Agent for Information Source Location in a Dynamic DSS

C. Mozzati[1], M.L. Taverna[1], M.L. Caliusco[1], O. Chiotti[1,2] and M.R. Galli[1,2]
[1]GIDSATD - UTN – FRSF, Lavaisse 610 - 3000 SANTA FE – Argentina
[2]INGAR – CONICET, Avellaneda 3657 - 3000 SANTA FE – Argentina
mrgalli@ceride.gov.ar, chiotti@ceride.gov.ar

Abstract

Inside an organization there is information that can only be generated by people, then, when a decision maker in any time and anywhere of an organization requires this information, he/she has to solicit it to who has the ability of generate it. To fulfill this information requirement, we are developing a multiagent system that we call dynamic Decision Support System (DSS). This system is composed by *Domain Representative Agents* (DRAs); an *Information Source Locator Agent* (ISLA) and mobile agents called *Query Coordinator Agents* (QCAs). The ISLA is the agent responsible for guiding information requirements from users to distributed DSS domains that offer greater possibilities of answering them. It has the ability to interpret queries formulated in natural language, to identify the relationships among the characteristics of queries and domains, and to learn from errors that it make during its operation to diminish the number of consulted domains in each information requirement.
The purpose of this work is to describe the ISLA architecture and its main components; using design patterns and knowledge engineering methodologies.

Keywords: Software Agents, Knowledge Management, Decision Support System

1. Introduction

Every section composing an organization (*domains*) is constantly carrying out decision-making processes. Many of these processes can be part of its habitual activities and thus their "structure" is perfectly defined. In other words, gathering the needed information can be completely automated, since the sources that provide the data and the required and available structure of those data are known. Several enterprise integration systems have been developed to provide the automatic transfer of information among the different domains in an organization [16]. But as a result of the current dynamics of business processes, the number of non-habitual decision-making processes is becoming greater and greater. These processes imply an information and knowledge requirement that is sometimes beyond the reach of the decision maker since he does not usually handles it.

Knowledge acquisition deals with the issues involved in knowledge extraction in its various forms. That is, from the organization's knowledge bases, data bases, 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 [15].

Two kinds of knowledge can be distinguished: explicit and tacit knowledge [13]. 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 [2].

An approach to support the knowledge management is based on developing a corporate or organizational memory (OM). Several works are focused in this direction ([15], [14]). An organizational memory is defined 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 [21].

An organizational 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 OM. That is, in an organization there is information that can only be generated by people, then, when a decision maker requires this information, he/she has to solicit it to whom has the ability to generate it. In this way, another type of support to facilitate the access to this knowledge is required.

In the GIDSATD (Research Group of Decision Support Systems) we are developing a multiagent system that we call dynamic Decision Support System (DSS) [4]. A dynamic DSS allows establishing contact among domains for the acquisition of necessary information for decision-making. This implies that both the information to be transferred and the domains to be communicated are not specified on the system at design time. The DSS itself should interpret the information requirement and infer which domain can answer it at the run time.

A dynamic DSS can act as a complement to traditional DSSs when the information they require are not disposable by the users. Many DSSs have been developed. In Turban and Aronson (1998) several examples are described, but they all assume that the data are well known by the decision maker [19].

A dynamic DSS operates in the following way: When a user of the system needs some information, he/she makes a query in natural language and the dynamic DSS transfers that information requirement to a domain that could satisfy it. For that purpose, the system determines the sites that offer a greater possibility of providing the required information and those are firstly targeted. Then, when it gets the answer from a domain, it sends this information to the domain that asked for it.

This system could be viewed as a multi-agent system (Fig. 1), where each activity required to operate it is under the responsibility of different software agents.

The main activities this system must perform are:

- Transferring queries and answers between the user and the system.
- Identifying the domains that are able to provide an answer.
- Searching the domain that effectively has the required information.

The first activity is assigned to a static agent in each domain, the *Domain Representative Agents* (DRAs); the second one to a static agent called the *Information Source Locator Agent* (ISLA) and the third one is assigned to mobile agents called *Query Coordinator Agents* (QCAs) [22].

