This agent ensures the routing of messages among active agents in the ISAME environment. Every message is sent to the CBA which in turn forwards it to the appropriate address(es). Thanks to the CBA, agents interacting in the ISAME environment do not have to know each other, because the CBA identifies potentially interested addressees in the sent message. However, an agent can always explicitly mention the message addressee. In such a case, the CBA sends the message only to the designated addressee. The main purpose of the CBA is to avoid that all agents have to register with each other as well as to retain the information on other available agents.

Registry Agents (RA)

Creation of these agents is inspired by UMDL architecture [5]. In the ISAME environment, the function of a RA is to keep the track of active agents. There is one RA for each of the following categories : UA_x, IA_y and QA_{xn}. These are registers allowing the CBA to easily identify the agent potentially interested in a given message. Moreover, when a registered agent desires to modify the published information about itself, it makes required changes only by notifying its particular registry agent; then, the changes will be noticed in the entire environment. Similarly, when an agent desires to withdraw from the environment, it only has to notify the registry agent, and the latter erases any trace of its passage in the environment, given their knowledge

of UA_x, QA_{xn} and Ia_y. This information includes the agent's profile, messages that were sent to or by an agent, or messages that are being processed by other agents.

Source Filtering Agent (SFA)

From a user's profile and statistics on IA_y known in ISAME, the SFA suggests a sub-set of IA_y for a given request.

Result Filtering Agent (RFA)

From the result picked up by a QA_{xn} and the user's profile supplied by the UA_x, the RFA undertakes the first filtering of results before results are sent to the UA_x. Two of these classes of agents, the UA_x and IA_y, are of external type. They are agents defined outside the ISAME environment, but reside on the clients' machines and have the communication interface required for the interactions with other ISAME-registered agents. QA_{xn} and SA are also called agents of internal type, because they are created within ISAME and hosted in the ISAME environment's server. QA_{xn} and SA remain on the server until they are considered useless.

3.3. Autonomy of Agents

Agents work in parallel and in an autonomous fashion. Each of them intervenes when it is needed according to its appropriate characteristics, the properties it had given itself and those known in the environment. This work consists of two main tasks: sending out mails and answering the incoming ones, as well as putting know-how for the benefit of maintenance services.

3.3.1. Mail Management

Each agent has the task of scrutinizing and answering the mail that comes in its mailbox. In effect, each agent is provided with a mailbox in the ISAME's communication area (CA). A mailbox is a data structure located on the environment's server where the CBA is found. The CBA deposes all the messages that should be delivered to agents. ISAME puts an office clerk or messenger at the disposal of each agent. The messenger directly and regularly delivers mail to the agent when the latter is awake, that is, capable of carrying out required activities. The office clerk works continuously and in an autonomous fashion. It resolves the agent's address to which it is attached, by calling upon a TCP/IP type of connection or another connection, if it is deemed necessary.

Different scenarios are possible if the agent is not active, that is, when it is dormant and therefore cannot receive the messages sent by the office clerk. These scenarios mainly depend on the type of agent. If an IA_y is not responding to a request for an information sent to it, the request will be destroyed. However, if it is a matter of answering a request for an information which is addressed to a UA_x, the message will be preserved and some other time the messenger will try to contact the UA_x. The agent controls the frequency of sent messages, as well as the priorities according to which these messages will be sorted out. It notifies its preferences to one of the registries assigned to the following classes of agents: UA, IA or QA. It also informs the messenger who has the mechanisms ensuring the independence of the architecture towards the availability of agents, and prevent the agent from uselessly monitoring its mailbox(es).

When an agent desires to send a message to another agent, it directly sends it to the CBA that delivers it to the appropriate agent(s)'s mailbox(es). In order to standardize communication with the CBA, every message sent to it must go through the unique address published in the ISAME environment. Such a standard is also applicable to the agents found on the server or on the CBA.

Currently, the unique address published in the ISAME environment consists of an address of TCP/IP type. All the agents registered at the ISAME environment must know the TCP/IP communications protocol. The CBA is as a unique point of entry for external and internal agents. For this purpose, the CBA's TCP/IP address must at all times remain public.

When a message is received, it is possible that the CBA cannot identify the message receiver or interpret the message. In both cases, the CBA must notify the sender the refusal to process the request. One of the following could be the rationale of this refusal: incapacity of understanding the request, error in the formulation of the request, expiry of the delay, lack of resources to process the request, etc.

3.3.2. Collaboration and Achievement of Maintenance Tasks

Each of the ISAME's agents must autonomously make its knowledge available for the benefit of other agents by responding to their requests. It must also achieve maintenance tasks required by the community of agents. Thus, the UA_x must be registered in the environment. The IA_y must also be registered in the environment, and continuously update their knowledge on each source of information they represent. The QA_{xn} must ensure that conditions for ending the search for information are not reached (all IA_y have supplied their results or the delay allocated for the request is reached). However, if the requests are persistent, a new request must be launched at a certain interval. Registries must be sure that the agents registered with them remain accessible. They must also eliminate those agents which are no longer registered with them.

Success of the environment depends on the execution of these two categories of tasks by all agents. The exchange of mails allows agents to collaborate and enables one or several users to simultaneously process several tasks.

3.4. Communications Space (CA)

Activities realized in the ISAME environment are all directed through the exchange of mails among agents. No central control is exercised. Rather, agents exchange services and information according to their needs and good will. In order to support the agents in this exchange of messages, the environment puts at disposal of each agent one mailbox through which each receives messages. A dedicated messenger is linked to each agent and directly delivers messages to the agent's communication manager. One or several entities (objects or others) are dedicated to the activities of receiving or sending messages on the agent's residence machine. This delivery system allows for better time management to each agent, because agents do not have to constantly monitor their mailbox in order to verify arrival of messages. Moreover, the clerk carries out some complementary tasks in order to prevent the agent from processing outdated messages.

As illustrated in Figure 2, all messages are distributed among the addressees' mailboxes through the CBA. If the addressees are explicitly indicated, the message will be delivered to corresponding mailboxes. There is, however, an exception to this rule: if an agent is no longer part of the environment and, consequently, its mailbox has been destroyed, the CBA sends back a message of "sorry" type to the sender to notify that the message could not be delivered. If the addressee is not explicitly mentioned, the CBA determines the list of the agents who could be interested in the message. In fact, the use of performative statement lists that are known by each agent does the work of determination. Regarding the maintenance agents, this list is

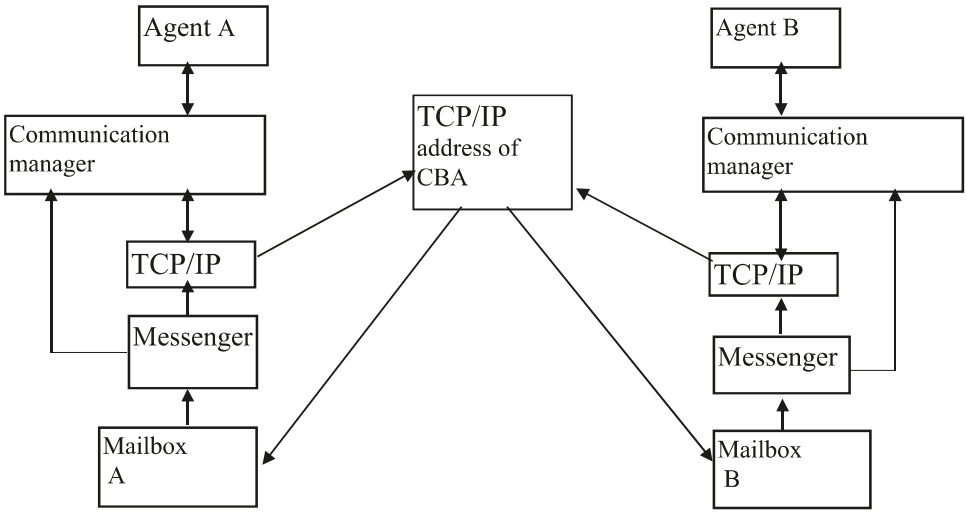

Fig. 2. Communications among agents through managers and messengers.

known by the CBA or available with RA agents for the benefit of UA_x, IA_y and QA_{xn} agents. In all cases, agents must communicate with the CBA in order to interact with each other. In the ISAME environment, the CBA's address is the only necessary public address. The agents which desire to register in the environment, send messages to other agents, or decline to use their addresses.

For these exchanges of messages to be carried without problems, a control mechanism must be put in place to ensure the synchronization of activities in the mailboxes. The messenger undertakes this task of synchronization, because it is the only participant that has direct access to the mailboxes. The CBA records the messages in the mailboxes through the messenger which withdraws them from the mailbox, and delivers them to the agent. In its cleaning function, the messenger is also responsible for deleting messages. The synchronization is undertaken through semaphore signals used by the messenger.

