A multi-agent platform for a corporate semantic web

Fabien Gandon, Laurent Berthelot, Rose Dieng-Kuntz

A Multi-Agent Platform for a Corporate Semantic Web

Fabien Gandon
INRIA - ACACIA Project
2004, route des Lucioles - B.P.93
06902 Sophia Antipolis FRANCE
[+33] (0)4 92 38 77 88
fgandon@sophia.inria.fr

Laurent Berthelot
INRIA - ACACIA Project
2004, route des Lucioles - B.P.93
06902 Sophia Antipolis FRANCE
[+33] (0)4 92 38 50 23
lberthel@sophia.inria.fr

Rose Dieng-Kuntz
INRIA - ACACIA Project
2004, route des Lucioles - B.P.93
06902 Sophia Antipolis FRANCE
[+33] (0)4 92 38 78 10
dieng@sophia.inria.fr

ABSTRACT
We describe the technical choices and the design of a multi-agents software architecture to manage a corporate memory in the form of a corporate semantic web. We then present our approach to tackle a distributed memory and distributed queries.

Categories and Subject Descriptors
I.2.11 [**Computing Methodologies**]: Distributed Artificial Intelligence – *Multiagent systems, Intelligent agents*

H.3.3 [**Information Systems**]: Information Search and Retrieval – *Search process, Selection process*

General Terms
Design, Algorithms, Management.

Keywords
Distributed Knowledge Management, MAS Architecture, Ontology, Semantic Web.

1. INTRODUCTION
The information technology explosion of the last decade led to a shift in the economy and market rules. Corporations had to adapt their organization and management to improve their reaction and adaptation time. Information systems became backbones of organizations enabling project-oriented management and virtual teams. Thus the industrial interest in methodologies and tools enabling capitalization and management of corporate knowledge grew stronger. This article describes some work carried out by our team in the CoMMA project, investigating the use of several emerging technologies to support corporate memory management and in particular advantages of the multi-agents paradigm to design a management framework for a corporate semantic web. The first part will briefly describe the problematics of distributed corporate memories and justify our choices. The second part summarizes the design rationale of the architecture of our system. The last part focuses on our current work on the problems caused by the distribution of the annotations structuring the corporate semantic web.

2. DISTRIBUTED HETEROGENEOUS CORPORATE MEMORIES
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 [7]. The stake in building a corporate memory management system is the coherent integration of this knowledge dispersed in a corporation with the objective to promote knowledge growth, knowledge communication and in general preserve knowledge within an organization [20].

Our research team is part of the European project CoMMA aiming at implementing a corporate memory management framework based on several emerging technologies: agents, ontology and knowledge engineering, XML, information retrieval and machine learning techniques. The project intends to implement this system in the context of two scenarios: (1) assisting the insertion of new employees in the company and (2) supporting the technology monitoring process. The technical choices of CoMMA are mainly motivated by three observations:

(1) A corporate memory is, by nature, an heterogeneous and distributed information landscape. Corporate memories are now facing the same problem of information retrieval and overload as the Web. The initiative of a semantic Web [3] is a promising approach where semantics of documents is made explicit through ontology-based annotations to guide later exploitation. XML being likely to become an industry standard for exchanging data, we use it to build and structure the corporate memory. The Resource Description Framework (RDF) with its XML syntax allows us to semantically annotate resources of a corporate memory and envisage it as a *corporate semantic Web*.

(2) The population of users of the memory is, by nature, heterogeneous and distributed in the corporation. Some agents can then be dedicated to interface users with the system. Adaptation and customization are a keystone here and CoMMA relies on machine learning techniques in order to make agents adaptive to users and context.

(3) Tasks to be performed on corporate memories are, by nature, distributed and heterogeneous. Both the corporate memory and its population of users are distributed and heterogeneous. Therefore, it seems interesting that the interface between these two worlds be itself heterogeneous and distributed. Programming progresses were achieved through higher abstraction enabling us to model systems more and more complex. Multi-agents systems (MAS) are a new stage in abstraction that can be used to understand, to model and to develop a whole new class of distributed systems [21]. MAS paradigm is well suited for designing software architectures to be deployed above distributed information

landscapes: on the one hand, individual agents locally adapt to users and resources they are dedicated to ; on the other hand, cooperating agents enable the whole system to capitalize an integrated view of the corporate memory.

3. OVERVIEW OF CoMMA

In this part we present the concepts and the models behind the CoMMA system. We then proceed with the design process and final architecture of the system, starting from the societal level down to the individual behaviors.

3.1 Agents in an Annotated Memory

The article "Agents in Annotated Worlds" [8] shows that "annotated environments containing explanations of the purpose and uses of spaces and activities allow agents to quickly become intelligent actors in those spaces". This remark is transposable to information agents in complex information worlds: *annotated information worlds are, in the actual state of the art, a quick way to make information agents smarter*. If a corporate memory becomes an annotated world, agents can use the semantics of annotations and through inferences help users exploit its content.

