Multi-agent Distributed Knowledge Management System

Pablo D. Villarreal, María L. Caliusco

GIDSATD- UTN Facultad Regional Santa Fe

Lavaisse 610, Santa Fe, Argentina, 3000

María R. Galli, Omar Chiotti

GIDSATD- UTN Facultad Regional Santa Fe

INGAR – CONICET

Avellaneda 3657, Santa Fe, Argentina, 3000

Abstract

The knowledge is the main asset of current organizations in order to be competitive in the new global market. Wherefore, a current challenge of organizations is to adequately manage its knowledge. The organization knowledge mainly resides in the people that integrate the organization, who are generally spatial and timely distributed. In this work we present a distributed knowledge management system designed to support the management of tacit knowledge that belongs to people of an organization. This system is based on the use of mobile agents, which receive the user's queries and visit the organization domains where this information can be generated. So, the proper system has to analyze the queries to define the domains that offer greatest possibilities of answering them. To make this decision it has an intelligent agent (ISLA) with a knowledge base where the knowledge about information managed by each domain is represented. Firstly we present the agent-based analysis and design, and the defined architecture of the knowledge management system. Secondly, we describe the architecture of ISLA, the main agent of this system.

Keywords: Knowledge Management, Multi-agent System, Decision Support System.

1 Introduction

The knowledge is the main asset of current organizations, not only to high technology enterprises but also to conventional productive systems, through the need of improving corporate work processes, product and service quality, and productivity; and reducing new product design time. The organization knowledge mainly resides in the people that integrate it, which are generally spatial and timely distributed. Wherefore, a current challenge of organizations is to adequately manage its knowledge. This activity is called knowledge management (KM). The aim of the knowledge management is not to solely manage the previous knowledge. The knowledge of the past, it is only valuable if it can proportionate a perspective of the future: the main usefulness of the knowledge management is its innovation support.

The challenges associated with knowledge management, can be classified into three general categories [4]: acquisition, organization, and distribution. Knowledge acquisition deals with the issues involved in knowledge extraction in its various forms. That is, from the organization's knowledge bases, databases, printed resources, and people. The knowledge acquisition implies to detect who are the people that have the knowledge on a specific area, how is the knowledge in that area currently stored, and how this knowledge can be made machine-readable. Knowledge organization deals with the issues about the best way of storing knowledge so that it can be retrieved when it is relevant. Knowledge distribution, must tackle the problem of getting the right knowledge to the right place at the right time.

Think about the availability of developing a system to support the KM requires analyzing the knowledge concept. To this aim, we define the knowledge as the information that is integrated and understood in the mind of a given subject. A review about several approaches to classify the knowledge and their relationships is presented in [10]. One of them distinguishes explicit and tacit knowledge [8]. Explicit knowledge is easily shared whereas tacit knowledge is highly personal. This last type of knowledge is not articulated and is mixed with emotions; it is the result of some internal processing. A part of the tacit knowledge, which implicitly belongs to somebody, can be made explicit, but there is a part of the tacit knowledge that definitely cannot be made explicit [1].

A characteristic of tacit knowledge is that it is task specific and it is related to ability [16]. That is, tacit knowledge gives some abilities that are dependent on the context of application. The knowledge is generated when information is combined with context and expertise [2]. This arise the question of the acquisition of the knowledge context and the validity of the tacit knowledge. The knowledge context may be defined as the collection of relevant conditions and environmental influences that make a situation unique and comprehensible. The context is primarily subjective and task oriented [10]. To sum up, part of the tacit knowledge and it context cannot be made explicit, then they cannot be automated.

An approach to support the knowledge management is based on developing a corporate or organizational memory. Several works are focused in this direction [13], [11]. An organizational memory is defined [15] as an explicit, disembodied, persistent representation of crucial knowledge and information in an organization, in order to facilitate their access, sharing, and reuse by members of the organization, for their individual or collective tasks.

A corporate memory is appropriate to represent the part of the tacit knowledge and its context that can be made explicit, but the other part of the tacit knowledge and its context, which belong to people, cannot be represented in the corporate memory. That is, in an organization there is information that can only be generated by people, then, when a decision maker in some time and somewhere of an organization requires this information, he/she has to solicit it to who has the ability of generating it. In this way, another type of support to facilitate the access to this knowledge is required.

In this work we present a multi agent system designed to support the management of tacit knowledge that belongs to people of an organization. This is a distributed knowledge management system based on the use of mobile agents, which receive the user's queries and visit the organization domains where this information can be generated. So, the proper system has to analyze the queries in order to define the domains that offer greatest possibilities of answering them. To make this decision, it has an intelligent agent (ISLA) with a knowledge base where the knowledge about information managed by each domain is represented. This agent has a knowledge retrieval component and a learning component that allows it a continuous improvement of the knowledge base and therefore of the knowledge management system operation. Firstly, we present the agent-based analysis and design of the knowledge management system, from which the architecture of the system is defined. Secondly, we describe the architecture of ISLA, the main agent of this system.

2 Analysis and Design of the Architecture of the Multi-Agent System

The research area of software agents is an emerging technology that supports the implementation of complex systems. As multi-agent technology begins to emerge as a viable solution for large-scale industrial and commercial applications, there is an increasing need to ensure that the systems being developed are robust, reliable and fit for purpose. To this end, there have been several proposed methodologies for analyzing, designing, and building multi-agent systems (MAS). Gaia [17], MaSE [3] and Prometheus [9] can be mentioned as examples. These are the most

complete and end-to-end methodologies that introduce some interesting and useful ideas. One of them is the convenience of thinking an information system as an organization in which agents play *roles* and participate of *interactions* among roles. Following this idea we used a particular approach based on the Gaia methodology. Firstly, we present this approach. Secondly, we present the analysis and design, based on this approach, of the system architecture to support the Distributed Knowledge Management.

