

# Distributed Artificial Intelligence for Distributed Corporate Knowledge Management

Fabien Gandon, Rose Dieng-Kuntz

ACACIA project - INRIA, 2004 route des Lucioles - BP93  
06902 Sophia Antipolis Cedex - France  
{Fabien.Gandon ; Rose.Dieng}@sophia.inria.fr

**Abstract.** We present a multi-agents architecture that was built and tested to manage a corporate memory based on the semantic Web technologies.

## 1 Introduction

The advent of information society led organizations to build intranets that are becoming corporate nervous systems. They memorize critical pieces of information and irrigate the organizational entities with them. A corporate memory is an explicit, disembodied and persistent representation of knowledge and information in an organization, in order to facilitate their access and reuse by members of the organization, for their tasks [10]. The stake in building a corporate memory management system is the coherent integration of this dispersed knowledge in a corporation with the objective to promote knowledge growth, knowledge communication and to preserve knowledge within an organization [30].

The field of multi-agents information systems is very active and most promising application areas are, among others, distributed Web-based collaborative work, information discovery in heterogeneous information sources and intelligent data management in the Internet or corporate intranets [21].

Some of these systems are specialized for information retrieval from heterogeneous information repositories. InfoMaster [18] uses a global schema and Carnot [7] a global ontology (Cyc) to build mappings for wrappers of heterogeneous sources. As in RETSINA [9], these systems rely on wrapper agents to provide an homogeneous view of the different sources while the integration is handled by middle agents planning query resolution, information integration and conflict resolution. Information Manifold [23] and InfoSleuth [25] have multiple ontologies but they do not handle mapping between them. SIMS [2] uses Description Logics to handle multiple ontologies and translate queries when there is no loss. Finally OBSERVER [24] takes into account the inter-ontology relationships to tackle the loss of information when translating queries.

SAIRE [26] and UMDL [31] manage distributed large scale libraries of digital documents to offer means to find relevant documents and manage indexing.

Finally some projects focus on knowledge management inside organizations. CASMIR [4] and Ricochet [5] focus on gathering information and adapting interactions to the user's preferences, learning interests to build communities and enable collaborative filtering inside an organization. KnowWeb [12] relies on mobile agents to support dynamically changing networked environment and exploits a domain model to extract concepts describing documents and use them to answer queries. RICA [1] maintains a shared taxonomy in which nodes are attached to documents and uses it to push suggestions to interface agents according to user profiles. Finally FRODO [13] is dedicated to building and maintaining distributed organizational memories with an emphasis on the management of domain ontologies.

The CoMMA project we present here, belongs to this last category. It aims at implementing and testing a corporate memory management framework based on agent technology. Two application scenarios had been submitted by industrial end-users, involving information retrieval tasks for employees:

- *Integration of a new employee to an organization*: the main user of the system is a newcomer that needs to acquire some knowledge to become fully operational and integrated to the organization.
- *Technology monitoring*: assist the detection, identification and diffusion of information about technology movements to disseminate innovative idea in the company and improve the response time by providing the relevant information as soon as possible.

Thus, CoMMA does not target the Internet and the open Web but corporate memory accessible through an intranet based on Web technologies *i.e.* an intraweb. The system does not directly manage documents, but annotations about documents. We suppose documents are referenced by URI and, as explained in section 2, we index them using semantic annotations relying on the semantic Web technologies. CoMMA focuses on three functionalities needed for the two application scenarios: improve precision and recall to retrieve documents, using semantic annotations; proactively push information using organization and user models; archive newly submitted annotations.

*On the one hand, individual agents locally adapt to users and resources they are dedicated to*: an interface agent adapts to its user; an archivist agent is associated to a repository of annotations and manages it locally providing access to this knowledge base to other agents; etc.

*On the other hand, thanks to cooperating software agents distributed over the network, the whole system capitalizes an integrated view of the corporate memory*: agents are benevolent with the common collective goal of managing the corporate semantic web; there is no real competition as it could be found in a market place type application; middle agents provide matchmaking for service resolution.

The MAS architecture of CoMMA, detailed in section 3, aims at providing flexibility, extensibility and modularity to allow a deployment above the distributed information landscape of a company, while enabling the user to access an integrated view of its content. CoMMA was a two-year project that ended February 2002 with a system implemented in Java using the FIPA compliant platform JADE [3]. The final

prototype was demonstrated and discussed during a trial and an open day and we give our return on experience in the last section on conclusion and discussion.