RDF [16] uses a simple triple model and an XML syntax to represent properties of Web resources and their relationships. It makes no assumption about a particular application domain. *With RDF, we describe the content of documents through semantic annotations* and then use and infer from these annotations to successfully search the mass of information of the corporate memory. Just as an important feature of multi-agent systems is the ability to integrate legacy systems, an important feature of a corporate memory management framework is the ability to integrate the legacy archives. An RDF annotation being either internal or external to the resources, existing documents may be kept intact and annotated externally.

Compared to the Web, a corporate memory has more delimited and defined context, infrastructure and scope: the corporation. In a corporate context we can more precisely identify stakeholders and the corporate community shares some common global views of the world. Thus an ontological commitment is conceivable to a certain extent. We proposed and tested a methodology to build *O'CoMMA* (Ontology of CoMMA) [12] on which is based the descriptions of the organizational state of affairs, of the users' profile and the annotations of the memory resources. O'CoMMA is formalized and shared thanks to RDF Schema (RDFS) [5] which is related to object models but with the properties being defined separately. Figure 1 shows a sample of RDF(S) : the formalization of a hierarchy of concepts and properties and an example of annotation with literal and conceptual properties. Current keyword-based search engines are limited to terms denoting extensions of concepts. The introduction of ontologies enables agents to access the intensional level. O'CoMMA is the keystone of our system: it is a full resource of the memory and it provides the building blocks for models, annotations and agent messages, with their associated semantics. To manipulate and infer from the ontology and annotations our team developed CORESE [6] a prototype of a search engine enabling inferences on RDF by using the query and inference mechanisms available in the Conceptual Graphs formalism.

An *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, but lately enterprises realized that enterprise models have a role to play in their information system also. In CoMMA, the corporate model gives the system an insight in the organizational context and environment to tune its interactions and reactions. It is materialized as RDF annotations about the organization.

Likewise, the *users' profile* captures all aspects of the user that were identified as relevant for the system behavior. It contains administrative information and preferences that go from interface customization to topic interests. It positions the user in the organization: role, location and potential acquaintance network. In addition to explicitly stated information, the system derives information from past usage by collecting the history of visited documents and possible feedback from the user. From this, agents learn some of the user's habits and preferences [15]. These learnt criterions are used for interfaces or information push.

Figure 1. RDF(S) Sample

Unlike a lot of other projects (e.g. InfoSleuth [17]), CoMMA does not stress the heterogeneous sources reconciliation aspect ; documents are heterogeneous but annotations are represented in RDF and based on a shared ontology. It is not a digital library

project (e.g. SAIRE [18]) since we do not deal directly with electronic documents but with their annotations to support knowledge management inside an organization. While CoMMA does not focus on collaborative profiling/filtering (e.g. CASMIR [4]) either, it provides an architecture of cooperating agents, being able to adapt to the user, and supporting information distribution in an organization. The duality of the definition of 'distribution' reveals two important problems to be addressed: (1) 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. (2) Distribution also means the action of 'distributing or spreading or apportioning' ; the problem here is how to make the relevant pieces of information go to the concerned (artificial or human) agent. It is with both purposes in mind that we designed the CoMMA architecture as presented in the following section.

3.2 Architecture

Information agents are part of the intelligent agents. A MAS is a loosely coupled network of agents that work together as a society aiming at solving problems that would generally be beyond the reach of any individual agent. A MAS is heterogeneous when it includes agents of at least two types. A Multi-Agents Information System (MAIS) is a MAS aiming at providing some or full range of functionality for managing and exploiting information resources. The application of MAIS to corporate memories means that agents' cooperation aims at enhancing information capitalization in the company. *The CoMMA software architecture is an heterogeneous MAIS*.

The MAIS *architecture is a structure that portrays the different families of agents and their relationships*. 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. In the case of a corporate memory, a given configuration is tightly linked to the topography and context of the place where it is deployed (organizational layout, network topography, stakeholders location), therefore it must adapt to this information landscape and change with it. The architecture must be designed so that the set of possible configurations covers the different corporate organizational layouts foreseeable. The configuration description is studied and documented at deployment time using adapted UML deployment diagrams to represent, hosts (servers, front-end...), MAS platforms, agent instances and their acquaintance graph. The architectural description is studied and fixed at design time. The architectural analysis starts from the highest level of abstraction (*i.e.* the society) and by successive refinements (*i.e.* nested sub-societies), it goes down to the point where the needed agent roles and interactions can be identified.

Our approach to design the CoMMA architecture shares with the A.G.R. model used in AALAADIN [10] and GAIA [21] methodologies the concern for an organizational approach where the MAS architecture is tackled, as in a human society, in terms of roles and relationships. The functional requirements of the system do not simply map to some agent functionalities but influence and are finally diluted in the dynamic social interactions of individual agents and the set of abilities, roles and behaviors attached to them. Considering the system functionality, we identified four dedicated sub-societies of agents as shown in figure 2 : (1) Sub-society dedicated to ontology and model (2) annotations-dedicated sub-society (3) User-dedicated sub-society (4) Connection-dedicated sub-society.