2.1. Applied Methodology

The methodology applied is based on the Gaia methodology [17]. We have selected the Gaia methodology because it proposes to the developers to think of building MASs as a process of organizational design. This idea suggests that is better design a MAS making a parallelism with the structure and behavior of the human organizations. This organizational approach followed by Gaia allows to capture particular features of the MASs, like the autonomous nature and the proactive behavior of the agents, that are more difficult of capturing with traditional techniques and methodologies (ie: object oriented and knowledge engineering models) adapted to the MASs development [18]. In addition, Gaia allows define the abstract levels of the MASs without taking into account the platform of development and the techniques and methodologies used to analyze, design and implement each agent. In this way, we have used the Gaia methodology to define the organization of the information systems, which define the architecture of the MAS. Then, we have designed the agents defining the particular software architecture of them.

The main concepts in Gaia are divided into two categories: abstract and concrete. The abstract concepts involve the *Roles* and the *Protocols*. A role is defined by four attributes: *Protocols*, *Permissions*, *Activities* and *Responsibilities*. Responsibilities are divided into two types: *Liveness Properties* and *Safety Properties*. The concrete concepts involve: *Agent types*, *Services* and *Acquaintances*. A role can be viewed as an abstract description of an entity's expected function. That is, a role is more or less identical to the notion of an office. A protocol defines the way that a role can interact with another role. The analysis phase implies defining the role model and the interaction model (protocols). The design phase implies defining the agent model, the service model and the acquaintance model. More details of Gaia methodology can be viewed in [17].

In some situations, the roles can be derived from the real-world organization that the MAS is designed to support. However, in other situations when the real-world organization does not exist or is not well defined, the identification of roles and protocols is not easy. Therefore, we have extended the Gaia methodology with the aim to help to the developers in the task of roles and protocols identification, due to Gaia does not provide a method to this task. In our approach, the idea is based on using other organizational concepts that can help in the analysis and design of MASs as an organization. One of these is the business process concept. Current management techniques point out that it is convenient to coordinate organization activities according to a set of business processes [5]. Once an organization goal is defined, it is possible to define one or more processes whose primary objectives will be to reach that goal. In this way, processes are useful to describe what have to be done to reach the goal of the organization. Then, by considering the process, its activities, inputs and outputs, and its clients, we can obtain the knowledge needed for the specification of the roles to be played in the organization and the protocols among those roles.

Following, we describe the stages of the applied methodology. The three first stages conform our approach that is based on the concept of business processes to identify roles and protocols. The last stages are part of the Gaia methodology. The stages are:

Analysis Phase:

Stage 1: starting from an organizational abstraction of the MAS, it is necessary to take into account that every organization has to define a goal and to direct all organizational agent efforts towards it. Then, the first step is to define the goal of the MAS.

Stage 2: define the processes to reach the organizational goal. Each process, is defined by:

- Inputs and outputs, and entities of the organization that provide inputs to process and receive its outputs.
- A set of activities and their respective precedence relationships. Each process activity is characterized by the following attributes: activity objective, inputs, outputs, task involved in the activity, constrains and resources (information) required by the activity. The achievement of successive objectives involved in each activity allows the organization (the system) to reach the goal.

Stage 3: identify roles and protocols in order to define the role model and the interaction model. Each activity of the processes should be performed by a role to be played in the organization that has to achieve the objective of that activity. Thus, a model of roles and protocols of the whole system in its abstraction as an organization is obtained. In this definition, some points should be taken into account:

- Process activities define the roles to be played to carry out these activities. Each activity is mapped to a role, but a role can perform one or more activities involved in the processes of the organization. If two or more activities share the same resource, these activities can be performed by a role in the organization.
- Performing an activity in the context of a process may imply different interactions with other process activities.

Such interactions are expressed by the activity inputs and outputs. These interactions define the protocols.

The result of this stage is a set of roles and a set of protocols for every process in the organization. These are the roles and the protocols that the MAS should perform to reach the goal for which it was designed.

Stage 4: define the schema of roles and protocol attributes in order to complete roles model and interactions model (Analysis stage in Gaia). Once the roles and protocols of the system have been identified, the following stage consists of making a detailed definition of such roles and protocols and of completing the role model with the role's schemas. Some elements defined in the processes can be used to help us to complete the role's schemas:

- An activity can be represented by a tasks sequence that has to be carried out as a transaction. The tasks involved in each activity of a process define the role activities, which represent in Gaia the internal activities that a role performs without interact with other role through a protocol.
- Process activity constraints generally define safety properties of the role that carry out those activities. Eventually, they can define new protocols.
- The resources define the resources in the role on which is the role's permissions defined.

With this stage we finished the analysis stage of the methodology with a complete definition of the abstract concepts used in Gaia.

Design Phase:

Stage 5: define the agent, the services and the acquaintance models, following the steps proposed by the design phase of Gaia.

2.2. The Analysis and Design of the MAS

In this section we describe the analysis and design of the system to support the distributed knowledge management, which have been done following the approach described above.

2.2.1. Defining the goal of the system

The idea of the system is to offer to the decision makers of each enterprise domain a query mechanism that allows them to find people who have the knowledge to generate the information required by them. Therefore, the goal of the system is "to process decision maker's queries in natural language and to find the possible information sources that allows to obtain answer to the queries requirements".

2.2.2. Defining the organizational processes.

Considering the system as an organization, we define three processes that the system has to carry out to achieve its goal. The inputs and outputs of these processes are showed in Figure 1.