## 2 Corporate Semantic Web: an Annotated World for Agents

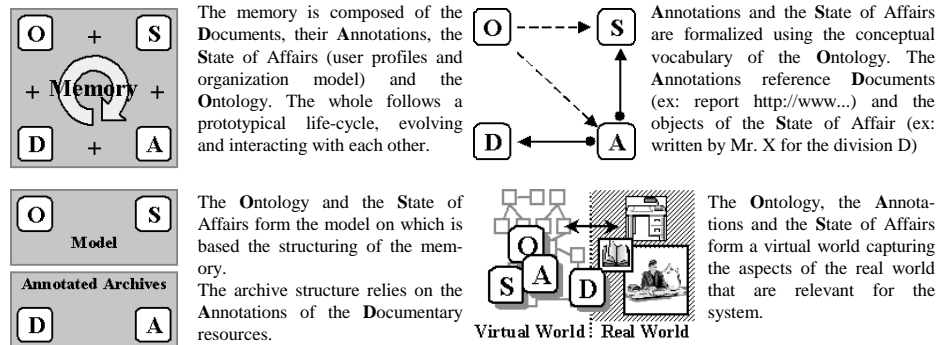
In their article about “Agents in Annotated Worlds” Doyle and Hayes-Roth [11] explained that annotated environments containing explanations of the purpose and uses of spaces and activities allow agents to quickly become intelligent actors in those spaces. For information agents in complex information worlds, it means that annotated information worlds are, in the actual state of the art, a quick way to make information agents smarter. If the corporate memory becomes an annotated world, agents can use the semantics of the annotation and through inferences help the users exploit the corporate memory.

The Resource Description Framework (RDF) [22] is a foundation to describe the data contained on the Web through metadata: it enables automatic processing and interoperability in exchanging information on the Web by providing a framework to annotate resources referenced by their Uniform Resource Identifier (URI). We use RDF to build *corporate semantic Webs* where software agents use the semantics of annotations and, through inferences, help users exploit the content of the memory. RDF provides a model for representing named properties of resources and property values and an XML syntax. The described resources are of any type. Schemas define the conceptual vocabulary that is used in RDF statements. RDF Schema (RDFS) [6] is a schema specification language that provides a basic type system for use in RDF models. It specifies the mechanisms to define the classes of resources described (e.g. books, Web pages, people, companies, etc.) and the properties used for descriptions (e.g. title, author, name, activity, etc.). Types of classes and properties are organized in two hierarchies. Property types define permitted values (*range*) and classes of resources they can describe (*domain*).

With RDF, we describe the content of documents and the organizational state of affair through semantic annotations and with RDFS we specify the ontology O'CoMMA [17] providing the conceptual primitives used to represent these annotations (Figure 1). Agents use and infer from these annotations to successfully search the mass of information of the corporate memory. The state of affairs includes a description of the organization and profiles of its members.

User profiles capture aspects of the user that were identified as relevant for and exploitable by agent behaviors. A profile is an RDF annotation about a person. It contains administrative information and explicit preferences such as topic interests. It also positions the user in the organization: role, location and potential acquaintance network, enabling the system to target push actions. In addition, the system derives information from the usage made by the user. It collects the history of visited documents and user's feedback and from this it learns some of the user's interests. These derived criteria are then used for results presentation or push technology enabling the emergence of communities of interest.

The enterprise model is an oriented, focused and somewhat simplified explicit representation of the organization. So far, the enterprise modeling field has been mainly concerned with simulation and optimization of the production system design. They provide benchmark for business processes and are used for re-engineering them. But organizations became aware of the value of their memory and the fact that organization models have a role to play in this application too [28]. In CoMMA, the model aims at supporting corporate memory activities involved in the application scenario by giving the agents insight into their organizational context and environment. Thus they can exploit the aspects described in this model for their interactions with other agents and, above all, with users. We used RDF to implement our organizational description, annotating the organizational entities (departments, activities, etc.) with their relations (manages, employs, includes, etc.).



**Fig. 1.** Structure of the memory

The ontology O'CoMMA plays a pivotal role enabling the description of annotations and organizational and user models that are exchanged between agents. Thus, it provides the semantic ground for the communication and co-operation of the different agents that form the CoMMA architecture. We now detail this architecture.

### 3 Multi-agents Architecture of CoMMA

The architecture of the CoMMA system was obtained following an organizational approach for designing a multi-agent system (MAS). The design rationale is detailed in [15]. We only describe here the final implemented architecture. We then focus the sub-society dedicated to the management of annotations, its principles and functioning.

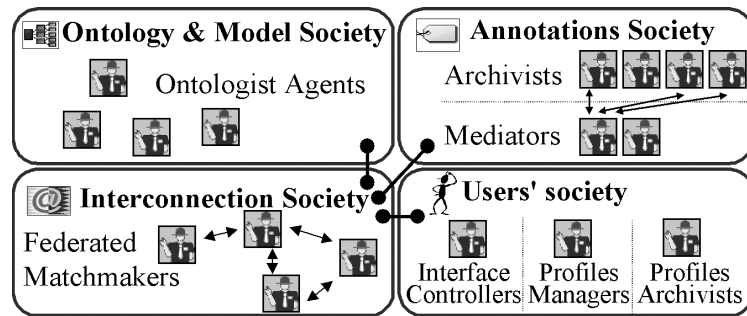
#### 3.1 The societies of CoMMA

The architecture is a structure that portrays the different kinds of agencies existing in an agent society and the relationships among them. A configuration is an instantiation

of an architecture with a chosen arrangement and an appropriate number of agents of each type. One given architecture can lead to several configurations. The CoMMA architecture was designed so that the set of possible configurations covers the different corporate organizational layouts foreseeable. The MAS flexibility and modularity is used to provide a maneuver margin and postpone some choices until the very last stage of deployment. In the case of a multi-agents corporate memory system, the configuration depends on the topography and context of the place where the system is rolled out (organizational layout, network topography, stakeholders location, corporate policies). The configuration must be tuned to this information landscape and change with it. Among the possible configuration we also try to choose the one optimizing CPU and network workload.