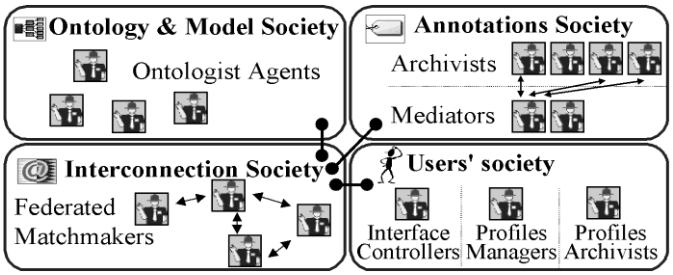

Figure 2. Sub-societies of CoMMA

Analyzing the resource-dedicated sub-societies (ontology, annotations and yellow pages for interconnection), we found that there was a recurrent set of possible organizations for these sub-societies: *hierarchical*, *peer-to-peer*, and *replication*. As discussed in [14] every organization has advantages and disadvantages ; depending on the type of tasks to be performed, the size and complexity of the resources manipulated, a sub-society organization will be preferred to another.

The agents from the *sub-society dedicated to the ontology and model* are concerned with the ontology and model exploitation during information retrieval activities and especially the queries about the hierarchy of concepts and the description of the organization where the system was deployed. Thus, they provide downloads, updates and querying mechanisms for other agents. For this sub-society, the three types of organizations are conceivable. CoMMA implemented a replication society where each agent have a complete copy of the ontology/model and can resolve queries by itself. It is acceptable since in our prototype the ontological commitment is centralized and the global ontology is updated and propagated over the agent society. Other options are interesting if the ontology/model is large or changes quite often and if a distributed mechanism in the MAS must support the consensus process as in FRODO [9].

The agents from the *annotation dedicated sub-society* are concerned with the exploitation of annotations structuring the corporate memory, they search and retrieve references matching users' queries. Here, only the hierarchical or the peer-to-peer society are conceivable: a replication society is not realistic since it would imply to replicate a full copy of the corporate memory for each resource agent. As we will detail in section 4, CoMMA implemented a hierarchical organization.

The agents from the *connection dedicated sub-society* are in charge of the matchmaking of the other agents based upon their respective needs and roles descriptions. CoMMA is implemented with JADE [1], an open source MAS platform compliant with the FIPA [11] specifications, that provides a Directory Facilitator Agent type. These agents are federable matchmakers organized in a peer-to-peer society managing the Yellow Pages.

The agents from the *user dedicated sub-society* are concerned with the interface, the monitoring, the assistance and the adaptation to the user. Because they are not related to a resource type like the previous ones, they cannot be studied using the typology we defined. We distinguished two recurrent roles in this type of sub-society: (1) the user interface management: to

dialogue with the users to enable them to express their request, to refine them and to present results in a comprehensive format (2) the management of user's profile: to archive and make the profiles available to other agents. More details are given in [13].

3.3 Roles, interactions and behaviors

From the architecture analysis we can derive the characteristics of the identified roles, their interactions and finally we implement the corresponding behaviors in a set of agent types.

Roles represent the position of an agent in a society and the responsibilities and activities assigned to this position and expected by others to be fulfilled. In the design junction between the micro-level of agents and the macro-level of the MAS, the role analysis is a key step. The previous part identified the following roles which characteristics are detailed in [14]:

- *Ontology Archivist*: maintains and accesses the ontology.
- *Enterprise Model Archivist*: maintains and accesses the enterprise model.
- *Annotation Archivist*: maintains and accesses an annotation repository.
- *Annotation Mediator*: manages and mediates among a set of Annotation Archivists.
- *Directory Facilitator*: maintains and accesses the yellow pages.
- *Interface Controller*: manages and monitors a user interface.
- *User Profile Manager*: manages updates of profiles of users logged nearby.
- *User Profile Archivist*: stores and retrieves users' profiles.

Following the role identification comes the specification of role *interactions*. Interactions consist in more than the sending of an isolated message. The conversation pattern needs to be specified with protocols and the agents must follow them for the MAS to work properly. Protocols are codes of correct behavior in a society for agents to interact with others. They describe a standard procedure to regulate information transmission between agents and institutionalize patterns of communication occurring between identified roles. The definition of a protocol starts with an acquaintance graph at role level, that is a directed graph identifying communication pathways between agents playing the considered roles. From that, we specify the possible sequences of messages. The acquaintance connections among the roles and the protocols adopted derive from both the organizational analysis and the use cases dictated by the application scenarios. The acquaintance graphs and the ACL message traces are depicted [14] using protocol diagrams [2], a restriction of the UML sequence diagrams, proposed within the AUML[1] initiative.

From the role and interaction descriptions the different partners of CoMMA proposed and implemented agent types that fulfill one or more roles. The *behavior* of an agent type combines behaviors implemented by the designers to accomplish the activities corresponding to the assigned roles. The behaviors come from the implementation choices determining the responses, actions and reactions of the agent. The implementation of the behavior is subject to the toolbox of technical abilities available to the designers, for instance, modules of the CORESE [6] search engine have been integrated in the behavior of the agents dedicated to the ontology, the models and the annotations.

[1] Agent Unified Modelling Language http://www.auml.org.