Finally, since ISAME is a distributed environment, it is the role of the entities that handle messages for the agents to solve the problems inherent to messages distribution. Thus, it is necessary to use a communication protocol of TCP/IP type. Both, the messenger-supported protocols and the protocol chosen by the agent's

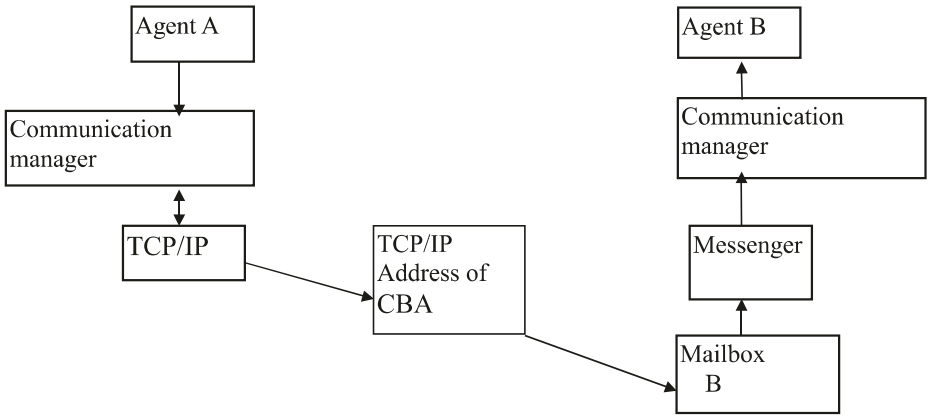

Fig. 3. Message exchange through an inter-process communication between two agents.

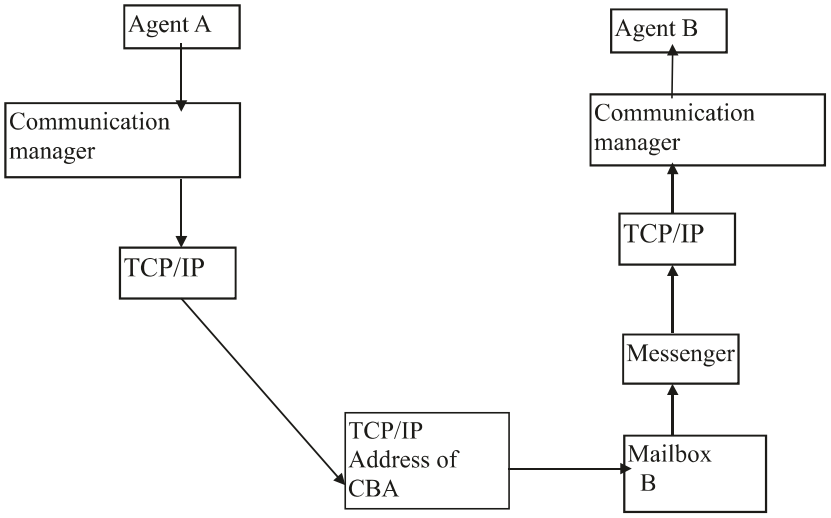

Fig. 4. Message exchange through TCP/IP.

communication manager, determine the communication protocol used by the agents.

Figure 3 illustrates an inter-process exchange of messages between two agents, A and B, available on the ISAME environment's server. The TCP/IP protocols are used independently from both the communications manager's physical location and the clerk. This is because the exchange of messages among objects is possible only if the agent and, therefore, its communication manager are in the ISAME environment's server, a place where the messenger is also found (Figure 4).

3.5. Communication among Agents in ISAME

Communication among agents is established through the exchange of messages transiting via the CBA and the communication space, before reaching an addressee. Messages' syntax and semantics are based on performative statements which follow the standards established by KQML. The language used for the content parameter is *ISAME-L*. Two form of ontology are also being defined: the Agent ontology for the share of information relative to the *Agents* and the *VirtualLibrary* ontology that expresses the request for information retrieval.

The process of routing messages via the CBA, the mailboxes and the messenger form a hybrid model for blackboard techniques and communications through messages [17], [20], [21]. This hybrid model centralizes services and processes common to all agents, but leaves the filtering and processing of data to the agents. The services and processes consist of filtering and distributing the messages. Among the advantages of the blackboard techniques, ISAME retains a certain level of communication standardization and the centralization of information services. This is for allowing the agents to be independent from each other, and for avoiding the occurrence of bottlenecks.

1. Communication Standardization

Although the CBA does not control the content of information exchanged among agents; it, however, ensures that all messages conform with the KQML language. In effect, if a message does not conform with the KQML language syntax, or its addressee cannot be found in the environment, the CBA deletes it. In the case where the CBA is able to identify the message sender, it sends to him an "error" message explaining why the message could not be correctly routed.

2. Centralized Service of Information Distribution

Before reaching its addressees, every message exchanged within the ISAME environment must transit via the CBA. The advantage of this is that agents do not have to know each of their correspondents or their precise addresses. The other advantage is the support of the message destined to a known addressee and the diffusion of messages of public interest. However, the control of messages exchanged among agents is not centralized through the CBA. Furthermore, in order to avoid bottlenecks, the agents are not provided with a public space. Rather, a private data space is put at the disposal of each agent in the mailbox. This space does not include the knowledge incorporated in the agent. It only includes messages destined to an agent. Moreover, the agent does not have to read the messages inscribed in its mailboxes, because these messages are delivered in the agent's own environment. Thus, the mailboxes serve to free the CBA from sending messages to the addressees. At the same time, like in communications through blackboards, these mailboxes allow the agents to exchange messages in an asynchronous fashion, that is, sending messages even though the addressee is not ready to receive them. The mailbox keeps the messages until the agent is ready to receive them.

Centralized reception of messages via the CBA can cause a bottleneck. But, this bottleneck can be easily avoided since several processes of receiving messages can simultaneously be put in place for the benefit of CBA-published addresses. Bottleneck can be avoided by providing the CBA with the capacity of routing several messages. Routing of these messages should be parallel, through the use of autonomous thread processes. As a result, it is impossible for an agent to block other agents' activities.

Communication through messages allows ISAME to preserve information in a message. Such communication also allows the agents to exercise certain control over information exchanges, while they are benefiting from centralized services for better synchronization of their

activities. However, communication through messages has the disadvantage of increasing the traffic on telecommunication networks, even though this type of communication allows the agents greater freedom.

Among the biggest risks inherent to this type of architecture is the fact that several elements of this environment may be inaccessible, outdated, or remain unused. This entails a waste of resources that otherwise could be better used. Therefore, it is necessary to establish cleaning mechanisms to verify that all agents continue to exist, free unused mailboxes, eliminate outdated mails in each mailbox, acknowledge that persistent requests still have some usefulness for the client, exclude low performance $IA_{\mathbf{y}}$, and so on.

These cleaning activities fall to ISAME's internal agents, mainly the registries, the CBA and the messenger monitoring the $IA_{\mathbf{y}}$, $UA_{\mathbf{x}}$ and $QA_{\mathbf{xn}}$ from the moment of their registration in the ISAME environment until their voluntary or forced withdrawal from this environment. Internal agents also manage logistics related to the mailboxes, because they (excluding the communications clerk) are persistent and their life span equals that of the environment. The clerks, for their part, inspite of being temporary (their life span corresponds to the life span of their agents in the environment), they possess critical information regarding the work of both the registries and the CBA. This is because, regularly, messengers communicate with their agents either through TCP/IP or locally. They know whether or not the agents are available. If an agent was declared inaccessible or outdated, the registry could take different courses of action: excluding the agent from the environment, destroying its mailbox and messenger by asking the CBA to order all other agents to ignore any request coming from that particular agent. If this process of denying a request concerns an agent of $UA_{\mathbf{x}}$ type, the registry responsible for the $QA_{\mathbf{xn}}$ will eliminate all $QA_{\mathbf{xn}}$ related to the excluded $UA_{\mathbf{x}}$. In this continuous cleaning operation, the messengers have another use: direct handling of mailboxes' messages including deletion of outdated messages (example: an information request followed by canceling of the request; a request coming from an agent which is no longer registered at ISAME; an expired request).

4. Implementation Details

Architecture presented in the previous section tries to grasp the set of needs relative to an intelligent information retrieval by autonomous agents. Although it is desirable to put these elements in place in order to provide all agents with an optimal work environment, the core of this system remains the CBA support agents, RAs, communication through messages. This section specifies these elements and presents the software and hardware environments on which programming and implementation tests have been made.

4.1. Subset of Implemented Concepts

The subset of concepts chosen for implementing and testing the proposed architecture includes services of virtual libraries and handling the content mainly represented by the user and information agents. This subset has been selected in order to allow for a realistic evaluation of the architecture's flexibility. This means:

1. showing that ISAME constitutes a unique access point of entry for virtual library's services and content;
2. evaluating the extent to which user and information agents remain totally independent from each other in their implementation mode and availability;
3. showing that implementation of user agents is independent from the complexity and diversity of services provided by the ISAME environment;
4. demonstrating that information and user agents can be registered as they become available;
5. showing that the use of mailboxes and clerks allows for the improvement of messages delivery connected to permanent agents;
6. allowing for the first evaluation of the tasks distribution method among support agents, as predicted in the initial model, as well as for the quality of services provided by ISAME to user and information agents.