The architecture was fixed at design time considering the functionalities CoMMA focuses on and the software components that will be included in the agents. It was divided into four dedicated sub-societies of agents (figure 2):

- A sub-society dedicated to ontology and organizational model;
- An annotation-dedicated sub-society;
- A user-dedicated sub-society;
- A connection-dedicated sub-society.



**Fig. 2.** Architecture of the MAS

In the four societies, ten agent roles were identified and specified and we detail them in the next section. Agent types have been implemented to play those roles and be provided within a complete solution to be deployed on an intranet. Every agent role was assigned to a partner of the project who had the expertise and the tools needed for implementing the corresponding behavior. Then an integration phased was carried out; it was extremely rapid and seamless since the combination of agents, ontology and specified protocols makes the different software components loosely-coupled both at run-time and design-time.

### 3.2 Specified Agent Roles and Implemented Agent Types

The user-dedicated sub-society relies on the fulfillment of four roles:

- *Interface Controller* (IC): this role is in charge of managing and monitoring the user interface. It is the only role with a limited lifetime which is the login session.
- *User Profile Manager* (UPM): this role is in charge of updating and exploiting the user's profile when the user is logged on to the system. It analyses the users requests and feedback to learn from them and improve the systems reactions.
- *User Profile Archivist* (UPA): this role is in charge of storing, retrieving and querying user profiles when requested by other agents.
- *User Profile Processor* (UPP): this role is in charge of performing proactive queries on the annotations using the user profiles to detect new documents that are potentially interesting for a user and push the information.

The IC role is played by one agent type. The IC agent is created at login and dies at logout. One of the major difficulty in implementing this agent is to try to provide a powerful and yet intelligible interface, that hides the complexity of the MAS and the semantic Web technologies. This agent is implemented using: Java Swing, an XSLT engine and a micro Web-browser to display the processed XML and the results. The IC interacts directly with the agents belonging to the ontology-dedicated sub-society but it relies on a contracted UPM agent to deal with the annotation-dedicated sub-society.

The UPM role is played by one agent type that uses machine learning techniques to learn the interest of the user and build an ordering relation to rank answers of the system and sort them before transmitting them to the IC. This agent was first implemented using the Weka library and then improved as detailed in [20]. The UPM also registers for new annotation notifications and forward them to the UPA.

One agent type, called UPA, is in charge of the UPP role and part of the UPA role concerning storage and retrieval of user profiles. Precise querying on user profiles is handle by another agent type (Annotation Archivist) that belongs to the annotation-dedicated sub-society. The UPA also compares new annotation notifications to user profiles to add consultation suggestions to the profiles.

CoMMA being based on the JADE platform [3], the agents of the connection sub-society play two roles defined by FIPA [14]:

- *Agent Management System* (AMS): this role is in charge of maintaining white pages where agents register themselves and ask for addresses of other agents on the basis of their name.
- *Directory Facilitator* (DF): this role is in charge of maintaining yellow pages where agents register themselves and ask for addresses of other agents on the basis of a description of the services they can provide.

AMS and DF agents are implemented and delivered with the JADE platform. They are directly used by the other agent types developed in CoMMA to form the acquaintance needed for their roles. The current prototype only uses the agent type as a service description, but further service description refinement is done in a second step as it will be detailed for the annotation-dedicated sub-society.

The agents from the ontology dedicated sub-society are concerned with the management of the ontological aspects of the information retrieval activity. The sub-society dedicated to ontology and model relies on two roles:

- *Ontology Archivist* (OA): this role is in charge of storing and retrieving the O'CoMMA ontology in RDFS.
- *Enterprise Model Archivist* (EMA): this role is in charge of storing and retrieving the organizational model in RDF.

Both roles are played by one agent type (OA) since they are really equivalent in their functioning and deployment. A replicated organization (identical roles and knowledge base) for the ontology sub-society is conceivable because CoMMA does not focus on the maintenance of multiple ontologies.

The agents of the annotation dedicated sub-society must play two roles:

- *Annotation Archivist* (AA): this role is in charge of storing and searching RDF annotations in a local repository it is associated to.
- *Annotation Mediator* (AM): this role is in charge of distributed query solving and annotation allocation. It is a mediator between agents requiring services on the memory and Annotation Archivists attached to one repository. It hides the distributed aspect of the memory from the other agents. This role also provides a subscription service for agents that wish to be notified of any new annotation added to the memory.