The implementation of CoMMA relying on JADE, the agent communication language is FIPA ACL, based on the speech act theory, which comes with standard protocols to be used or extended. Messages are encoded en RDF.

4. ANNOTATION DISTRIBUTION

The submissions of queries and annotations are generated by agents from the user-dedicated society and routed to the annotation-dedicated society. The latter is a hierarchical society: the agents playing the Annotation Mediator role (AM) are in charge of managing agents playing the Annotation Archivist role (AA). The AM provides its services to other societies handling distributed query solving and allocation of new annotations to 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. The agents playing the role of AA and AM are benevolent and, once deployed, temporally continuous. After presenting the problematics of distribution, we shall see how the ontology and the Semantic Web frameworks can support these agents in their tasks.

4.1 Problematics of distribution

Distributed Databases field [19] 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 $site_1$ will have reports with a property 'title' ranging from "Criminality in agent societies" to "MAS control" and $site_2$ will have reports from "Naive agents" to "Zeno paradox". *Vertical fragmentation* means that information is split according to types of concepts and properties, for instance $site_1$ will have reports with their titles and authors and $site_2$ will have articles with their abstract and keywords. Fragmentation choices are made by 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 solving is closely related to the choices made for the distribution of annotations.

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, on the population of triples of 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 specialised 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 $PropertyType_y$. 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 specialised 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. It 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 specialise individual agents according to the topography of the company network (e.g. 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 specialisation of these properties is then considered to be part of the preferences of the AA and can be used for bidding.

4.2 Allocating newly submitted annotation

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 [11] with the AAs. The AM measures how close the new annotation is from the ABIS and CAP of the candidate AAs to decide which one of them should win the bid. We detail the pseudo-distance calculation.

Step (1): definition of constants used for distances

$$\mathrm{Max_L} = 256 \quad (1)$$

is the maximum range for the ASCII byte code of a character.

Max_C and Max_R are the maximum path length in the subsumption hierarchy of respectively the primitive concept types and the conceptual relation types from the root to a leaf.

$$\mathrm{N} = \mathrm{Max_C} \times 2 \,/\, \mathrm{Max_L} \quad (2)$$

is the constant used to normalise the distances when combining distances on literal and distances on primitive types. $\mathrm{Max_C} \times 2$ is an upper bound for the length of two subsumption paths.

$$\mathrm{W_C} = 4 \qquad \mathrm{W_R} = 8 \qquad \mathrm{W_L} = 1 \quad (3)$$

are weights respectively for concept types, conceptual relation types and literals. They are used to balance the importance of these factors in the pseudo-distance calculations.

Step (2): distance between two literals

Some properties have a literal range type that we need to compare. Let $C_{X,i}$ the ASCII byte code of the i^{th} character in upper case ($C_{X,i} \in [0,\mathrm{Max_L}[$) of $Lit_X = C_{X,0}, C_{X,1}, C_{X,2}, C_{X,3}, \ldots, C_{X,s}$ a literal coded by the sequence of length $s+1$ of the codes of its characters. Let :

$$\mathrm{Abscissa}(\mathrm{Lit_X}) = \sum_{i=0..s} \frac{C_{x,i}}{\mathrm{Max_L}^{\,i}} \geq \sum_{i=0..s} \frac{0}{\mathrm{Max_L}^{\,i}} = 0 \quad (4)$$

This Abscissa is positive or null and bounded by B :

$$\mathrm{B} = \sum_{i=0..s} \frac{\mathrm{Max_L} - 1}{\mathrm{Max_L}^{\,i}} = (\mathrm{Max_L} - 1) \cdot \sum_{i=0..s} \frac{1}{\mathrm{Max_L}^{\,i}} \quad (5)$$

The sum of the finite geometric sequence is itself bounded.

$$B = (\mathrm{Max_L} - 1) \cdot \frac{1 - (1/\mathrm{Max_L})^{s+1}}{1 - (1/\mathrm{Max_L})} = \mathrm{Max_L} - (1/\mathrm{Max_L})^{s} < \mathrm{Max_L} \quad (6)$$

Thus $Abscissa(Lit_X) \in [0, Max_L[$, which explains the value of N.

We now consider the difference :

$$D = \mathrm{Abscissa}(\mathrm{Lit_B}) - \mathrm{Abscissa}(\mathrm{Lit_A}) \quad (7)$$

where Lit_A and Lit_B are two literals. If they are the same, their abscissas are equal and $D=0$. Else, let Lit_A come before Lit_B in alphabetical order. This means the first difference in reading these strings is a character in Lit_A that comes alphabetically before the character at the same position in Lit_B. This can be formalized as : If $Lit_A < Lit_B$ then $\exists i \in [0..s]$ such that $\forall j < i \; C_{A,j} = C_{B,j}$ (*i.e.* the strings may have a common beginning) and $C_{A,i} < C_{B,i}$ (*i.e.* the first difference is in alphabetical order). The value of D (complementing the shortest string with characters of code 0 if necessary, so that both strings have the same length) is given by :

$$D = \sum_{k=0..s} \frac{C_{B,k} - C_{A,k}}{\mathrm{Max_L}^{\,k}} = \frac{C_{B,i} - C_{A,i}}{\mathrm{Max_L}^{\,i}} + \sum_{k=i+1..s} \frac{C_{B,k} - C_{A,k}}{\mathrm{Max_L}^{\,k}} \quad (8)$$