The concepts established in the first phase can be regrouped in four categories: *languages* and

forms of ontology, agent types, associated messages. Each of these categories will be followed by a description of selected elements.

1. Languages and Forms of Ontology

KQML, ISAME-L, AgentOntology, VirtualLibraryOntology are totally supported by ISAME's implementation.

First, we have to mention that we use different terms to present ISAME. Architecture, System, Environment, Platform and Language are all related to ISAME. Environment means that agent can communicate and cooperate with each others with certain constraints such as knowing KQML and ISAME language. ISAME language is a subset of KQML containing information expression and specific agent ontology. It is based on a list of reserved words with appropriate parameters. AgentOntology contains a reserved word to determine information on the agents and their appropriate values. This information can be represented as a sequence performative (reserved (value)). For instance,

query (agent_ID:content(<sigleton>|<set>))

VirtualLibraryOntology is used to express agent communication when queries are sent by UA. This ontology supports three kinds of performative: Error Code, Query and Results. As well as AgentOntology, VirtualLibraryOntolgy contains reserved word and associate value related to each performative. The previous representation can be used for VirtualLibraryOntology too, only the set of performatives is different.

These languages and ontology are implemented through a hierarchy of classes using the UML formalism [1], [3]. In Figure 5, each of these classes constitutes an analyzer. The KQMLMessage and ISAMEMessage language analyzers allow for the validation of message syntax exchanged among the agents and the message envelop's semantic. AgentOntology and VirtualLibrary analyzers carry out validation of ISAME message contents. Each implanted agent comprises a language and ontology. It filters each message it receives through the appropriate language and type of ontology. The message interpretation and control of the activities associated with each received message is carried out by the derived classes of *MessageInterpreter*.

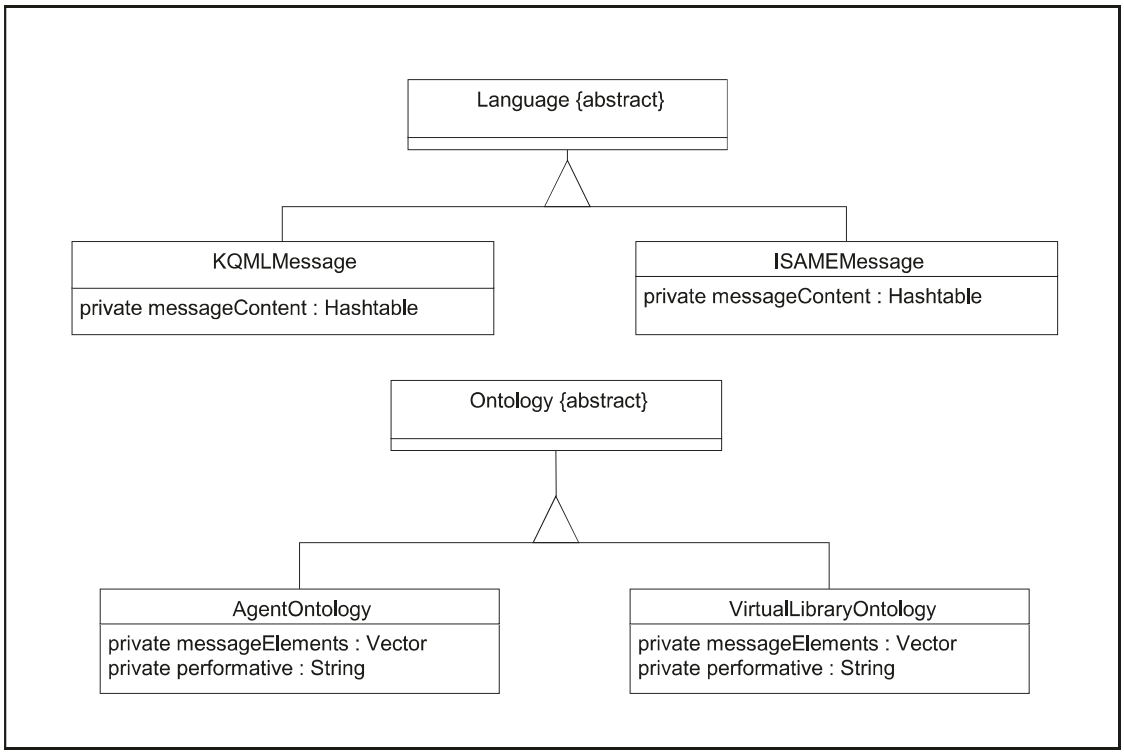

Fig. 5. Hierarchy of classes of languages and types of ontology.

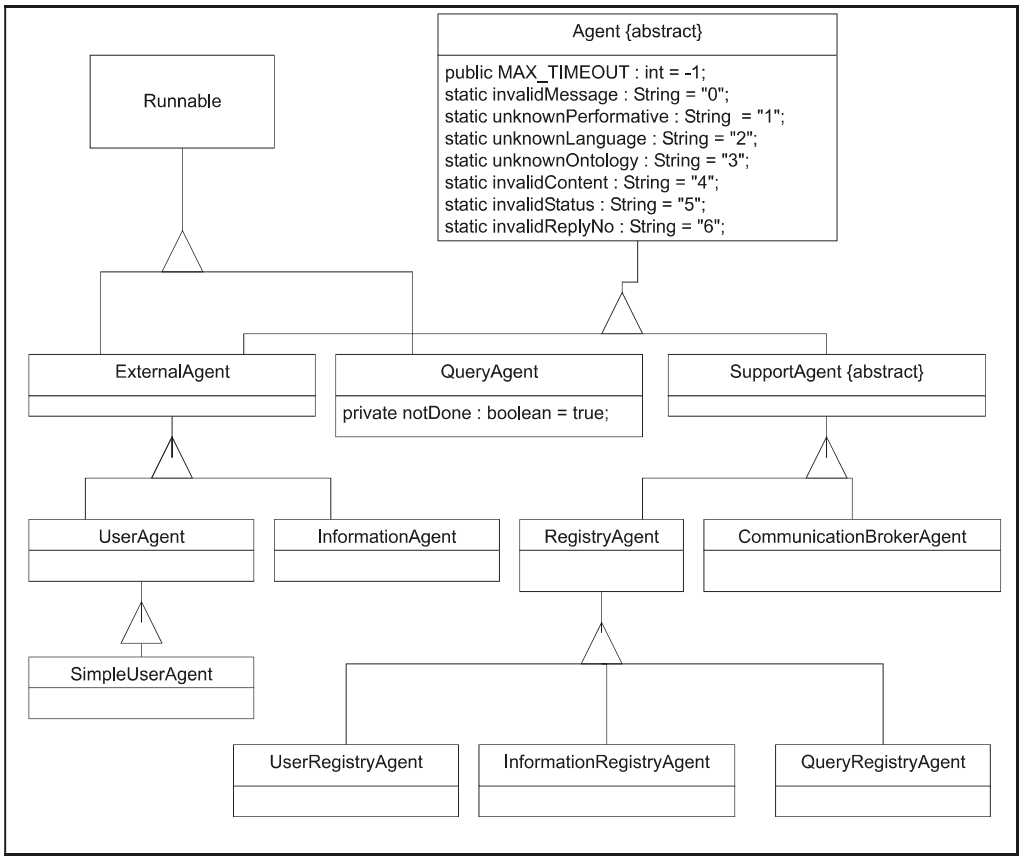

Fig. 6. Hierarchy of agent classes.

2. Agent Types and Associated Messages

The CBA is the ISAME environment's unique point of entry for all external agents, because the *socket* address published for entering into communication with the environment is the CBA's message reception address. The RAIA plays the SFA's role by providing the QA_{xn} agents with recommendation services on information sources. This recommendation is carried out at the reception of a *recommend-all* type of message. Finally, in order to achieve efficient result-filtering activities, in the absence of the RFA, the QA_{xn} is responsible for a simplified filtering.

Generic versions of users and information agents are available in the environment. These generic user agents can be derived to create agents that are specific to user needs and information sources. A simplified version of a user agent, the *SimpleUserAgent* has been derived from the *UserAgent* class in order to validate registering and withdrawing mechanisms, as well as the life cycle of QA_{xn} agents. Figure 6 presents the hierarchy of classes of agents.

The abstract class *MessageInterpreter* constitutes a thread process that carries out interpretation and execution of each message received by the agents. For each class of agents, new classes must be derived from the *MessageInterpreter*. This permits the message handling process to correspond with the context and role of each agent. These interpreters are the know-how related to each agent.

Several interpreters can be activated in parallel for a same agent. They allow the agent to simultaneously achieve several tasks. Each interpreter has its own activity agenda, managed by the sequence of activities associated to the type of processed message. At the reception of a message, the agent carries out a series of validations. It verifies the syntax, and ensures that the language, ontology and performative statements conform the context. Then, it verifies if the message corresponds to the one sent earlier

(in this case the message is routed to the interpreter that is waiting for a response), or the message is independent (in this case a new instance of interpreter will be created to handle the message). The set of interpreters of messages presented in Figure 7 undertakes the interpretation of messages through associated procedural methods (ex.: *interpretTellMessage, interpretBorkerOneMessage*). To ensure efficient management, all interpreters that are working for an agent are associated to it, through an object *MessageInterpreterQueue*.