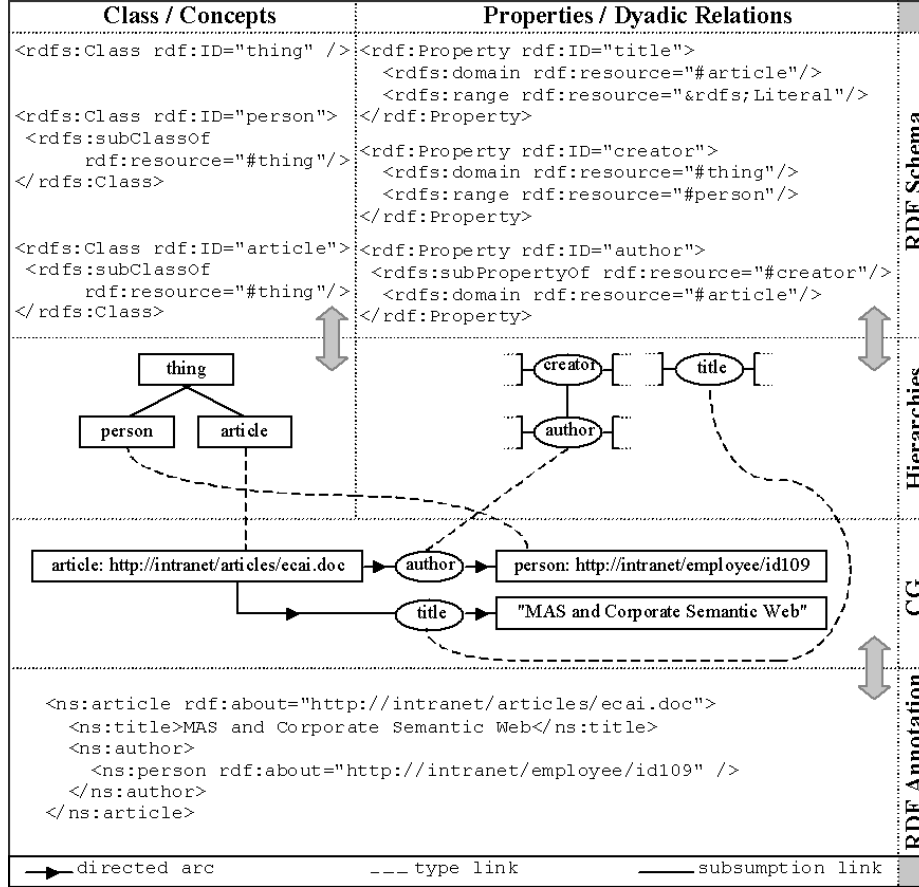
Two agents types were developed to fulfil these two roles. AAs are also used to include user profiles repositories in the query solving process. A hierarchical organization was chosen for this society because it separates the task of maintaining local annotation repositories from the task of managing distributed query solving and annotation allocation which allows us to distribute the workload.

### 3.3 Agents to handle annotation distribution

We now focus on the annotation dedicated sub-society to explain the principles it is based on and its implementation.

#### 3.3.1 RDF Annotations

RDF(S) was influenced by the graph data models and we manipulate it as a restriction of Sowa's Conceptual Graphs (CGs) [29]. A CG is a bipartite graph, where every arc links a concept node and a conceptual relation node. Both concept and conceptual relation have a type that can be either primitive or defined by a monadic lambda expression. Concept and relation types are organized in two hierarchies which are sets of type labels partially ordered by the subtype relation. Figure 1 shows that RDF schemas and statements can be translated, respectively, into type hierarchies and directed bipartite graphs. The RDF triple model only supports binary relations, thus RDF annotations generate CGs with dyadic relations of primitive type and the signature of relations are derived from the *domain* and *range* constraints.



**Fig. 3.** RDF(S) and CG model mapping

Type hierarchies are at the heart of ontologies for information searching and communication since they support inferences used when querying: specialization, generalization and identification. The CG projection operator is well adapted to annotation retrieval as it performs matching taking into account specialization of relations and concepts. The query language used is RDF with variables to indicate unknown parts and co-references, regular expressions to constrain literal values and operators to express disjunction or negation. The result of the projection is translated back into RDF. This mechanism provides a semantic search engine known as CORESE [8] and an API that we used to implement the behavior of the AA, AM and OA. We shall see now how our agents exploit the underlying graph model, when deciding how to allocate new annotations and when resolving distributed queries.



### 3.3.2 Annotation Agents

The duality of the definition of the word 'distribution' reveals two important problems to be addressed :

- Distribution means dispersion, that is the *spatial property of being scattered* about, over an area or a volume; the problem here is to handle the naturally distributed data, information or knowledge of the organization.
- Distribution also means the *act of distributing* or spreading or apportioning; the problem then is to make the relevant pieces of information go to the concerned agent (artificial or human). It is with both purposes in mind that we designed this sub-society.

The annotation-dedicated society is in charge of handling annotations and queries in the distributed memory. The submissions of queries and annotations are generated by agents from the user-dedicated society. Users are provided with a graphical interface to guide them in the process of building semantic annotations using concepts and relations from the ontology OCoMMA (see Figure 6 for a screenshot of the second prototype). From the annotation built in the GUI, the Interface Controller agent generates an RDF annotation; through this agent the user appears just like another agent to the rest of the MAS.

The submitted queries and annotations are then routed to the annotation-dedicated society. As we said, the latter is a hierarchical society: the agents playing the AM role are in charge of managing agents playing the AA role. The AM provides its services to other societies to solve their queries and, to do so, it requests the services of the AAs. On the other side, the AA role is attached to a local annotation repository and when it receives a request, it tries to fulfil it with its local resources in a way that enables the AM to handle the distributed dimension of the problem. The agents playing the role of AA and AM are benevolent and, once deployed, temporally continuous.

Distributed Database field [27] distinguishes two types of fragmentation: horizontal and vertical. By drawing a parallel between data / schema and knowledge / ontology we adapted these notions to RDF annotations.