The absolute value of the remaining sigma sum is bounded by:

$$B' = \sum_{k=i+1..s} \frac{\mathrm{Max_L} - 1}{\mathrm{Max_L}^{\,k}} = \frac{\mathrm{Max_L} - 1}{\mathrm{Max_L}^{\,i+1}} \cdot \sum_{k=0..s-i-1} \frac{1}{\mathrm{Max_L}^{\,k}} < \frac{1}{\mathrm{Max_L}^{\,i}} \quad (9)$$

Therefore as $C_{A,j} < C_{B,j}$ we proved that:

$$\frac{C_{B,i} - C_{A,i}}{\mathrm{Max_L}^{\,i}} \geq \frac{1}{\mathrm{Max_L}^{\,i}} > \left| \sum_{k=i+1..s} \frac{C_{B,k} - C_{A,k}}{\mathrm{Max_L}^{\,k}} \right| \quad (10)$$

And we can conclude that $D>0$ *iff* $Lit_A < Lit_B$, thus:

$$\mathrm{Lit_A} < \mathrm{Lit_B} \Leftrightarrow \mathrm{Abscissa}(\mathrm{Lit_A}) < \mathrm{Abscissa}(\mathrm{Lit_B}) \quad (11)$$

Based on the abscissa we define an Euclidean distance:

$$\mathrm{Dist_L}(\mathrm{Lit_A}, \mathrm{Lit_B}) = |\, \mathrm{Abscissa}(\mathrm{Lit_B}) - \mathrm{Abscissa}(\mathrm{Lit_A}) \,| \quad (12)$$

As an example, if Lit_A = *"abandon"* and Lit_B = *"accent"* then:

- $\mathrm{Abscissa}(\mathrm{Lit_A}) \sim 65.25880898635597$
- $\mathrm{Abscissa}(\mathrm{Lit_B}) \sim 65.26274521982486$
- $\mathrm{Dist_L}(\mathrm{Lit_A}, \mathrm{Lit_B}) \sim 0.0039362334688917144$

Step (3): pseudo-distance literal - literal interval

The ABIS and CAP provide bounding interval for literal properties. We define a pseudo-distance between a literal value Lit_X from an annotation and a range $[B_{low}, B_{up}]$:

$$\begin{aligned} &\mathrm{Dist_l}(\mathrm{Lit_X}, [\mathrm{B_{low}}, \mathrm{B_{up}}]) = 0 \text{ if } \mathrm{Lit_X} \in [\mathrm{B_{low}}, \mathrm{B_{up}}] \\ &\text{else} = \mathrm{Min}(\mathrm{Dist_L}(\mathrm{Lit_X}, \mathrm{B_{low}}), \mathrm{Dist_L}(\mathrm{Lit_X}, \mathrm{B_{up}})) \end{aligned} \quad (13)$$

This is only a pseudo-distance since it is not an application from Literal×Literal to $\Re^+$ but from Literal×[Literal,Literal] to $\Re^+$.

Step (4): distance between two ontological types

To compare two primitive types, we use the length, in number of edges, of *the shortest path between these types in the hierarchies of their supertypes*. The calculation of this distance is a problem equivalent to searching the least common supertype and the two distances from this supertype to the considered types:

$$\mathrm{Dist_H}(\mathrm{T_1}, \mathrm{T_2}) = \mathrm{SubPath}(\mathrm{T_1}, \mathrm{LCST}) + \mathrm{SubPath}(\mathrm{T_2}, \mathrm{LCST}) \quad (14)$$

where *LCST* is the Least Common SuperType of T_1 and T_2 and *SubPath(T,ST)* is the length, of the subsumption path from a type T to one of its supertype ST.

This distance measures a semantic closeness since the least common supertype of two types captures what these types have in common. $\mathrm{Dist_H}$ complies to the four features of distances:

$$\mathrm{Dist_H}(\mathrm{T_1}, \mathrm{T_1}) = 0 \quad (15)$$

since the least common supertype of (T_1,T_1) is T_1 and the shortest path to go from a node to itself is not to cross an arc.

$$Dist_H(T_1,T_2) = Dist_H(T_2,T_1) \quad (16)$$

since the considered path is not directed and therefore the shortest path from T_1 to T_2 is also the shortest path from T_2 to T_1.

$$Dist_H(T_1,T_2) = 0 \Rightarrow T_1=T_2 \quad (17)$$

since the only way to have a null distance is not to cross an arc which is only possible if the two nodes are merged.

$$Dist_H(T_1,T_3) \leq Dist_H(T_1,T_2) + Dist_H(T_2,T_3) \quad (18)$$

it is proved *ad absurdio* : if $Dist_H(T_1,T_2) + Dist_H(T_2,T_3)$ was smaller than $Dist_H(T_1,T_3)$, it would mean that there exists a path $(T_1,...,T_2,...,T_3)$ shorter than the shortest path from T_1 to T_3 which is absurd. Therefore $Dist_H$ is a distance.