3. Objects and Agents Distribution Among the Server and Client Machines

In the ISAME architecture, the environment's server regroups the support agents, request agents, mailboxes and messengers. User agents, information and communication managers are

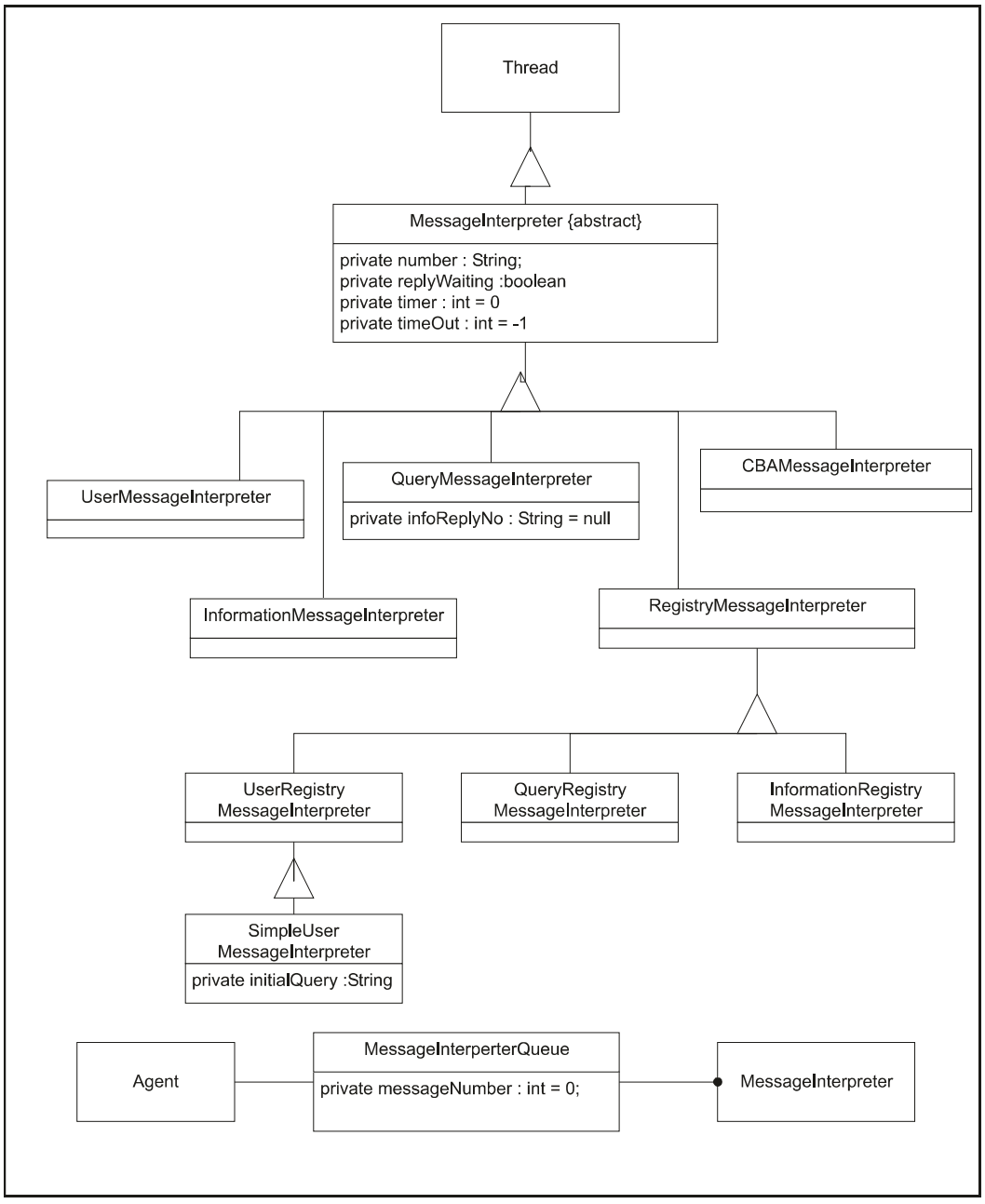

Fig. 7. Message interpreters hierarchy of classes.

located on a client machine, the machine of the agent designer. Distribution of elements among the server and the client follows a very simple rule: the ISAME environment supplies services to user agents and the information defined by institutions and individuals who want to benefit either continuously or punctually from ISAME's services.

Even though the support and request agents implemented in the framework of this experiment are all located in the same machine, communication between the messengers and communication managers is entirely based on TCP/IP. As predicted, all communication managers sent their messages to the CBA through TCP/IP. The messengers also communicate with the managers through TCP/IP.

4. Communications Mechanisms

Two groups of classes are necessary for the implementation of communication mechanisms among the agents. On the one hand, reception and sending of messages must be placed inside each agent. On the other, the mail system allows the CBA to route messages by transiting them via the ISAME environment, before they reach the addressee.

For sending and reception of messages, the *CommunicationCenter* class places the necessary tools inside each agent. This communication center is made of two objects: a receiver *MessageReceiver* captures incoming external messages, and a sender *MessageSender* sends the messages outside the environment. When a message is received, it is inserted in a queue, that is in FIFO order (First In First Out). Then, a message manager associated with the agent, that is a *MessageHandler* that withdraws it from the queue, and puts it back for processing. When a message from a *MessageHandler* comes into the agent's location, the agent carries

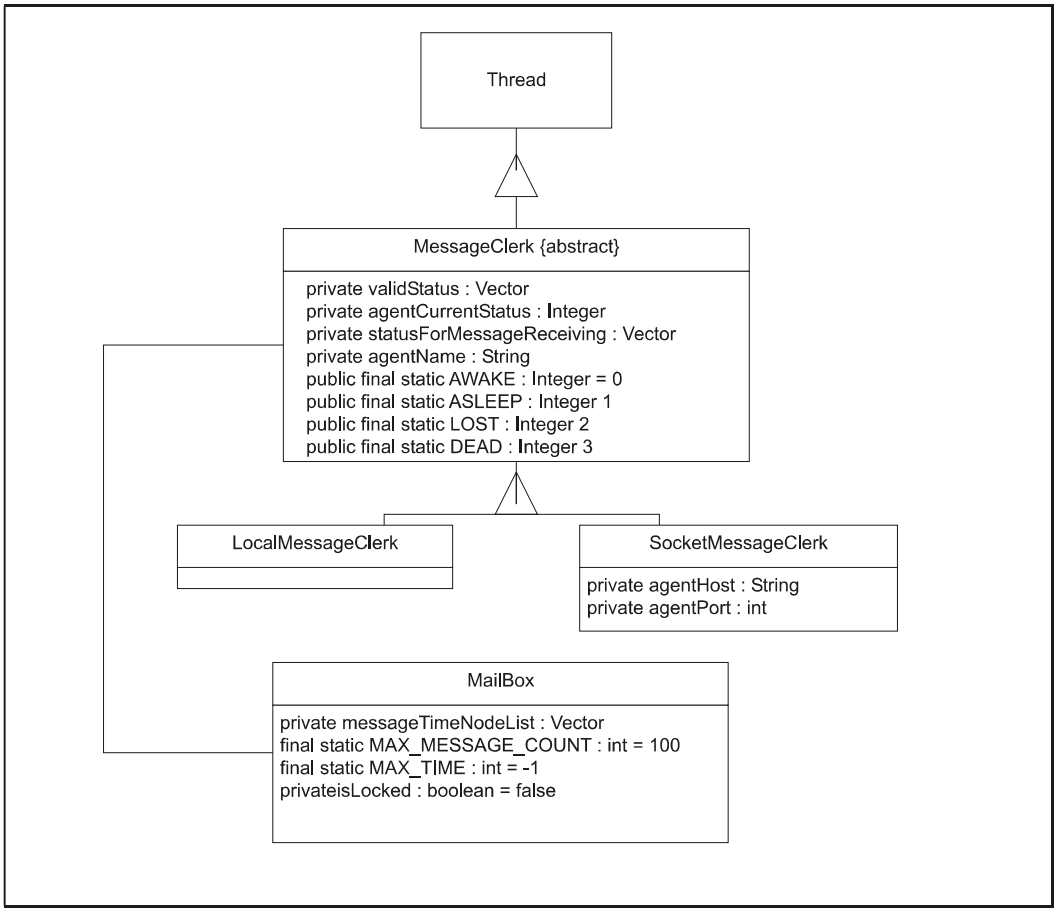

Fig. 8. Class hierarchy and associations for message switching from the ISAME environment.

out the necessary verifications and routes it towards the appropriate *MessageInterpreter*. To interpret a message, an agent must insert it in the *MessageSender*'s queue of messages ready for sending. For the agents to carry out an efficient management of the messages' communication ring, the *MessageReceiver, MessageSender* and *MessageHandler* constitute autonomous thread processes that function parallelly. Thus, each agent can simultaneously receive, process and send messages.