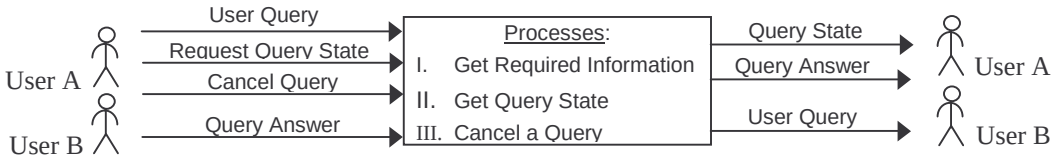

Figure 1. Inputs and outputs of the processes

The three processes defined are: (I) get the required information, (II) get the states of an issued query for information and (III) cancel a query. The first process, showed in Figure 2, is the main one. It consists of seven activities and its goal is to obtain an answer to a user's query. The first activity of this process has the purpose of obtaining the user's query. The second activity has the purpose of coordinating all users' query generated in the domains selecting the next query to be processed. The third activity has the purpose of determining the possible information sources (domains) for providing the required information to the user's query. The forth activity has the purpose of visiting the domains that the previous activity classified as possible providers of the required information. The fifth activity has the purpose of obtaining the query's answer. The sixth activity has the purpose of delivering the answer to the user that issued the query and the seventh activity has the object of learning from this search, thus improving the knowledge about domains as regards the information they can provide.

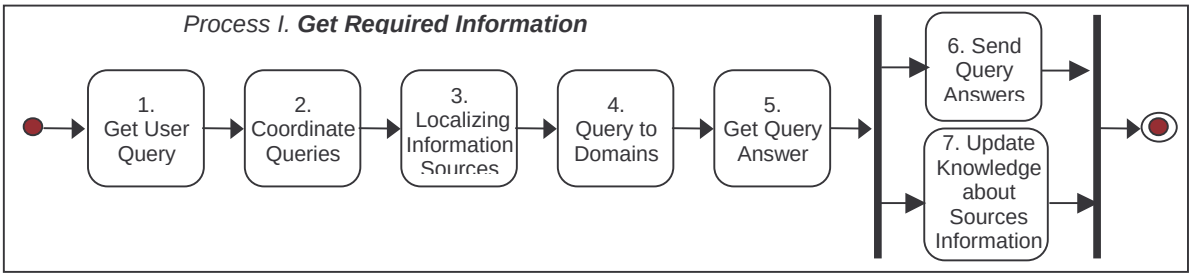

Figure 2. The main process to be carried out by the system viewed as an organization.

Also, in this stage, the activities are defined in detail. The attributes of each process activity are defined in the activity schema. Figure 3 presents the schema of the *Localizing Information Sources* activity.

Activity	*Localizing Information Sources*
Object	Determine the domains that are potentially capable of providing the required information.
Inputs	User query.
Outputs	List of domains to be queried.
Tasks	1. Receive a user's query for identify possible information sources. 2. Match a user's query in natural language with the knowledge about domains information. 3. Generate a list of domains that are potentially capable of answering the user query.
Constraints	Maximum number of queries to be received. Maximum number of queries to be simultaneously answered.
Resources	Knowledge Base about information managed by domains.

Figure 3. Attributes of Localizing Information Sources activity.

2.2.3 Identifying the organizational roles and protocols

Analyzing the processes with their activities we have identified the roles and protocols of the system. The system roles and its mean are described to next.

DomainUser. This role makes queries and query answers. Also, it can request the query state or can cancel a query. This role is not mapped to an agent because it represents the user and his interaction with the system. However, it is necessary to represent it due to this role sets protocols with other roles of the system.

DomainRepresentative. This role carries out tasks that involve the interaction with users. It makes actions that allow a domain to receive queries from the domain users and to receive answers sent by other domains. In addition, it allows a user to request for the state of a query and to cancel a query. The objective of this role is to perform *GetQueryUser*, *GetQueryState* and *CancelQuery* activities, which are part of the defined processes. There is a *DomainRepresentative* role in each domain.

QueryCoordinatorsAdministrator. The objective of this role is to perform the *CoordinateQueries* activity of the main process. Its main functions are to coordinate the user's query that arrived from different domains and to select the next query to be processed in the system.

InformationSourceLocator. The objective of this role is to perform the *LocalizingInformationSource* and *UpdateKnowledgeAboutInformationSources* activities of the main process. This role manages the knowledge about information managed by each domain and based on this information it generates a domain ranking list for a query. This list includes the possible domains that might answer the query.

QueryCoodinator. The objective of this role is to perform the *QueryToDomains* activity of the main process and other activities of other processes. Its main functions are to deliver the query to the possible domains, to obtain the query answer generated by some domain and to inform the query results.

ResponsesSupplier. The objective of this role is to perform the *SendQueryResponse* activity of the main process. There is a *ResponseSupplier* role in each domain that sends the query responses to the domain that originated it.

Once these roles have been identified, the protocols are defined following the activities and their relationships in the process. Analyzing the main process, the *DomainRepresentative* role interacts with the *DomainUser* role performing the *GetQuery* protocol to get the query from the user. Then, the *DomainRepresentative* role interacts with *QueryCoordinatorsAdministrator* role performing the *ReceiveQuery* protocol to allow the last role to coordinate the query. In this way, the *QueryCoordinatorsAdministrator* role performs its *SelectQuery* activity and assigns the query to a *QueryCoordinator* role performing the *AssignQuery* protocol. First, the *QueryCoordinator* role obtains the domains list to be visited performing the *ObtainDomainList* protocol with the *InformationSourceLocator* role. Using this protocol, the *QueryCoordinator* role gives the query to the *InformationSourceLocator* role performing the *RequestDomainsList* subprotocol. Then, this role performs the *GenerateDomainRanking* activity to generate the domain ranking list and finally it returns this list to the *QueryCoordinator* role performing the *GiveDomainList*

subprotocol. Following, the *QueryCoordinator* role visits each domain of the list and interacts with the *DomainRepresentative* role performing the *GiveQuery* protocol for delivering the query to the *DomainRepresentative* role in each domain. Then, the *DomainUser* role reads the query using the *GiveQuery* protocol. Once the *DomainUser* role generates an answer to the query, this role gives this query answer to the *ResponseSupplier* role performing the *GiveAnswer* protocol. Then, the *ResponseSupplier* role sends the query answer to the *DomainRepresentative* role, of the domain in which the query was originated, using the *GiveInformation* protocol. The *DomainRepresentative* role gets the query answer and then the *DomainUser* role recoveries the answers using the *InformResults* protocol. Figure 4 presents the roles and protocols derived from the main process. The process activities assigned to each role are shown by a tuple (Process_Number; Activity_Number). Protocols that settle interactions among roles are shown with arcs among roles, and the main internal activities that each role has to perform are indicated with a loop.