Step (5): distance between a concept type and a literal
We decided that the distance between a primitive type and an arbitrary literal is a constant greater than any type distance. Let

$$Dist_{LC}(T_1,L_X) = (Max_C \times 2 + 1) \quad (19)$$

If one prefers to consider a literal as a basic type at the top of the hierarchy, then we could replace (19) by (20) :

$$Dist_{LC}(T_1,Lit_X) = Depth(T_1) + 1 \quad (20)$$

where $Depth(T_1)$ is the length of the shortest path from the root of the hierarchy to the primitive type T_1.

Step (6): pseudo-distance annotation triple - property family
Let $Triple_A = (T_{RA}, T_{A1}, T_{A2})$ a triple from an annotation and let $Triple_B = (T_{RB}, T_{B1}, T_{B2})$ a triple from the ABIS.

In an RDF triple, T_{A1} and T_{B1} are primitive concept types, let

$$D_{C1} = W_C \times Dist_H(T_{A1}, T_{B1}) \quad (21)$$

Now, considering the T_{A2} and T_{B2} :

- If both are primitive concept types then let :

$$D_{C2} = W_C \times Dist_H(T_{A2}, T_{B2}) \quad (22)$$

- If one is a primitive concept type T and the other is a literal L

$$D_{C2} = W_C \times Dist_{LC}(T,L) \quad (23)$$

- If both types are literals then from the ABIS we know $[B_{low}, B_{up}]$ and from the annotation we know the literal Lit_X. Let:

$$D_{C2} = W_L \times N \times Dist_I(Lit_X, [B_{low}, B_{up}]) \quad (24)$$

Finally we calculate the distance between the relation types, let

$$D_R = W_R \times Dist_H(T_{RA}, T_{RB}) \quad (25)$$

The final pseudo-distance between the annotation triple and a property family of the ABIS is given by:

$$Dist_{TFABIS}(Triple_A, Triple_B) = D_{C1} + D_R + D_{C2} \quad (26)$$

Step (7): pseudo-distance annotation triple - ABIS
The pseudo-distance between a triple and an ABIS is the minimal pseudo-distance between this triple and the ABIS triples.

$$Dist_{TABIS}(Triple,ABIS) = \min_{Triple_i \in ABIS}(Dist_{TFABIS}(Triple,Triple_i)) \quad (27)$$

Step (8): pseudo-distance annotation triple - CAP
The calculation of the pseudo-distance $Dist_{TCAP}(Triple,CAP)$ is the same as for the ABIS except for the primitive type distance: when comparing two triples, if the type of the annotation is a specialisation of the type of the triple from the CAP, the length of the path between them is set to *0*. This is to take into account the fact that the CAP captures preferences and that anything more precise (as a specialisation) is included in the preferences.

Step (9): pseudo-distance annotation - ABIS / CAP / AA
We sum the pseudo-distances for the triples of the annotation :

$$Dist_{AABIS}(An_X, ABIS) = \sum_{Triple_j \in An_X} Dist_{TABIS}(Triple_j, ABIS) \quad (28)$$

$$Dist_{ACAP}(An_X, CAP) = \sum_{Triple_j \in An_X} Dist_{TCAP}(Triple_j, CAP) \quad (29)$$

where An_X is an annotation.

Finally, we sum the pseudo-distances to ABIS and CAP:

$$Dist(An_X, AA_y) = Dist_{AABIS}(An_X, A_y) + Dist_{ACAP}(An_X, C_y) \quad (30)$$

where AA_y is an archivist agent, An_X is an annotation and A_y and C_y are the ABIS and CAP of agent AA_y. Using this pseudo-distance, the AM compares bids of AAs and allocates newly submitted annotations to the closest agent.

4.3 Query distribution

Query solving, involves several distributed annotation bases ; answers are a merging 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 [11] to allow multiple stages with subqueries being exchanged between the AM and the AAs. The decomposition algorithm consists of four stages: preprocessing for query simplification, constraints solving, questions answering and final filtering. These stages, detailed in the following subsections, manipulate the query structure through the Document Object Model (DOM[2]). It 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 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.

Step (1): query simplification
A preprocessing is done on the query before starting the decomposition algorithm. A query may hold co-references. Simple co-references (two occurrences of a variable where one reference is a node of a subtree of the query and the other one is the root of another subtree) are merged by grafting the second tree on the first one. Complex co-references would generate distributed constraint problems. They are erased and replaced by simple variables for the duration of the distributed solving ; they are then reintroduced at the last stage for the final filtering.

Step (2): constraints solving
To cut down the network load, the decomposition starts with the solving of constraints contained in the query - Figure 3. The grouping of constraints limits the number of messages being exchanged by constraining the queries as soon as possible.

We group constraints according to the concept instance used for their domain value. We choose a group of constraints among the

[2] The DOM is specified by the W3C - *http://www.w3.org/DOM/*

deepest ones and create a subquery by extracting it and asking for the possible URIs of its root concept. The aim is to replace this subconstraint in the global query by this list of possible URIs and iteratively reduce the depth of the global query.