In order to meet the needs of all agents, different classes have been derived from both the *MessageReceiver* and *MessageSender*. In order to send their message to the CBA for switching, all agents, excepting the CBA, use a message sender based on the *socket, SocketMessageSender*. For the CBA's needs, a class of *CBAMessageSender* has been derived. Similarly, all agents use a message receiver based on the *sockets, SocketMessageReceiver*. The classes used to facilitate the CBA's work are presented in Figure 8.

4.2. Hardware and Software Environment

The set of classes in the ISAME environment, set of users and interface classes have been realized with the help of the following software:

- operating systems:
  Windows 95/98/NT/2000/XP
- programming language: **Java**
- development tools: Borland C++ 5.1, Symantec Visual Café 1.0, Java Development Kit (JDK) 1.0.1, and Sun Microsystem 1.1 Development Kit, JavaAgent's object library and also with the J2SE SDK 1.4 version.

The ISAME environment and its agents have been developed on PC machines connected to a Microsoft Windows network NT. The core of ISAME (support agents, request agents, mailboxes and messengers) is located in ISAME's server. The other machine behaves like a client of the ISAME environment and is the location of the agent created to carry out testing. All agents distributed on the server or among the clients communicate via sockets and through TCP/IP links. The number of sockets can vary from one machine to another.

According to RFC 2391, for most TCP/IP hosts, port range 0-1023 is used by the servers listening for incoming connections. Clients trying to initiate a connection typically select a port in the range of 1024-65535. However, this convention is not universal (and not always followed). It is possible for client nodes to initiate connections using a port number in the range of 0-1023, and there are applications listening on port numbers in the range of 1024-65535 [22]. In the case of ISAME, it attributes addresses from 1024 to 4999 to client processes. These are the *SocketMessageClerk* and *SocketMessageSender* objects. For server processes, ISAME uses numbers that are 5000 and more. These are the *SocketMessageReceiver* objects. The characteristics of the server used for developing and testing the ISAME architecture are the following:

- processor: Pentium 4, 1.8 GHz.
- RAM: 256 Mb.
- Network card: 3Com EtherLinkXL COMBO 10/100Mb Ethernet Adapter.
- Physical (IP) address: 206.106.88.156.
- *Socket* addresses for ISAME only:
  - CBA, the ISAME environment point of entry: socket 5000.
  - RAUA, RAQA and RAIA: addresses from sockets 5002 to 5004.
- For testing, 10 socket addresses have been reserved to query agents (QA): from sockets 5005 to 5014. This number of addresses can be increased according to the server's capacity.

The characteristics of the client machine used are:

- Processor: Pentium 4, 1.8 GHz.
- RAM: 256 Mb.
- Network card: NT 3COM NIC ETHERNET 10/100 PCI.
- Physical (IP) address: 206.167.88.146.
- *Socket* address used for the test user agent: socket 5001.

5. Analysis of the Implementation Results

Objects implemented in ISAME allow for achieving a certain number of tasks such as registering and withdrawing user agents and information, processing requests, and any interaction related to this processing. In this context, the ISAME environment is active on the server machine (CBA, RAUA, RAQA are thus waiting for messages) and a new user agent desires to take advantage from information retrieval services offered by ISAME. For this purpose, the user agent must follow the following steps:

1. registration in the ISAME environment;
2. sending of the user request to the ISAME environment;
3. eventual withdrawal from the ISAME environment.

These three steps have been executed in the ISAME environment. Figures from 9 to 15 show content of the messages exchanged among agents.

The evaluation of ISAME is based on the results obtained from the implementation, expected potential of unimplemented elements, and comparison to the UDML project. Until now, there has been no doubt that UDML is the most ambitious research project in the sector of information retrieval.

5.1. ISAME as Access Point

The ISAME environment offers a unique access point to the virtual library's set of services and content. However, although information filtering services and research results are part of the ISAME environment support services, the quality and quantity of information content remain always strongly dependent on the quality and quantity of content supplied by the agents registered at ISAME. This is due to the fact that information sources are external to the ISAME environment, and thus, they are not controlled by the environment as such. This problem is not peculiar to ISAME. Rather, it is inherent to any multi-agent architecture.

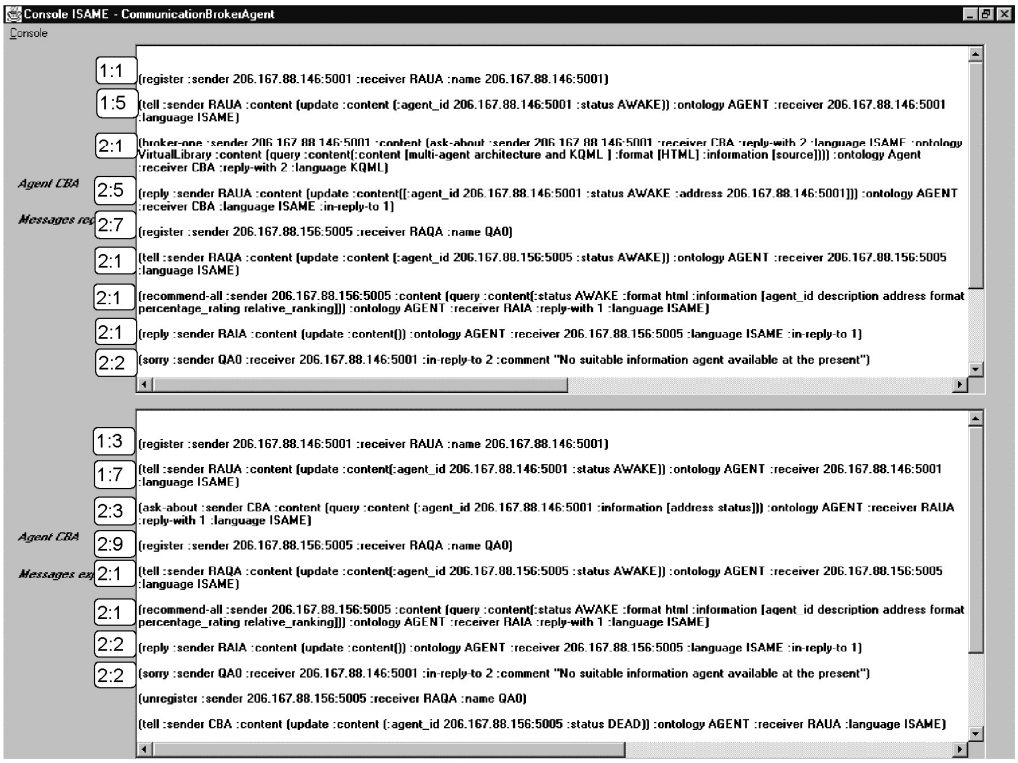

Fig. 9. Messages exchanged by the CommunicationBrokerAgent.

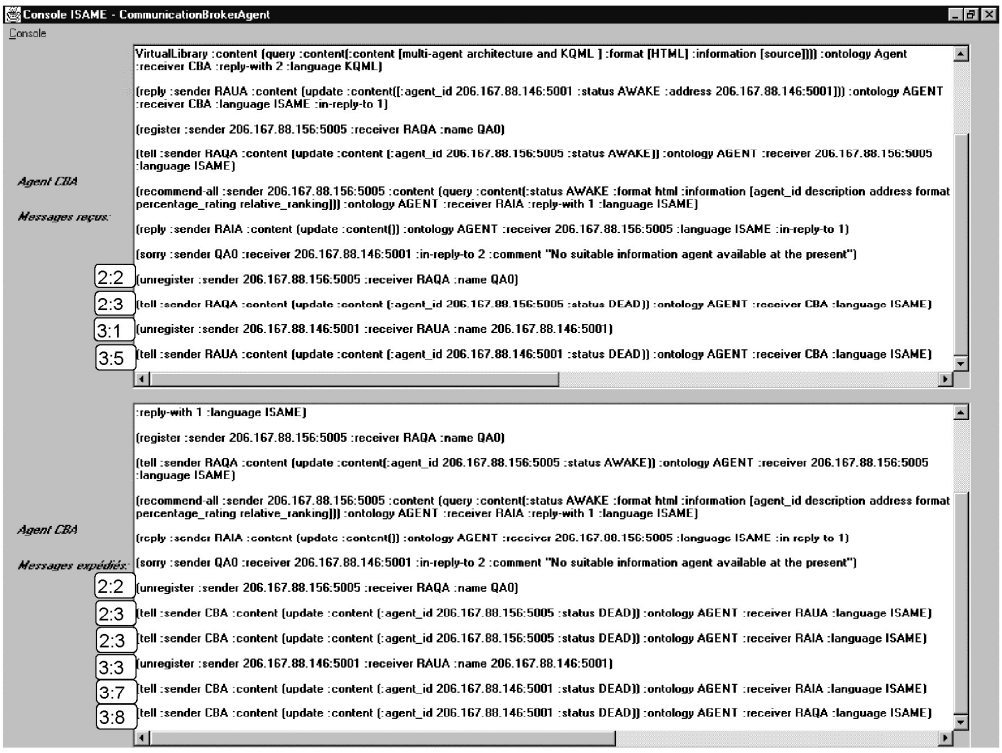

Fig. 10. Messages exchanged by the CommunicationBrokerAgent (continued).