- *Horizontal fragmentation* means that information is split according to the range of properties; for instance site<sub>1</sub> will have reports with a property 'title' ranging from "Criminality in agent societies" to "MAS control" and site<sub>2</sub> will have reports from "Naive resource distribution" to "Zeno paradox in loops".
- *Vertical fragmentation* means that information is split according to types of concepts and properties; for instance site<sub>1</sub> will have reports with their titles and authors and site<sub>2</sub> will have articles with their abstract and keywords. Fragmentation choices are made by the administrators when deploying the agents.

The stake is to find mechanisms to decide *where to store newly submitted annotations* and *how to distribute a query* in order not to miss answers just because the needed information are split over several AAs. These two facets of distribution are linked since the performance of distributed query resolution is closely related to the choices made for the distribution of annotations.

### 3.3.3 Annotation Distribution

In order to determine which AA should be involved during the solving of a query or to which one an annotation should be given, we compare the content of their archive thanks to a light structure called ABIS (Annotation Base Instances Statistics). It captures statistics, maintained by the AA, about its annotation base: the number of instances for each concept type, the number of instances for each property type and the number of instances for each family of properties. A family of properties is defined by a specialized signature corresponding to at least one instance present in the archivists base:

$$[\text{ConceptType}_x] \rightarrow (\text{PropertyType}_y) \rightarrow [\text{ConceptType}_z]$$

where the concept types are possibly more precise than the signature of *PropertyType<sub>y</sub>*. For instance, if there exists a property type *Author* with the following signature:

$$[\text{Document}] \rightarrow (\text{Author}) \rightarrow [\text{Person}],$$

we may have families of properties such as:

$$[\text{Article}] \rightarrow (\text{Author}) \rightarrow [\text{Student}],$$

$$[\text{Book}] \rightarrow (\text{Author}) \rightarrow [\text{Philosopher}].$$

This means that for each of these specialized signatures, there exists, in the archive of the corresponding AA, at least one instance using exactly these types. If a family does not appear in the ABIS, it means there is no instance of this very precise type. The ABIS captures the types for which an AA contributes to the memory. The ABIS of an AA is updated each time an annotation is loaded in the base: the annotation is decomposed into dyadic relations and possibly isolated nodes; for literal properties, the bounding interval  $[B_{low}, B_{up}]$  of their literal values is calculated.

When a system is deployed, AAs are started but they may have no annotation in their bases. Their statistics being void, the ABIS is not relevant to compare their bids. Moreover, it is interesting to be able to specialize individual agents according to the topography of the company network (eg. an AA on a machine of the human resources department for users' profile). The CAP (Card of Archives Preferences) is a light structure that captures the RDF properties for which the agent has a preference and, if specified, their range boundaries. Any specialization of these properties is considered to be part of the preferences of the AA and can be used for bidding.

Submitted annotations are not broken down *i.e.* we store them as one block. When a new one is submitted, the AM emits a Call For Proposal and starts a contract-net protocol [14] with the AAs. The AM measures how close a new annotation is from the ABIS and CAP of the candidate AAs in order to decide which one of them should win the bid. Details of the pseudo-distance calculation can be found in [16], it exploits a semantic distance defined on the hierarchies of the ontology and a lexicographical distance for literal values. Figure 4 shows an example of protocol for the allocation of a newly submitted annotation to an archivist agent.

When the annotation has been allocated, the AM that handled the CFP sends the content of the annotation to the UPAs that registered for new annotation notification. This triggers the push chain.

### 3.3.4 Query Distribution

Query solving involves several annotation distributed bases; answers are a merger of partial results. To determine if and when an AA should participate to the solving of a query, AAs calculate the overlap between their ABIS and the properties at play in the query. The result is an OBSIQ (Overlap Between Statistics and Instances in a Query), a light structure which is void if the AA has no reason to participate to the query solving or which otherwise gives the properties for which the AA should be consulted. Using the OBSIQ it requested before starting the solving process, the AM is able to identify at each step of the decomposition algorithm and for each subquery it generates, which AAs are to be consulted. The communication protocol used for the query solving is an extension of the FIPA query-ref protocol [14] to allow multiple stages with subqueries being exchanged between the AM and the AAs.