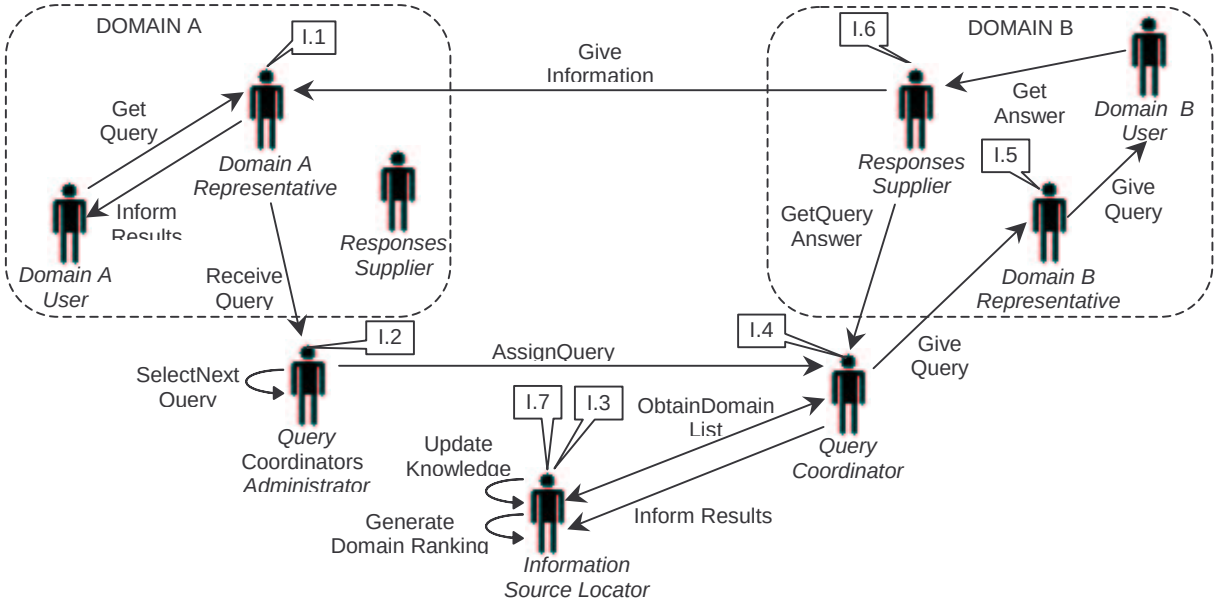

Figure 4. The Roles and Protocols from the main process *Get the Required Information*.

2.2.4 Defining the schema of the roles and the protocol attributes

Once the roles and the protocols are identified, we defined the schema of the roles and the protocol attributes to complete the role model and the interaction model. This is made following the Gaia methodology. As an example, Figure 5 shows a schema of the *InformationSourceLocator* role.

Role Schema: *InformationSourceLocator*
Protocols and Activities: ReceiveQuery, MatchQuery, GenerateDomainsList, GiveDomainsList, GetQueryAnswer, UpdateKnowledge
Permissions: reads *UserQueries* *QueryAnswer* changes *BaseKnowledge*
Responsabilities Liveness: InformationSourceLocator = ReceiveQuery.MatchQuery.GenerateDomainsList. GiveDomainsList | |GetQueryAnswer.UpdateKnowledge Safety: UserQueries < n // number of user queries simultaneously attended QueryAnswers < m // number of query answers simultaneously attended

Figure 5. Schema of the *InformationSourceLocator* role.

2.2.5 Defining the agent, service and acquaintance models.

Defining the agents model implies to define the mapping among roles and agents. In this way, the roles to be carried out by agents are settled. Figure 6 shows the four agents types that constitute the multi-agent system. These agents are the Domain Representative Agent (DRA), the Information Source Locator Agent (ISLA), the QueryCoordinatorsAdministratorAgent (QCAA) and the Query Coordinator Agent (QCA). In this model, it can see the roles that each agent has to perform to achieve the goal of the system. The number of DRA's instances depends on the number of domains to be supported by the system, due to there is a DRA for each domain. Once the agent model has been defined we have to define the services model. This model is out of the scope of this paper.

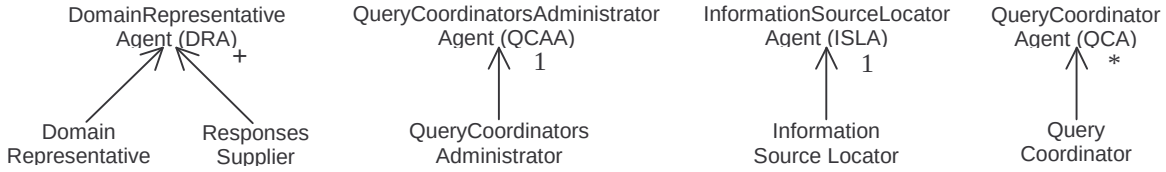

Figure 6. The agent model

Figure 8 shows the acquaintance model. This is the last model to be generated. It shows the communication links between agent types. The DRA has a loop that point out a communication link between agents of the same type.