Among the candidate AAs, agents concerned by this subquery are identified, thanks to the OBSIQ they provided at the start, and contacted to try to solve the subquery using their local resources:

- If a piece of RDF matches the query and provides the URI of its root concept, the AA sends it to the AM.
- If an annotation violates a constraint, it is dismissed.
- If an annotation answers partially, and if the root concept of the result has a URI, the AA returns the incomplete answer with the URI since the missing part can be found somewhere else thanks to this unique ID.
- If an annotation answers partially, but does not have a URI at the root concept (existential quantification), then the AA does not return it since it cannot be completed elsewhere.
- If an annotation answers the query but does not have a URI for the root concept, the AA returns the whole annotation.

Partial results are merged in the local base of the requesting AM. The AM then reduces the original query using the URIs it has learnt and generates a list of smaller queries. For each one of these queries, it applies this again until no constraint is left.

Step (3): Questions answering

After the constraint solving, the AM has, in its base, complete annotations and partial answers with URIs. Using these unique identifiers, it is now able to request the information asked by the user for each one of the resources identified. Therefore, after solving the constraints, the AM emits queries to fill the question fields. The AM starts from the root of the original query since its potential URIs should have been found by now. It generates a subquery each time it finds a question during its walk through the tree - Figure 3. Some URIs may still be missing for unconstrained nodes and intermediate queries may have to be issued to solve them. If the initial query was not constrained at all, there is no URI to constrain the root when starting the question solving. Thus the flow of data to solve it would potentially be too high and could result in a network jam. In that case, the AM switches to a degraded mode and asks the AAs to solve the whole query locally, potentially loosing some answers.

Step (4): Filtering final results

Once the questions have been answered, the AM projects the original query on the temporary base it has built and extracts the final correct full answers. This stage enables it to finalise the merging of partial results and to take into account the possible cross-references occurring in the constraints, that were discarded during the pre-processing. The AM can then send back the result to the external requester agent.

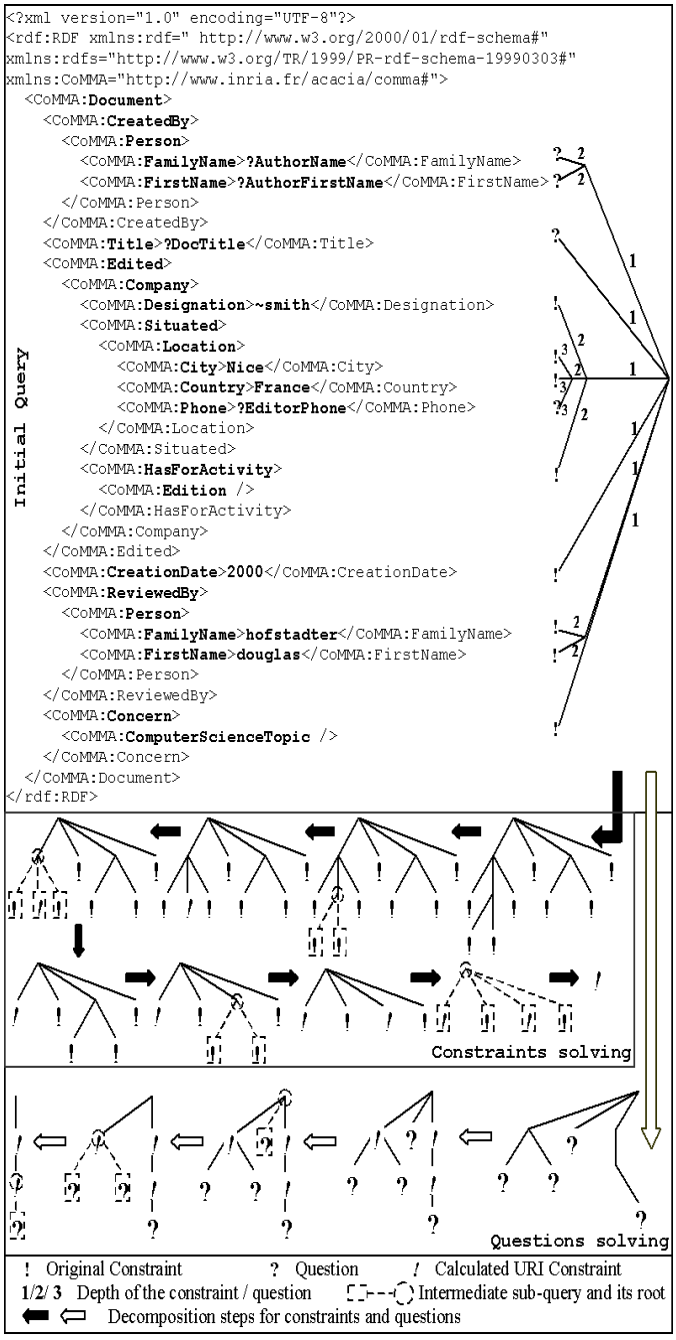

Figure 3. Query decomposition process

5. DICUSSION AND CONCLUSION

The prototype implemented in JAVA was evaluated by end-users from a telecom company (T-Nova System Deutsch Telekom) and a construction research center (CSTB) with archives containing up to 1000 annotations. Interface and ergonomics problems were raised by the users but the usefulness and the potential of the functionalities offered by the system were unanimously acknowledged. In particular, the ontology-oriented and agent-oriented approach were appreciated by the end-users for their powerfulness and by the developers for the new approach they support to specify and distribute implementation while smoothing the integration phase. Concerning the integration phase of the