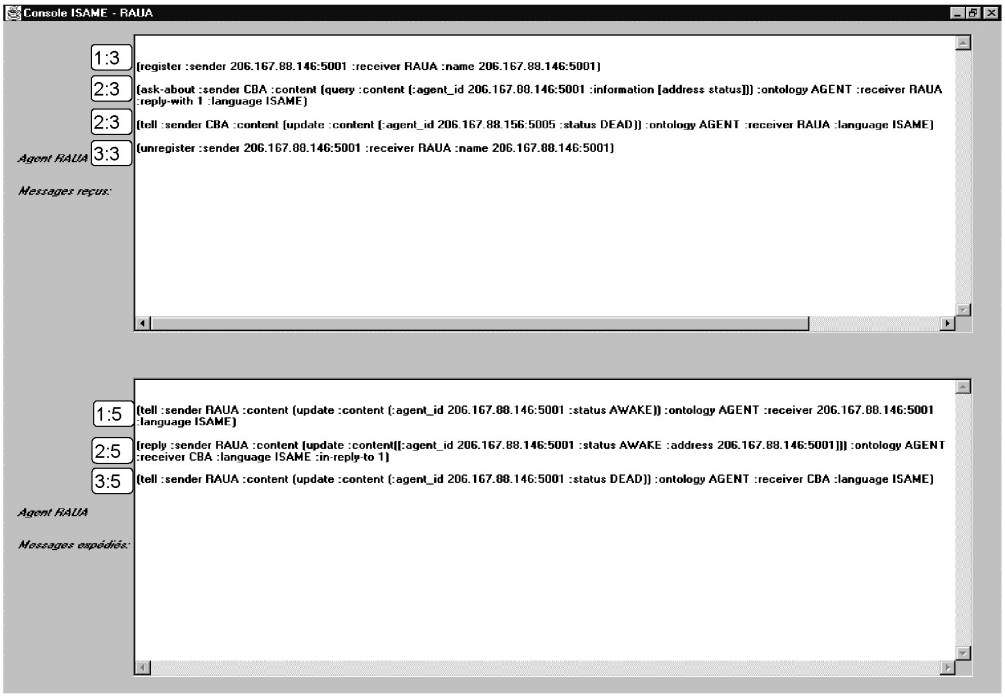

Fig. 11. Messages exchanged by RAUA.

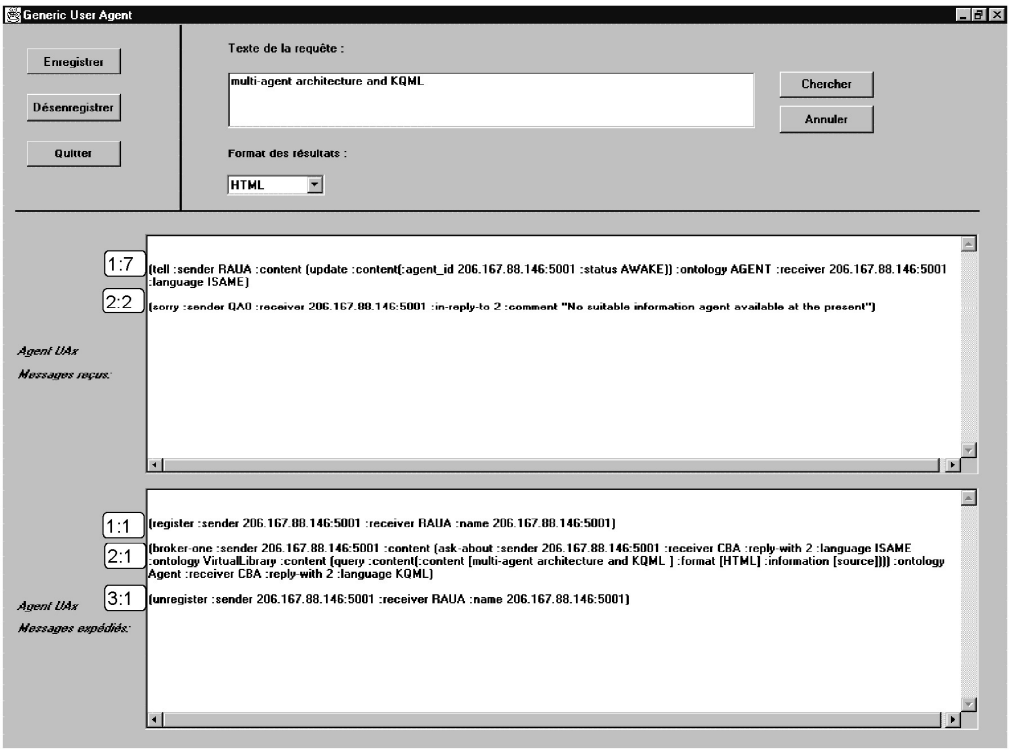

Fig. 12. Messages exchanged by the UAx.

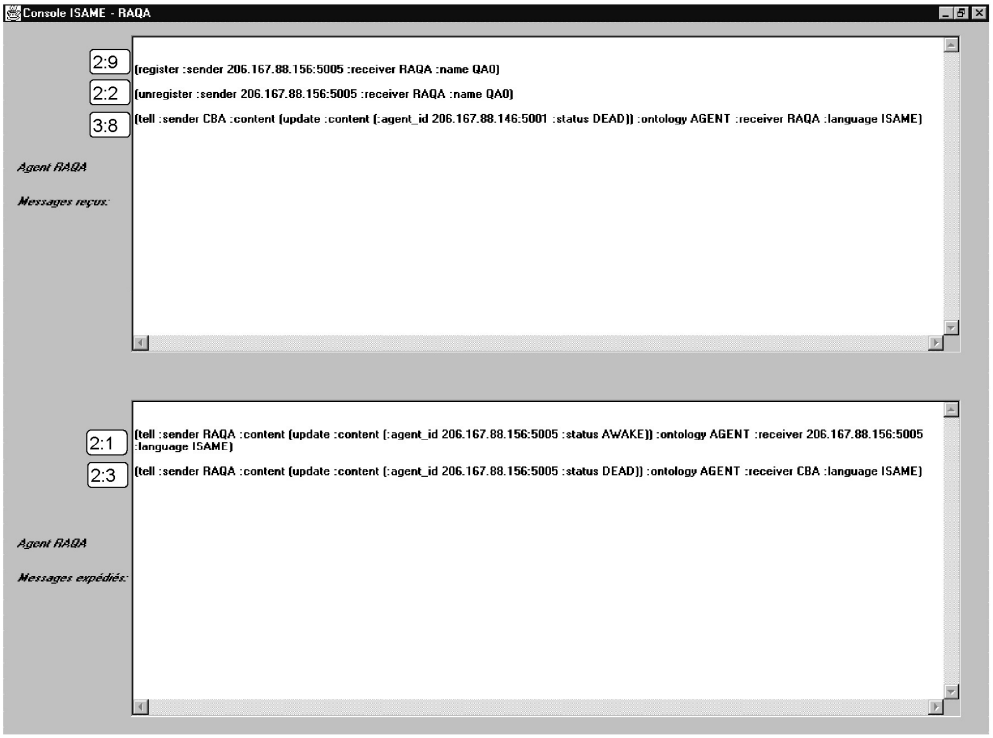

Fig. 13. Messages exchanged by RAQA.