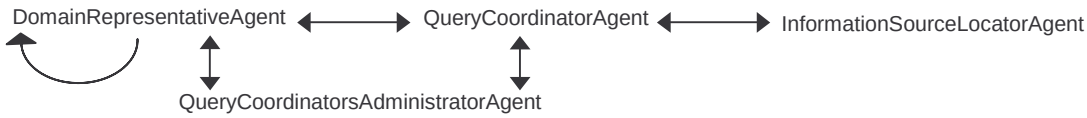

Figure 7. The Acquaintance Model

2.2.6 Defining Mobile Agents

Both mobile and static agents can be used to solve the same problems. In spite of the current diffusion of mobile agent technology [2], the proposed methodologies for developing MASs do not provide methods to determine when to use mobile agents. In our system we used an approach to identify mobile agents, which is not completely presented here due to it is out of the scope of this paper. The approach takes into account the software quality attributes that the MAS should possess due to mobile agents can contribute to achieve some of them. It proposes to identify agents that may be defined as mobile ones, identifying the agent's execution environment and looking at the communication pathways between agents of different execution environment. These communication pathways are obtained from the acquaintance model.

Following this approach, we assigned mobility property to the QCA. The mobility property in this agent can help to achieve the reliability and performance quality attributes that have been defined for our system. The execution environment types identified are: the environments in each domain where the DRA is executed, and the environment where the ISLA and the QCAA are executed. The QCA is executed in both environments due to its mobility property. The QCAs move to each domain of the domain list.

3 Architecture of the Information Source Locator Agent

In the architecture of the MAS defined above, the ISLA has been defined as the agent responsible of identifying the domains that offer greatest possibilities of answering the query formulated by the user. To this end, it should have a representation of the class of information that is managed in each domain and, in turn, it should transform the query formulated in natural language, to a format that allows to be compared with the representation of each domain.
The ISLA should update its knowledge about the domains for two main reasons: a) the dynamics of the organization implies a constant modification of the knowledge that each person possesses, and b) the initial domain representation is made in a subjective way and subject to errors.
According to the responsibilities described in the previous section, we propose a model for the ISLA with four components: *Communication Component, Knowledge Base (KB), Knowledge Retrieval Component* and *Learning Component* (see Fig. 8). A *Communication Component* is responsible for communicating the ISLA with the QCAs. This component will not be described in this paper and the other components are described above.

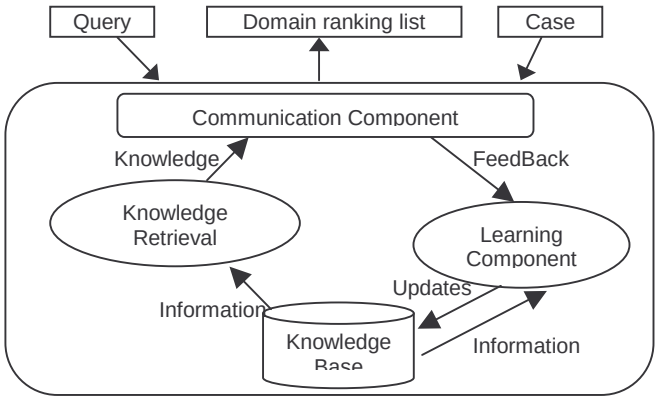

Figure 8. Information Source Locator Agent model

3.1 Knowledge Base

To facilitate all the processes carried out by the ISLA, its KB should store the representation of the information managed by each domain (Domain representation), the results of the searches carried out by the QCAs (Cases Base)

and tools necessary to interpret the natural language (morphological variants, n-grams, Stop-words and Separators). The class diagram of the KB is shown in Fig. 9.

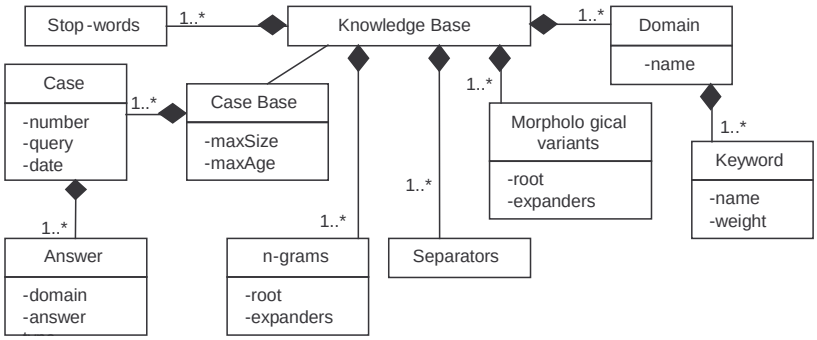

Figure 9. Class diagram of the Knowledge Base.

3.1.1 Domain representation

We have proposed a mechanism to identify the domains that should have the knowledge required by each query, adapting the existing methodologies for information retrieval from big document collections [14]. It is based on a vectorial representation of the available information in each domain. That is, for each domain d_j it should identify a set of keywords Kd_j representative of the information available in each domain. Each keyword can be either a word or a construction (sentence or expression) (For example: Data Base; due date; just in time). Kd_j is called the taxonomy of d_j. To each keyword t a weight w_t^j should be associated, which defines (in a 0-1 range) its importance with respect to the description of the domain context. Experts of each domain provide it using a strategy based on the AHP method [12], which we proposed in [14].

3.1.2 Morphological variants

The structural modifications that affect nouns, articles, adjectives and pronouns to express their gender and number, and verbs to denote their mode, tense, voice, person and number are called morphological variants. To diminish the size of domain representations, the keywords there included should have the same gender, number, and mode. We call this word as "root" and all morphological variants of this root are its "expanders". For example, if *sale* is the root, its expander could be *sales, salesman, salesmen, saleswoman, saleswomen, sell, sold, selling*, and so on. The ISLA KB should contain at least all morphological variants of the keywords included in the domains taxonomy. Keywords and all their morphological variants share the assigned weights.

3.1.3 N-grams

The keywords included in the domains representation could be n-grams. Then, it is necessary to include into the ISLA KB all the n-grams enclosed into the domains taxonomy (roots) with their respective morphological variants. For example, if data base is the root, its expander could be data bases.

3.1.4 Stop-words

They are all these words that do not help to discriminate between domains that can answer the queries or not. These stop-words are pronouns, prepositions, interjections, auxiliary verbs, articles.

3.1.5 Separators

They are all the symbols that can separate words as space, period, colon, semicolon, comma, brackets, bar and so on.