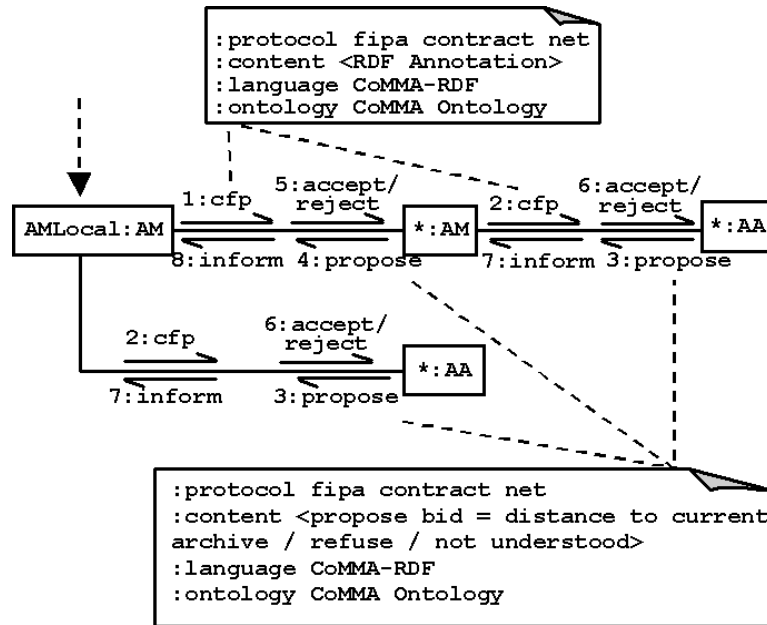


Fig. 4. Protocol for annotation submission

The decomposition algorithm consists of four stages: preprocessing for query simplification (e.g. remove cross-references), constraints solving (e.g. documents with a title containing "XML"), questions answering (e.g. find the name of the author) and final filtering (e.g. cross references such as "the title must contain the name of the author"). These stages manipulate the query structure through the Document Object Model (the DOM is an interface to manipulate an XML document as a forest). In our case, the structure is a tree that represents an RDF pattern and contains nodes representing resources or properties, except for the leaves that may be resources or literals. The resource nodes may have an URI and the AMs use them as cut/join point during query solving to build small subqueries that can be sent to the AAs to gather the information that could be scattered in several archives and merge the partial results. The four stages are detailed in [16].

### 3.3.5 Role of the Ontology

Figure 5 shows an example of message sent by an agent requesting the title of available memos. There are three nested levels: (a) a level using the FIPA ontology for general speech acts (b) a CoMMA ACL level using speech acts involved in the memory management (c) an RDF level using the O'CoMMA primitives to describe the pattern to look for.

The ontology is the cornerstone of distributed artificial intelligence mechanisms managing distributed knowledge. The ontological consensus provides a foundation on which we can build other consensus as, for instance, computational consensus: here, it is used as a consensual common space that enables us to defined a shared semantic (pseudo)-distances to compare the bids from different agents ; it also provides the primitives used to annotate the memory and to describe the type of knowledge architect agents have in their base so that the mediator can decide which agent it is relevant to contact to solve a given query. The example of the annotations-dedicated society shows how the ontology, the semantic Web and the agents are complementary to propose solutions from distributed artificial intelligence to the problem of distributed knowledge management.

```

a { (QUERY-REF
  :sender (agent-identifier :name localUPM@apollo:1099/JADE)
  :receiver (set (agent-identifier :name AM@apollo:1099/JADE))
  :content
    b { ((all ?x (is-answer-for
      (query
        :pattern
          c { <?xml version ="1.0"?> <rdf:RDF xml:lang="en"
            xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:comma="http://www.inria.fr/acacia/comma#">
              <comma:Memo><comma:Designation>?</comma:Designation>
              </comma:Memo>
            </rdf:RDF>
          ) ?x ) ) )
    :reply-with QuerylocalUPM987683105872
    :language CoMMA-RDF
  a :ontology CoMMA-annotation-ontology
    :protocol FIPA-Query
    :conversation-id QuerylocalUPM987683105872 )

```

(a) Primitives FIPA (b) Actes du Langage CoMMA (c) Requête RDF

Fig. 5. Message from an agent requesting title of memos

## 4 Results Evaluation and Conclusion

We presented a multi-agent system for the management of a corporate memory. The MAS architecture enabled us to integrate seamlessly several emerging technologies : agent technology, knowledge modeling, XML technologies, information retrieval techniques and machine learning techniques. Using this paradigm and relying on a shared ontology, different teams respectively developed in parallel the agents requiring their specific expertise (knowledge modeling, machine learning , etc.). The prototype functionalities developed were:

- A graphical user interface enabling logging, consultation of pushed documents, user profile edition and queries and new annotations formulation. Queries and annotations are formulated manually but the GUI assists the use of an ontology.
- A machine learning algorithm sorting the results by analyzing users' feedback.
- A mechanism pushing new annotations to potentially interested users; information resources are suggested by comparing new annotations and user profiles.
- Distributed algorithms to allocate new annotations and solve distributed queries.

The prototype was evaluated by end-users from a telecom company (T-Nova Deutsch Telekom) and a construction research center (CSTB) through two trials (at the eighth month and the twenty second month) during the two-year project. The very last prototype was presented and discussed during an open day at the end of the twenty third month.