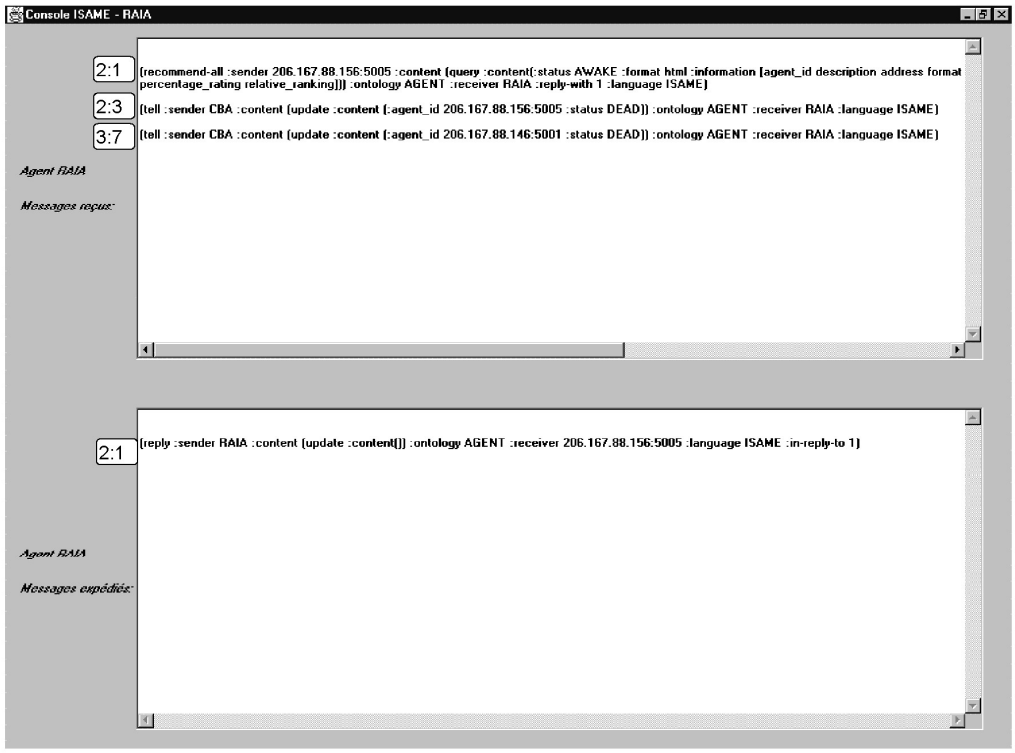

Fig. 14. Messages exchanged by RAIA.

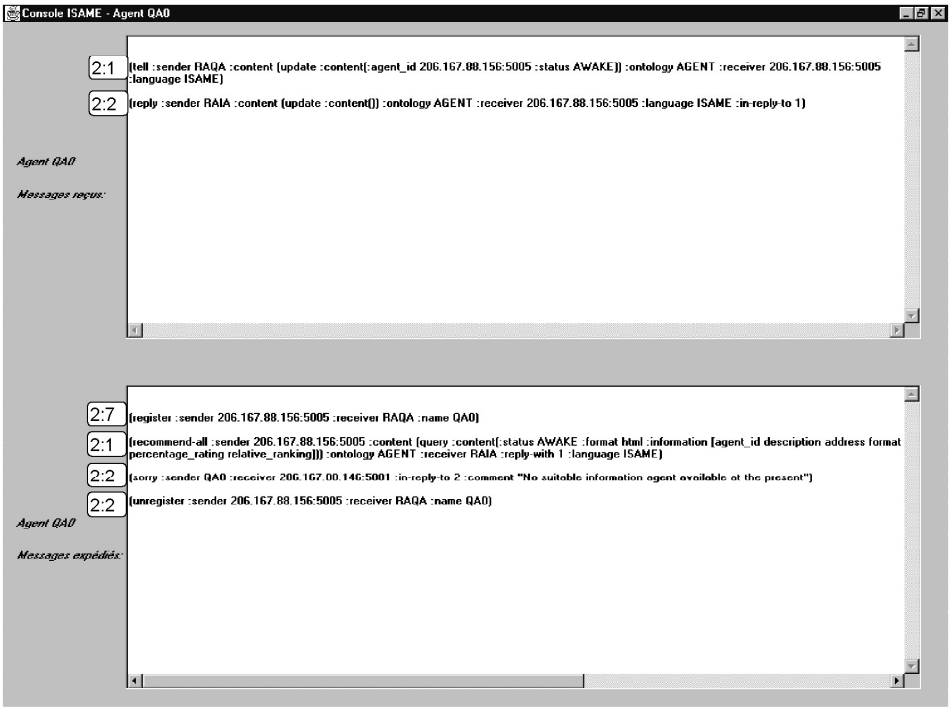

Fig. 15. Messages exchanged by QA0.

To minimize the impact of the problems inherent to distributed information retrieval systems, we proceeded by supplying the agents with a range of tools such as languages and tools for homogenizing communication, languages interpreters, and types of ontology. This is done in order to avoid the ambiguity and the simplified integration of new agents to the environment. The validity of such tools must be tested through the implementation of user and information agents. These characteristics are not only found on ISAME, but also on most multi-agent architectures dedicated to intelligent research. Notably, this is the case of UDML projects and JavaAgent that are based on agent-registering centralized systems and services exchanged among agents.

Like ISAME, UMDL uses several registry agents that preserve information on different categories of agents: users, information and mediation agents (agents that are responsible for the planning of requests, called *Query Planners*, and the registries themselves). If in ISAME, UAx and IAy must explicitly send their message to the appropriate registry agent (RAUA or RAIA), the UMDL environment does not seem to make this distinction explicit. The documentation now available on UMDL does not allow for determining how this distinction is being carried out.

5.2. Independence of User and Information Agents

Results obtained from the implementation of ISAME support the argument for the independence of agents. In effect, thanks to the work of support agents, request agents, internal communication agents and common forms of ontology, the UA and IA agents remain totally independent from each other, in their mode of implementation as well as in their availability.

A request placed by an UA_x at ISAME is totally delegated to a QA agent until the return of results. This request benefits from a set of services supplied by the environment (UA and IA agents' profiles, handling of persistent requests, optimizing choice of information sources, filtering and delivering results).

Total isolation of the UA_x from the selection of information sources, gathering and filtering of information results and knowledge, largely simplifies the work of designing user agents. From the users' perspective, however, this isolation may become an unacceptable constraint for the designers who desire to exercise more control over the request. During the processing of a user request, UA and IA agents do not have to think about the complexity and diversity of services supplied by ISAME. This independence comes from the standardization of communication languages and types of ontology. In effect, as far as external agents use valid performative statements, and as long as these agents respect the syntax and semantic of known languages and types of ontology in the environment, the quality of interactions among external and internal agents is secured.

It should be noted that dynamism of user and information agents is not always supported in intelligent information retrieval projects. A small number of reviewed research projects present an open architecture allowing for the integration of new information sources or services. Several of these projects, although based on a multi-agent approach, consist of closed boxes supplying the user with a fixed request interface. That is an imposed user agent. Moreover, only system designers are able to add or withdraw information sources from the environment. Thus, designers totally control the contents supplied to the users.

5.3. Optimization of Communication through Mailboxes

Thanks to mailboxes' services, in order to secure the return of information from solicited agents, ISAME's external agents do not need to be permanently connected. Similarly, an agent wishing to supply an answer does not depend on the availability of its partner. This useful task is carried out by the messenger system and mailboxes provided by ISAME.

However, if an agent remains inactive in ISAME for a long period of time, the message put in waiting will not be delivered. Such messages will rather be destroyed.

Communications among UMDL agents are based on the CORBA (Common Object Request Broker Architecture) standard [16]. This standard allows for communication among distributed objects. Agents delegate exchanged messages through the CBA. Mailboxes increase the

total number of messages exchanged in ISAME, by requiring an appropriate coordination among different participants. However, communication through CORBA leads to other constraints: imposition of one or several ORBs (Object Request Broker) and registration of agents or their objects of communications at this ORB. This work can be largely extended in the case where the UMDL server takes care of the entire number of all the constraints.

5.4. Comparison with system JATLite

Under different aspects we could compare our work with JATLite: The main part of the system is the *Communication Broker,* which facilitates communication among all agents. This agent needs help from the *Register Agent* and maintains communication with each agent mailboxes. To preserve continuous flow, the clerk will carry the queue in order to prevent outdated messages being processed by an agent. As mentioned in Table 1 and Figure 1, we could show that ISAME environment is composed of both categories of agents: External agent and Internal Agent. *Information Agent* and *User Agent* are both *External agents. Query Agent, Source Filtering Agent, Result Filtering Agent, Registry Agent* are *Internal Agent*. The *Communication Area* manages the exchange through TCP/IP Protocol among the mechanism of Messenger and /or Mailboxes. The messenger takes care of the synchronization. In JATLite the role of the Router Server includes different services such as: Message Queuing, Message Routing, Agent Name Server, Register, Security, List Agents, Disconnect and message reservation. In this paper, this role is being managed by the *Communication Broker*, but this agent delegates responsibilities among other agents or supports the agents. In this context this agent is also a support agent.

6. Conclusion

This paper proposed a multi-agent architecture, called ISAME and designed for intelligent information retrieval from heterogeneous distributed sources. ISAME constitutes a virtual library that supplies the agents with a simplified access to a set of dynamic information sources available under electronic formats, as well as services for facilitating and optimizing information retrieval.

The elaboration of ISAME has several aspects including: analysis of the virtual library concept, evaluation of multi-agent systems, as well as possible solution to implementation needs, elaboration of a multi-agent architecture supporting the entire needs and services related to virtual libraries and implementation and analysis of the implemented architecture.

ISAME proposes a series of support agents that serve as intermediaries among agents who produce and consume information. At the communication level, ISAME tries to short-circuit the problems that are inherent to blackboard techniques by implementing a hybrid technique based on the use of mailboxes and messengers. The architecture also uses TCP/IP communication protocols, and proposes a series of garbage-collection mechanisms that allow to avoid the preservation and propagation of information among agents, or messages that become inaccessible or outdated, as well as the use of resources that became undesirable. To the extent that the ISAME architecture is essentially linked to the problem of information retrieval, it contributes to establishing elements that allow the users, or information source monitors, to participate in the electronic exchange of information through specialized agents. These elements comprise the use of a precise minimal subset of performative statements of KQML and ISAME-L types and two pre-defined forms of ontology, VirtualLibrary and Agent.