3.1.6 Cases base

The ISLA is responsible for keeping updated the representation of the information available in the system domains so as to enable a constant improvement in the system efficiency. For that purpose, it needs to analyze the query responses, i.e., which domain d_R provided the answer to query q_i, and in which position this domain was placed in the domain ranking list calculated by the ISLA. This information is provided by the QCAs once the required answer is found and will be called *case*. Each case *i* is stored in the cases base, including the following information:

Number of case: *i*
Query keywords: Kq_i
Domain that provided the answer: d_R

3.2 Knowledge Retrieval Component

Let D be the set of all domains being part of the system. Given a query q_i, the ISLA Knowledge Retrieval Component must define the subset of domains $D_P \subset D$, which have the potential for answering the information

requirement.

The user in natural language in his native tongue formulates the query. Then, to carry out this classification, the set of keywords Kq_i that represents the query q_i must be obtained. Each domain $d_j \in D$ must be compared with the query representation. To do that, the Retrieval Status Value $RSV(d_j,q_i)$ for each domain d_j must be computed every time a query arrives to the ISLA. This index is computed as follows [13]:

$$RSV(d_j, q_i) = \sum_{t \in Kqi \cap Kdj} w_t^j \quad (1)$$

where the w_t^j are obtained from the domain representation in the KB.

Domains with RSV greater than zero are assigned to the D_P set. Given a query q_i, the domains classified into D_p set are listed in decreased order of RSV. Starting from the RSV differences, the QCAs decide a search strategy.

To carry out these tasks we propose to model this component using the "pipes and filters" architectural style shown in Figure 10. Each filter processes the data and sends it to the next filter. The query, represented by an ordered set of characters, is the input data to the first filter, and the domain ranking list is the output data from the last.

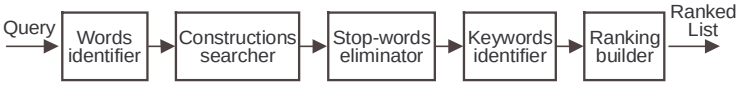

Figure 10. Knowledge retrieval component architecture

The *Words Identifier* filter identifies and eliminates the separators, using the separators set contained in the KB and provides an ordered list of words to the *Constructions Searcher* filter. This filter, accessing to the constructions set of the KB, identifies constructions and sends to the *Stop-words Eliminator* filter, an ordered list of words and/or constructions. Then, the connectors are identified using the stop-word set of the KB and they are eliminated. Starting of this list of words and constructions query representative, it is necessary to identify their roots with the end of determinate their weights from the domain representation in the KB. This task is made by the *Keyword Identifier* filter. The keywords list (Kq_i) is received by the *Ranking Builder* filter, which accessing to the KB obtains the name of all the domains of the organization and the keywords weight in each domain representation. Also, it knows the equation (1) to compute the RSV.

3.3 Learning Component

Since experts have defined domain representations, it is not strange that errors may affect the system efficiency. That is, domain d_R that can answer the query q_i does not have the first place in the domain ranking list and other domains d_j are better placed because $RSV(d_j,q_i) > RSV(d_R,q_i)$. These errors may be due to two main causes: the value of weights w_t^j is not the right one and/or not all keywords characterizing the domain are included.

It is necessary to highlight that the experts themselves in the taxonomy definition process can cause these errors, but they could also be originated by the evolution of domains that could take place as time goes by. Anyway, it will be necessary to update the system KB to avoid these errors. Analyzing the results of all those cases in which there were classification errors can carry out this updating. That is, domain d_R that answered to the respective query q_i did not have the greatest $RSV(d_R,q_i)$ and therefore it was not ranked first.

For the ISLA learning process we have defined an interpretative type Case Based Reasoning (CBR) strategy. In this CBR, cases are remembered to understand situations and to justify the election of an interpretation to begin reasoning with [7]. The ISLA should select some cases and interpret them, reasoning about what has happened in the cases and learning from their solutions. To carry out the learning, the interpretative CBR proposes the following steps [6]:

- *Retrieval*: Finding a set of cases with similar characteristics.
- *Interpreting and Proposing*: Analyzing the characteristics of the set of recovered cases to arrive to conclusions and propose changes.
- *Justifying and Criticizing*: Evaluating if the proposed changes will have a positive effect.
- *Evaluating*: Verifying that after performing the proposed changes, the expected result is obtained.

3.3.1 Retrieval

To carry out a convenient recovery of cases, they should be previously classified as *positive* or *negative* according to search results:

- *Positive cases* are those in which the domain that answered d_R has the first position in the ranking.
- *Negative cases* are those in which the domain d_R does not have the first position in the ranking.

Only when some negative cases are stored in the cases base, the learning process is required. Then, the information involved in these cases must be classified in three matrices, which establish work memory. The object of this

classification is to detect the need of modifying the weights of some keywords in the representation of a domain and/or incorporating new words into domain taxonomy. All matrices will have the same format, although they store different information (see Fig. 11).

Keywords	Domains			
	d_1	**d_2**	**...**	**d_j**
t_1t_2				
t_1				
t_2				
...				

Figure 11. Classification matrix format

The first field in all matrices, called keywords, stores all possible keywords combinations that belong to each Kq_i. For each domain d_j, the following fields store the name of the cases *i* that present different characteristics for each matrix.

Matrix 1 – (To find keywords with low weight): case *i* was answered by the corresponding domain d_j and the α keywords indicated in the first field belong to Kq_i and to Kd_j.

Matrix 2 – (To find keywords with high weight): case *i* was answered by domain d_R ($d_j \neq d_R$), $RSV(d_j,q_i) > RSV(d_R,q_i)$ and the α keywords indicated in the first field belong to Kq_i and to Kd_j.

Matrix 3 – (To find missing keywords in a taxonomy): case *i* was answered by the corresponding domain d_j and the α keywords indicated in the first field belong to Kq_i and not to Kd_j.