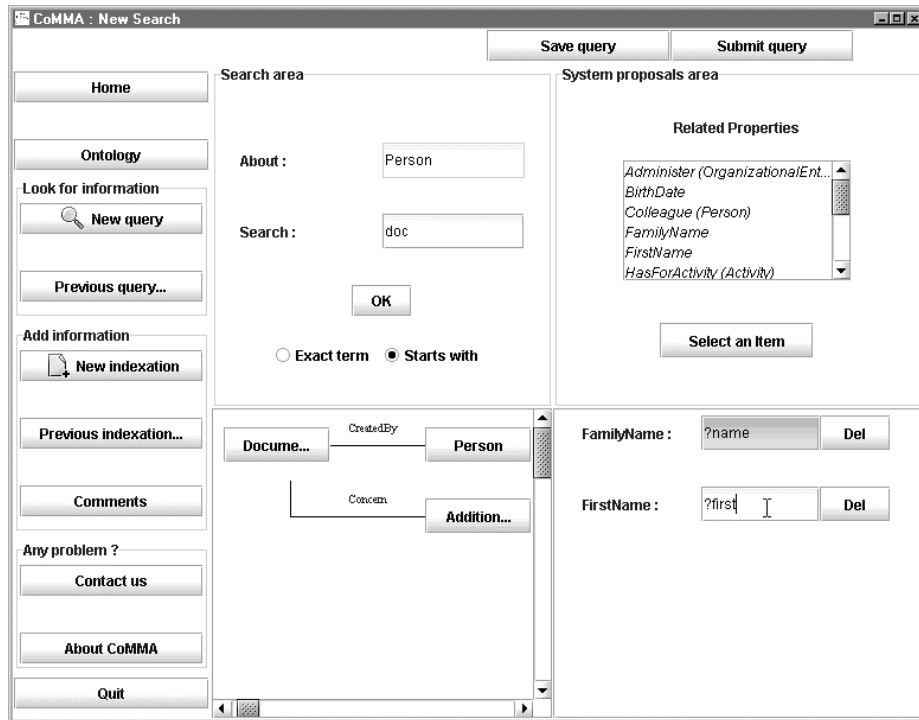
Since the MAS architecture is used to integrate a lot of different components that are vital to the system (GUI, CG search engine, XSLT engine, machine learning algorithm, etc.) if one of them goes wrong, the whole system evaluation may be hampered. Indeed, it is extremely hard to evaluate a component independently from the other components on which it may relies. A consequence is, for instance, that if there is a problem at the interface level, it may hamper a good evaluation of the information retrieval capabilities as a whole.

The first trial showed that the system meets both group and individual needs (*usefulness*) but the interfaces were not user-friendly (*usability*). The reason was that first interfaces were built for designers and knowledge engineers to test the integration, and not for end-users. As a result, the users could not have a clear view of the functionalities of the system. Interfaces were completely reengineered for the second trial.

For the second trial, the evaluation was prepared by a series of iterative evaluations with users-as-designers (*i.e.* end-users who participated directly to the re-design of the interfaces) in a design-and-test cycle. Results clearly showed that the CoMMA system was not only still *useful* (its functionalities were accepted by users), but also now *usable*: the GUIs being less complex (figure 6), users accepted them, and were not reluctant to manipulate them.

The fact that annotations were generated manually was not a problem in our scenarios, however this is not the case in general. To semi-automate the annotation process, we are studying two approaches:

- the use of natural language processing tools as in the SAMOVAR project [19].
- the use of wrappers for semi-structured sources e.g. web pages with a recurrent structure.



**Fig. 6.** Query Interface in CoMMA.

Both evaluations were "small-scale" evaluations: small number of users, small number of annotations (about 1000), and small duration of use. This short-period of use did not allow us to observe searching, indexing, and learning phenomena. Larger focused trials are being envisaged.

Trials also showed an effective specialization of the content of the annotation archives. One important point underlined by the first results is that the choice of the specialization of the archives content must be very well studied to avoid unwanted imbalance archives. This study could be done together with the knowledge engineering analysis carried out for the ontology building. It would also be interesting to extend the pseudo-distances to take into account the number of triples present in the archives to balance their sizes when choosing among close bids. We witnessed a noticeable reduction of the number of messages exchanged for query solving - compared to a simple multicast - while enabling fragmented results to be found. However a new algorithm exploiting additional heuristics and decomposition techniques is being studied to further reduce the number of messages exchanged for solving.

From the end-users point of view, the final system was both a real proof of concept and a demonstrator. It is not a commercial tool, but it did play its role in diffusing research results and convincing new partners to consider the MAS solution for distributed knowledge-based systems. From the developer point of view, the ontology-

oriented and agent-oriented approach was appreciated because it supported specification and distribution of implementation while smoothing the integration phase. The modularity of the MAS was appreciated both at development and trial time. During the development, the loosely-coupled nature of the agents enabled us to integrate changes in specifications and contain their repercussions.

Thus, CoMMA was a convincing experience to show that an approach based on knowledge engineering (formalizing knowledge about resources of an intraweb through semantic annotations based on an ontology) and distributed artificial intelligence (multi-agents information system loosely coupled by a cooperation based on semantic messages exchanges) can provide a powerful paradigm to solve complex distributed problems such as organizational memory management.

## Acknowledgements

We thank our colleagues of ACACIA and CoMMA (IST-1999-12217) for our discussions, and the European Commission that funded the project.