Originality or the power of ISAME architecture resides in the way it simplifies information retrieval from heterogeneous and distributed information sources by making these resources transparent to the users. Although only a part of the ISAME's architecture has been implemented, it is probable that, due to the work of intelligent agents, the users will not have to worry about the implementation method, localization, availability and the interaction language related to each of the sources.

ISAME also offers advantages of an open multi-agent structure, since it allows, at any time, new agents representing the users and information sources to subscribe and leave the environment, that means an environment where the

resources are dynamically modified according to each one's availability.

However, some of ISAME's current elements must be reconsidered in the future, in order to improve the architecture and various processes. Firstly, in order to avoid any catastrophic loss of information collected by the user, it is necessary to revisit the method used to exclude agents. Moreover, it is also desirable that ISAME not only integrates a more elaborate model for establishing agent profiles, but also preserves, for a certain period of time, the profiles of excluded agents. This is a necessary requirement when an agent is being re-registered in reasonable delays. Finally, the implementation of the CBA and registry agents could be reviewed in a manner allowing for a distribution of these agents on different servers. In the end, this permits interconnection of different ISAME environments that supply agents with more hardware and software resources.

Acknowledgement

The author would like to thank the anonymous referees for their work and insightful comments on this paper, as well as Mrs Sabine Kebreau for helpful comments and contributions to the revised version of this paper. This work has been supported in part by the Telelearning Network Centres of Excellence of Canada.

References

[1] BAUER, B., MULLER, J.P., ODELL, J. (2000), "Agent UML: A formalism for specifying multi-agent software systems," *First International Workshop on Agent Oriented Software Engineering, Limerick Ireland*, June, pp. 91–104.

[2] BIRMINGHAM, W.P. (1995), "An Agent-Based Architecture for Digital Libraries," *D-Lib Magazine*, July.

[3] CASTRO, J., KOLOP, M., MYLOPOULOS, J. (2001), "UML for Agent-Oriented Software Development: The Tropos," (UML' 01 Conference), University of Toronto, October. Available at the following URL address: `http://www.cs.toronto.edu/km/tropos/uml01jm.pdf`

[4] COHEN, P.R., LEVESQUE, H.J. (1995), "Communicative Actions for Artificial Agents," *Proceedings of the First International Conference on Multi-Agent Systems*(ICMAS '95), June, San Francisco, pp. 65–72.

[5] *Digital Library Project Research Proposal.* Available at the following URL address: `http://http2.sils.umic|h.edu/UMDL/HomePage.html`

[6] FININ, T. ET AL. (1993), "Specification of the KQML Agent-Communication Language," Available at the following URL address: `http://www.cs.umbc.edu/kqml/mail/kqml`

[7] GENESERETH, M.R., FIKES, R.E. (1992), *Knowledge Interchange Format Version 3.0 Reference Manual.* Report Logic-92-1, Stanford University, Logic Group, USA.

[8] GIORGINI, P. (2001), "Multi-Agent Architecture as Organizational Structures," Submitted to Journal of Autonomous Agents and Multi Agent Systems. Available at the following URL address: `http://www.cs.toronto.edu/ mkolp/org-mas-ijcis.pdf`

[9] JENNINGS, N.R., CORERA, J.M., LARESGOITI, I. (1995), "Developing Industrial Multi-Agent Sytems," *Proceedings of the First International Conference on Multi-Agent Systems* (ICMAS '95), June, San Francisco, pp. 423–430.

[10] KING, J.A. (1995), "Intelligent Retrieval," *AI Expert*, 10(1), January, pp. 15–17.

[11] KING, J.A. (1995), "Intelligent Agents : Bringing Good Things to Life," *AI Expert*, February, pp. 17–19.

[12] KNOBLOCK, C.A., ARENS, Y. (1994), "An Architecture for Information Retrieval," *AAAI Symposium on Software Agents*, March, Stanford University, pp. 49–56.

[13] LANDER, S., LESSER, V. (1994), *Sharing Meta-Information to Guide Cooperative Search Among Heterogeneous Reusable Agents.* Technical Report pp. 94–48, University of Massachusetts, Amherst, USA.

[14] LAWRENCE, S., GILLES, C.L., BOLLACKER, K. (1999), "Digital Librairies and Autonomous Citation Indexing," *IEEE Computer*, 32(6), pp. 67–71.

[15] MAYFIELD, J., LABROU, Y., FININ, T. (1995), "Evaluation of KQML as an Agent Communication Language," *Workshop on Agents Theories, Languages and Architectures*, IJCAI '95, August, Montreal, Québec, pp. 282–291.

[16] MOWBRAY, T.J., ZAHAVI, R. (1995), *The Essential CORBA: Systems Integration Using Distributed Objects.* Object Management Group Publishers.

[17] NII, H.P. (1993), "Blackboard Systems at the Architecture Level," *Expert systems With Applications*, 7, pp. 43–54.

[18] SINGH, M.P. (1993), "Declarative Representations of Multiagent Systems," *IEEE Transactions on Knowledge and Data Engineering*, 5(5), October, pp. 721–740.

[19] WOOD, M.F., DELOACH, S.A. (2000), "And Overview of the Multiagent Systems Engineering Methodology," *Lecture Notes in Computer Science*, AOSE 2000, Volume 1957, Issue, pp. 207–222.

[20] WOOLDRIDGE, M. (1992), *The Logical Modelling of Computational Multi-Agent Systems.* Ph.D. Thesis, Department of Computation, University of Manchester, U.K., August.

[21] WOOLDRIDGE, M., JENNINGS, N.R. ET AL. (2000), "The Gaia Methodology For Agent-Oriented Analysis And Design," *Journal of Autonomous Agents and Multi-Agent Systems*, 3(3), pp. 285–312. Available at the following URL address: `http://citeseer.nj.nec.com/wooldridge00gaia.html`

[22] RFC 2391, (1998), `http://rfc-2391.rfc-index.org/rfc-2391-3.htm`

Received: August, 2000
Revised: February, 2003
Accepted: February, 2003

Contact address:
Sophie-Julie Pelletier
Softimage
Montréal, Canada

Samuel Pierre
Department of Computer Engineering
École Polytechnique de Montréal
C.P. 6079, Succ. Centre-Ville
Montréal, Qué., Canada, H3C 3A7
Phone: (514) 340-4711 ext. 4685
Fax: (514) 340-3240
e-mail: `samuel.pierre@polymtl.ca`

Hai Hoc Hoang
Department of Computer Engineering
École Polytechnique de Montréal
C.P. 6079, Succ. Centre-Ville
Montréal, Qué., Canada, H3C 3A7

SOPHIE PELLETIER holds a B.S. degree in computer science from the University of Sherbrooke (Canada) and a Master degree in applied sciences from École Polytechnique of Montréal. As a researcher at CITI (Centre d'innovation en technologies de l'information), she has developed many generic and specific applications based on the multi-agent systems, and this, in collaboration with various private enterprises. She now is a headmaster in information technology at Microcell Telecommunications Inc. in Montréal.

SAMUEL PIERRE received the B.Eng. degree in civil engineering in 1981 from École Polytechnique de Montréal, the B.Sc. and M.Sc. degrees, both in mathematics and computer science, in 1984 and 1985, respectively, from the UQAM, Montréal, the M.Sc. degree in economics in 1987 from the Université de Montréal, and the Ph.D. degree in electrical engineering in 1991 from École Polytechnique de Montréal. He is currently a Professor of computer engineering at École Polytechnique de Montréal where he is Director of the Mobile Computing and Networking Research Laboratory (LARIM) and Industrial Research Chair NSERC/Ericsson in Next-Generation Mobile Networking Systems.

Dr. Pierre is the author of four books, co-author of two books and five book chapters, as well as over 200 other technical publications including journal and proceedings papers. He is a Fellow of the Engineering Institute of Canada. His research interests include wireline and wireless networks, mobile computing, performance evaluation, artificial intelligence, and electronic learning (e-learning). He is a senior member of IEEE and a member of ACM. He is an Associate Editor of IEEE Communications Letters and IEEE Canadian Review. He also serves on the editorial board of Telematics and Informatics published by Elsevier Science.

HAI HOC HOANG was born in Thai-binh, Vietman. He received the B.Eng. degree in mining engineering from École Polytechnique (Montréal) in 1967, the B.S. degree in economics, and the M.S. and Ph.D. degrees in operations research form the University of Montréal (Canada) in 1967, 1970 and 1973, respectively. In 1972 he joined the Faculty of École Polytechnique (Montréal), where he currently is Professor at the Department of Computer Engineering. He has served as a consultant for several government agencies and industries. His current interests are in modeling and optimizing transportation, communication and computer systems. Dr. Hoang is a member of the ACM, IEEE and INFORMS.