When the number of cases stored in a cell is higher than a predetermined value *N*, it is considered that there is enough information to propose some change in the KB. These cases are recovered to go on with the analysis.

3.3.2 Interpreting and Proposing

Once a set of cases sharing the same characteristics is recovered, they must be analyzed to propose changes in the KB so that the answering domains are better ranked. This process is carried out using four rules:

Rule 1: IF β cases ($\beta > N$) occur, which contain the same α keywords in the query representation and
- a) The same domain d_R answers, but it is not the first one in the ranking (d_{1i}),
- b) α keywords belong to Kd_R

(all this information gathered from Matrix 1),

THEN these keywords weights in d_R representation should increase, maintaining the original weights relation and the condition $0 \leq w_t^d \leq 1$.

$$\left(w_{tk}^{dR}\right)_{new} = min\left\{1;\left(w_{tk}^{dR} + \frac{1}{\beta}\left(\sum_{i=1}^{\beta}\left(RSV\left(d_{1i},q_i\right) - RSV\left(d_R,q_i\right)\right)\right)\frac{w_{tk}^{d_R}}{\sum_{t_i \in T} w_{ti}^{d_R}}\right)\right\} \qquad \forall t_k \in T \tag{2}$$

where T is be the keywords set shared in β cases (α=#T) (first field of the matrix) and d_{1i} is the first domain ranked for case *i*

Rule 2: IF β cases ($\beta > N$) occur, which contain the same α keywords in the query representation and
- a) the same domain d_J has a greater RSV than that of the answering domain (d_{Ri}),
- b) α keywords belong to Kd_J

(information gathered from Matrix 2),

THEN those keywords weights in the d_J representation should diminish.

$$\left(w_{tk}^{dJ}\right)_{new} = max\left\{0;\left(w_{tk}^{dJ} - \frac{1}{\beta}\left(\sum_{i=1}^{\beta}\left(RSV\left(d_J,q_i\right) - RSV\left(d_{Ri},q_i\right)\right)\right)\frac{w_{tk}^{dJ}}{\sum_{t_i \in T} w_{ti}^{dJ}}\right)\right\} \qquad \forall t_k \in T \tag{3}$$

where d_{Ri} is the domain that answers the query q_i:

Rule 3: IF β cases ($\beta > N$) occur, which contain the same α keywords in the query representation and
- a) The same domain d_R answers, but it is not the first one in the ranking (d_{1i}),
- b) α keywords do not belong to Kd_R

(information gathered from Matrix 3),

THEN these keywords should be incorporated into the d_R representation with a weight $(w_{tk}^{dR})_{new}$ calculated as follows:

$$\left(w_{tk}^{dR}\right)_{new} = min\left\{1;\ \frac{1}{\alpha\,\beta}\sum_{i=1}^{\beta}\left(RSV\left(d_{1i},q_i\right)-RSV\left(d_R,q_i\right)\right)\right\} \qquad \forall\, t_k \in T \tag{4}$$

Rule 4: IF β cases (β > *N*) occur, which contain the same α keywords in the query representation and

a) The same domain d_R answers, but it is not the first one in the ranking (d_{1i}),
b) the same domain d_J has a greater RSV than that of the answering domain (d_R),
c) α keywords belong to Kd_R and Kd_J

(all this information gathered from Matrices 1 and 2),

THEN these keywords weights in d_R representation should increase and diminish in d_J representation.

As this rule imposes the execution of stronger restrictions, we suggest that the weights variation assures that the β cases become positive. Then,

$$\left(w_{tk}^{dR}\right)_{new} = min\left\{1;\left(w_{tk}^{dR}+1/2\,\frac{w_{tk}^{d_R}}{\sum_{t_i\in T} w_{ti}^{d_R}}\max_i\left\{RSV\left(d_J,q_i\right)-RSV\left(d_R,q_i\right)\right\}\right)\right\} \qquad \forall\, t_k \in T \tag{5}$$

$$\left(w_{tk}^{dJ}\right)_{new} = max\left\{0;\left(w_{tk}^{dJ}-1/2\,\frac{w_{tk}^{d_J}}{\sum_{t_i\in T} w_{ti}^{d_J}}\max_i\left\{RSV\left(d_J,q_i\right)-RSV\left(d_R,q_i\right)\right\}\right)\right\} \qquad \forall\, t_k \in T \tag{6}$$

3.3.3 Justifying and Criticizing

As a result of the application of any of the aforementioned rules, a *hypothesis* is obtained. A hypothesis represents a possible state of the KB.

For each hypothesis we must identify which keywords will have their weights changed, the domains they belong to and the value of the new weights. After doing that, it is necessary to evaluate the effect this modification will cause on the ISLA efficiency. Although the proposed changes guarantee improvements in some negative cases, they could also damage others that belong to the positive cases set.

In order to evaluate which hypothesis is convenient to be applied, a *performing measure* (η) is defined, whose value increases as cases are closer to become positive.

$$\eta = \frac{1}{n}\sum_{i=1}^{n}\frac{RSV\left(d_{Ri},q_i\right)}{RSV\left(d_{Ri},q_i\right)+\sum_{p\in Db}\left(RSV\left(d_p,q_i\right)-RSV\left(d_{Ri},q_i\right)\right)} \tag{7}$$

where *n* is the number of cases in the Case Base, and $Db=\{d_j / RSV(d_j,q_i) > RSV(d_{Ri},q_i)\}$ is the set of domains that have a better position than d_R in each case.

If the η value for the KB is increased when applying a hypothesis, the latter can be applied. When several hypotheses are in conflict, the one that produces the highest η value should be applied.

3.3.4 Evaluating

Although the changes carried out in the KB always improve the previous state, the historical sequence of modifications could harm the system efficiency. To avoid this, the ISLA records the generated hypotheses in memory, so that every time a new case is received it can analyze which the result has been in the previous states. In this way, it may have the possibility of undoing or re-doing changes.