## References

1. J.L. Aguirre, R. Brena, F. Cantu-Ortiz, (2000). Multiagent-based Knowledge Networks. To appear in the special issue on Knowledge Management of the journal *Expert Systems with Applications*.
2. Y. Arens, C.A. Knoblock, W. Shen. Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems*, 6(2):99-130, 1996.
3. F. Bellifemine, A. Poggi, G. Rimassa, Developing multi agent systems with a FIPA-compliant agent framework. *Software Practice & Experience*, (2001) 31:103-128
4. B. Berner, and E. Ferneley, (1999), "CASIMIR: Information Retrieval Based on Collaborative User Profiling", In *Proceedings of PAAM'99*, pp. 41-56. [www.casimir.net](http://www.casimir.net)
5. C. Bothorel, and H. Thomas, (1999), "A Distributed Agent Based-Platform for Internet User Communities", In *Proceedings of PAAM'99*, Lancashire, pp. 23-40.
6. D. Brickley, R. Guha, Resource Description Framework Schema Specification 1.0, W3C Candidate Recommendation 27 March 2000
7. C. Collet, M. N. Huhns, and W. Shen. Resource integration using a large knowledge base in CARNOT. *IEEE Computer*, pages 55-62, December 1991
8. O. Corby, R. Dieng, C. Hébert, A Conceptual Graph Model for W3C Resource Description Framework. In *Proc. ICCS'2000 Darmstadt Germany*
9. K. Decker, K.P. Sycara., Intelligent adaptive information agents. *Journal of Intelligent Information Systems*, 9(3):239-260, 1997.
10. R. Dieng, O. Corby, A. Giboin, M. Ribière. Methods and Tools for Corporate Knowledge Management. In S. Decker and F. Maurer eds, *IJHCS special issue on knowledge Management*, vol. 51, pp. 567-598, Academic Press, 1999.
11. P. Doyle, B. Hayes-Roth, Agents in Annotated Worlds, In *Proc. Autonomous Agents*, ACM Press / ACM SIGART, Minneapolis, MN USA (1998) p173-180
12. M. Džbor, J. Paralic and M. Paralic, Knowledge Management in a Distributed Organisation, In *Proc. of the BASYS'2000 - 4th IEEE/IFIP International Conference on Information Technology for Balanced Automation Systems in Manufacturing*, Kluwer Academic Publishers, London, September 2000, ISBN 0-7923-7958-6, pp. 339-348

13. L. van Elst, A. Abecker, Domain Ontology Agents in Distributed Organizational Memories  
In Proc. Workshop on Knowledge Management and Organizational Memories, IJCAI, 2001.
14. Foundation for Intelligent Physical Agents, FIPA Specifications, 2001
15. F. Gandon, A Multi-Agent Architecture for Distributed Corporate Memories, Proc. 16th  
European Meeting on Cybernetics and Systems Research (EMCSR 2002) April 3 - 5, 2002,  
Vienna, Austria, pp 623-628.
16. F. Gandon, A Multi-Agent Platform for a Corporate Semantic Web, to appear in Proc.  
AAMAS 2002.
17. F. Gandon, Engineering an Ontology for a Multi-Agents Corporate Memory System, In  
Proc. ISMICK'01, Université de Technologie de Compiègne, p209-228.
18. M. Genesereth, A. Keller, O. Duschka, Infomaster: An Information Integration System, in  
proceedings of 1997 ACM SIGMOD Conference, May 1997.
19. J. Golebiowska, R Dieng-Kuntz, O. Corby, D. Mousseau, (2001) Building and Exploiting  
Ontologies for an Automobile Project Memory, First International Conference on Knowl-  
edge Capture (K-CAP), Victoria, October 23-24.
20. A. Kiss, J. Quinqueton, Multiagent Cooperative Learning of User Preferences, Proc. of  
European CMLP & PKDD, 2001.
21. M. Klush, *Intelligent Information Agent: Agent-Based Information Discovery and Man-  
agement on the Internet*, Springer, pages IX-XVII, 1999
22. O. Lassila, R. Swick, Resource Description Framework (RDF) Model and Syntax Specifi-  
cation, W3C Recommendation 22 February 1999
23. A.Y. Levy, D. Srivastava, and T. Kirk, Data model and query evaluation in global informa-  
tion systems. *Journal of Intelligent Information Systems* 5(2):121-143, September 1995
24. E. Mena, V. Kashyap, A. Sheth and A. Illarramendi, "OBSERVER: An approach for Query  
Processing in Global Information Systems based on Interoperation across Pre-existing On-  
tologies," Proceedings of the 1st IFCIS International Conference on Cooperative Informa-  
tion Systems (CoopIS '96), Brussels, Belgium, June 1996
25. M. Nodine, J. Fowler, T. Ksiezyk, B. Perry, M. Taylor, A. Unruh, Active Information  
Gathering In Infosleuth™; In Proc. Internat. Symposium on Cooperative Database Systems  
for Advanced Applications, 1999
26. J.B. Odubiyi, D.J. Kocur, S.M., Weinstein, N. Wakim, S. Srivastava, C. Gokey, J. Graham.  
SAIRE – A Scalable Agent-Based Information Retrieval Engine, Proc. Autonomous Agents  
97, Marina Del Rey, CA.
27. M.T. Özsu, P. Valduriez, Principles of Distributed Database Systems, 2nd edn., Prentice-  
Hall, 1999
28. Rolstadås, Development trends to support Enterprise Modeling, in Rolstadås and Andersen  
*Enterprise Modeling: Improving Global Industrial Competitiveness*, Kluwer, p3-16, 2000
29. J.F. Sowa, Conceptual Structures: Information Processing in Mind and Machine, Addison-  
Wesley, 1984.
30. L. Steels, Corporate Knowledge Management, *Proc. ISMICK 1993*, pp. 9-30
31. P.C. Weinstein, W.P. Birmingham, E.H. Durfee. Agent-Based Digital Libraries: Decen-  
tralization and Coordination. *IEEE Communication Magazine*, pp. 110-115, 1999