development, the agent technology proved to be extremely valuable: the different agents have been developed by distant partners having the needed experience and starting from shallow agents ; but since the agents are loosely coupled software components and that their role and interactions have been specified using a consensual ontology, the integration and setup of a first prototype was achieved in less than two days. Concerning the annotation-dedicated sub-society, we showed how some aspects of the underlying graph model of the Semantic Web framework could be exploited to handle allocation of annotations and distributed query solving, in particular in a multi-agents system. The first tests on the prototype we implemented showed an effective specialization of the content of the annotation archives. We also 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. 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 building the ontology. 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. Weights and alternative algorithmic options will have to be tuned and compared to evaluate the performance of the overall system.

6. ACKNOWLEDGMENTS

We warmly thank our colleagues of the CoMMA consortium for our fruitful discussions and the European Commission that funds the CoMMA project (IST-1999-12217).

7. REFERENCES

[1] Bellifemine, Poggi, Rimassa, Developing multi- agent systems with a FIPA-compliant agent framework. Software Practice & Experience, 2001, 31:103-128

[2] Bergenti, Poggi, Exploiting UML in the Design of Multi-Agent Systems. Engineering Societies in the Agents World - LNAI, Springer, 2000, vol. 1972, 106-113.

[3] Berners-Lee, Hendler, Lassila, The Semantic Web, Scientific American, May 2001, 35-43.

[4] Berney, Ferneley, CASMIR: Information Retrieval Based on Collabo-rative User Profiling, In Proc. PAAM'99, 41-56.

[5] Brickley, Guha, Resource Description Framework (RDF) Schema Specification. W3C Candidate Recommendation, 27 March 2000 (http://www.w3.org/TR/2000/CR-rdf-schema-20000327/)

[6] Corby, Dieng, Hebert, A Conceptual Graph Model for W3C Resource Description Framework. In Proc. International Conference on Conceptual Structures, 2000, Springer LNAI 1927, 468-482

[7] Dieng, Corby, Giboin, Ribière. Methods and Tools for Corporate Knowledge Management. In Decker and Maurer, International Journal of Human-Computer Studies, special issue on knowledge Management, vol. 51, 567-598, Academic Press, 1999.

[8] Doyle, Hayes-Roth, Agents in Annotated Worlds, Proc. Autonomous Agents, ACM, 1998, 173-180

[9] Elst, Abecker, Domain Ontology Agents in Distributed Organizational Memories. Proc. Workshop on Knowledge Management and Organizational Memories, IJCAI 2001.

[10] Ferber, Gutknecht, A meta-model for the analysis and design of organizations in multi-agent systems. IEEE Computer Society, Proc. 3rd ICMAS, 128-135, 1998.

[11] Foundation for Intelligent Physical Agents, FIPA Specifications, 2001, (http://www.fipa.org/)

[12] Gandon, Engineering an Ontology for a Multi-Agents Corporate Memory System, Proc. International Symposium on the Management of Industrial and Corporate Knowledge 2001, 209-228.

[13] Gandon, Dieng, Corby, Giboin, A Multi-Agents System to Support Exploiting an XML-based Corporate Memory, Proc. Practical Application of Knowledge Management 2000, 10.1-10.12.

[14] Gandon, A Multi-Agent Architecture for Distributed Corporate Memories, From Agent Theory to Agent Implementation, at 16th EMCSR, 2002, 623-628.

[15] Kiss, Quinqueton, Multiagent Cooperative Learning of User Preferences, Proc. European Conference on Machine Learning Principles and Practice of Knowledge Discovery in Databases, 2001.

[16] Lassila, Swick, Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation, 22 February 1999, (http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/)

[17] Nodine, Fowler, Ksiezyk, Perry, Taylor, Unruh, Active Information Gathering In Infosleuth™; In Proc. Internat. Symposium on Cooperative Database Systems for Advanced Applications, 1999

[18] Odubiyi, Kocur, Weinstein, Wakim, Srivastava, Gokey, Graham. SAIRE – A Scalable Agent-Based Information Retrieval Engine, Proc. Autonomous Agents 97

[19] Özsu, Valduriez, Principles of Distributed Database Systems, 2nd ed., Prentice-Hall, 1999.

[20] Steels, Corporate Knowledge Management. Proc. International Symposium on the Management of Industrial and Corporate Knowledge, 1993, 9-30

[21] Wooldridge, Jennings, Kinny, The Gaia Methodology for Agent-Oriented Analysis and Design, Autonomous Agents and Multi-Agent Systems, 2000, 3(3):285-312.