4 Conclusions and Future Work

A knowledge management system has to support the explicit knowledge and the tacit knowledge. However, considering the specific characteristics of the tacit knowledge, the management of it is complex and different at the management of the explicit knowledge. Therefore, in this work we proposed a multi-agent system (MAS) to support the management of the tacit knowledge distributed in the domains of an organization. The MAS, through a process that consists in a query mechanism, allows to deduce in what domain of the organization the required tacit knowledge can be located and to support the organization of the knowledge about the possible domains in where the tacit knowledge is located.

The designed multi-agent system has some characteristics to be highlighted. It is easy to use by the decision makers because he/she has only to formulate a query in natural language. It is adaptable due to it has not been designed for a particular organization. It is dynamic due to it has a learning mechanism to update the knowledge about the information that can be generated by the tacit knowledge in the domains. Furthermore, the mobility of the QCAs helps to achieve the reliability and performance quality attributes that we consider important for this system.

In the development of the MAS, we have used an approach extending the Gaia methodology following the idea of designing the MAS as an organization. This organizational view allows to capture the main properties that identify a MAS, like the autonomous nature and the proactive behavior of the agents. In this way, we have used the business process concept to identify roles and protocols. Starting the analysis of the MAS defining the process to be carried out in the organization of the system allows afterwards to identify and define the roles collection more appropriated, which define the organizational structure of the system. The applied approach based on the Gaia methodology made possible to define the architecture of the MAS without to define the implementation details.

Once defined the architecture of the MAS, the agents that make up the MAS were designed defining the architecture of them. In this work, we have described the architecture of the agent ISLA. This agent has an important role in the tacit knowledge management. To perform this role it has a knowledge retrieval component that allow to determine where the required knowledge can be found. Furthermore, a learning component have been designed to allow the agent to adapt the organizational changes, keeping the knowledge about the organization's knowledge up to date.

On the one hand, a prototype of the MAS is being implemented at this moment, using the Aglets Software Development Kit. On the other hand, a test prototype of the agent ISLA has been implemented in order to test its abilities using our faculty as organization for the analysis. The first test performed has shown that the principal difficulty is presented during the domain representation definition. Another problem is the poor performance due to the big amount of information that it has to keep updated. The learning process has shown a good yield, but even more time of operation is required to evaluate it correctly.

In a future work we will extend this system adding a support for the explicit KM. To do that, we will incorporate a distributed organizational memory. The RA of each domain will be the responsible to management the organizational memory of the domain represented by it.

References

[1] Brézillon P & Pomerol J-Ch Contextual knowledge sharing and cooperation in intelligent assistant systems. Le Travail Humain, Vol. 62, No. 3, (1999), pp. 223-246.

[2] Chess, D., Harrison, C.G., Kershenbaum, A.: Mobile agents: Are they a good idea?. *In Mobile Object Systems: Towards the Programmable Internetwork*, ed. by J. Vitek and C. Tschudin. Berlin, Germany: Lecture Notes in Computer Science (1997) pp. 25-47.

[3] DeLoach, S., Wood, M., Sparkman, C.: Multiagent Systems Engineering. *International Journal of Software Engineering and Knowledge Engineering*. Vol. 11, no.3 (2001) pp. 231-258.

[4] Dieng, R. Knowledge Management and the Internet. *IEEE Intelligent System.* (May/June, 2000), pp. 14-17

[5] Klein, M.: Reengineering Methodologies and Tools. *Information System Management*. (1994) pp. 30-35.

[6] Kolodner, J.,*"Case Based Reasoning'*, Morgan Kaufman Publishers, USA, 1993.

[7] Lopez de Mantaras, R. And Plaza, E. Case-Based Reasoning: An overview. .*AI Communications Journal*, Vol. 10, No. 1, (1997), pp. 21-29.

[8] Nonaka, I. A Dynamic Theory of Organisational Knowledge Creation. Organisational Science. Vol. 5, No. 1, (1994), pp. 14-37.

[9] Padgham, L. and Winikoff, M. Prometheus: A Methodology for Developing Intelligent Agents, in the Proc. of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems, 2002.

[10] Pomerol, J. and Brézillon, P. About some relationship between Knowledge and Context. Submitted to the 3rd International Conference on Modeling and Using Context(CONTEXT-01).Series Lectures in Computer Science, Springer Verlag, 2001.

[11] Rabarijaona, A; Dieng, R.; Corby, O.; Ouaddari, R. Building and searching an XML-based corporate memory. *IEEE Intelligent System.* (May/June, 2000), pp. 56-63.

[12] Saaty, T.L., *"The Analytical Hierarchy Process"*. New York: McGraw-Hill, 1980.

[13] Schwartz, D. and Dov Te'eni, D. G. Tying Knowledge to action with kMail. *IEEE Intelligent System.* (May/June, 2000), pp. 33-39

[14] Stegmayer,G., Taverna,M.L., Chiotti, O. and Galli, M.R., The Agent Routering Process of Dynamic Distributed Decision Support System, *Journal of Computer Science & Technology*, Vol. 5, No. 2, (2001), pp. 30-43

[15] Van Heijst, G.; Van der Spek and Kruzinga, E. Organizing corporate memories. Proc. 10th Banff Workshop on Knowledge Acquicition for Knowledge-Based Systems 1996.

[16] Wallis, C. Consciousness, context and know-how. http://www.arts.uwaterloo.ca , 2000.

[17] Woooldridge, M., Jennings, N., Kinny, D.: The Gaia Methodology for Agent-Oriented Analysis and Design. *Journal of Autonomous Agents and Multi-Agent Systems*. Vol. 3, No. 3, (2000), pp 1-26.

[18] Zambonelli, F., Jennings, N., Omicini, A, Wooldridge, M. *Agent-Oriented Software Engineering for Internet Applications*. In Coordination of Internet Agents (eds. A. Omicini, F. Zambonelli, M. Klusch and R. Tolksdorf) Springer Verlag, 326-346, 2001.