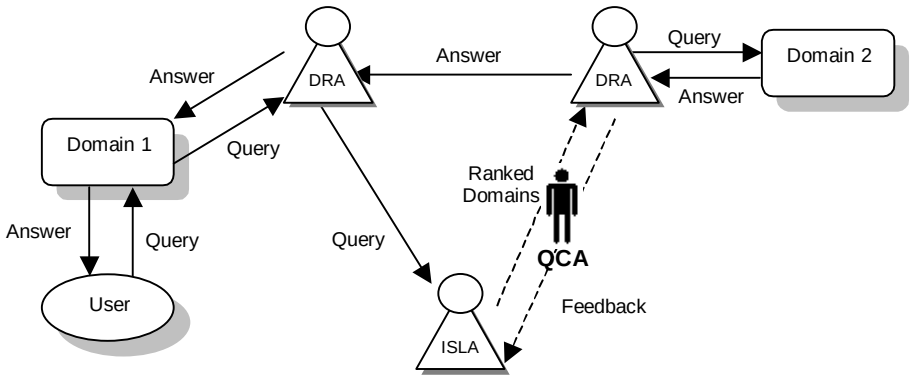

Fig. 1. Multiagent architecture for the dynamic DSS

Whenever an information requirement is formulated from a domain, its DRA sends the query to the ISLA. It reads and analyzes the query and determines a ranked list of domains that would provide the solicited information. Then the ISLA gives the ranked list to a QCA, which visits, in order, these domains with the query until it find an answer. Once it has visited the domains of the ranked list, if it finds a domain with the required information, it goes back to the ISLA again and informs the obtained results, allowing it to obtain new information to update its KB for future queries. Simultaneously, the DRA of the answering domain sends the information to the original DRA.

In this system, the ISLA is responsible for the management of the knowledge about the information handled by each domain and it guides the mobile agents journey. The design of its search strategy requires a special attention. Stegmayer et al. ([17][18]) established the principles for the selection of domains able to answer the queries and the need of implementing a learning mechanism to update its knowledge about the domains.

The ISLA is responsible for the system efficiency. That is, to gather the required information asking a small number of domains. The detailed study of this agent is the purpose of this work. Immediately we analyze the ISLA responsibilities. Then, the ISLA architecture is proposed, whose principal components (the Knowledge Base, the Knowledge Retrieval component and the Learning component) are described in detail.

2. The Role of the Information Source Locator Agent

The ISLA plays the role of guiding the QCAs journey. Its main responsibility is to identify the domains that offer greater possibilities of answering the query formulated by the user. The purpose is to prevent each query from being systematically directed to all domains, increasing the traffic in the net and interfering with the normal operation of domains. For that purpose, the ISLA must have the ability to analyze an information requirement, and determining to which domains it will be

directed. That is, it must weight the system domains able to answering the information requirement. To do this, the ISLA must know the knowledge about the information handled by each domain and must have the capacity for interpreting the queries formulated in natural language.

Another responsibility of the ISLA is to update its knowledge about each domain to improve the system effectively. Analyzing the results of all those cases in which there were weighting errors, this updating can be carried out. This process can be defined as a learning process that must be structured so that it can be automatically performed by the system [18].

3. Architecture of Information Source Locator Agent

Software agents is an emerging technology and many diverse Agent Oriented Software Engineering (AOSE) approaches and methodologies have been proposed [8]. Nevertheless, many existing MAS methodologies intentionally do not support intelligent agents; rather, they aim for generality and treat agents as black boxes. Kendall et al. [9] introduce the use of design patterns [6] for intelligent agent design. Selecting patterns as a methodology for agent development is being justified by referring to the previous successes of applying patterns in traditional software technology [20].

Particularly, the layered pattern provides an overall structure to a complicated system with many modules [10]. Each layer provides services to the layer above it and serving as a client to the layer below. Agents should be decomposed into layers because i) higher level or more sophisticated behavior depends on lower level capabilities, ii) layers only depend on their neighbors, and iii) there is a two-way information flow between neighboring layers [3].

According to the responsibilities described in the previous section and the layered agent architectural pattern we propose the following multi-layered architecture for the ISLA (see Fig. 2):

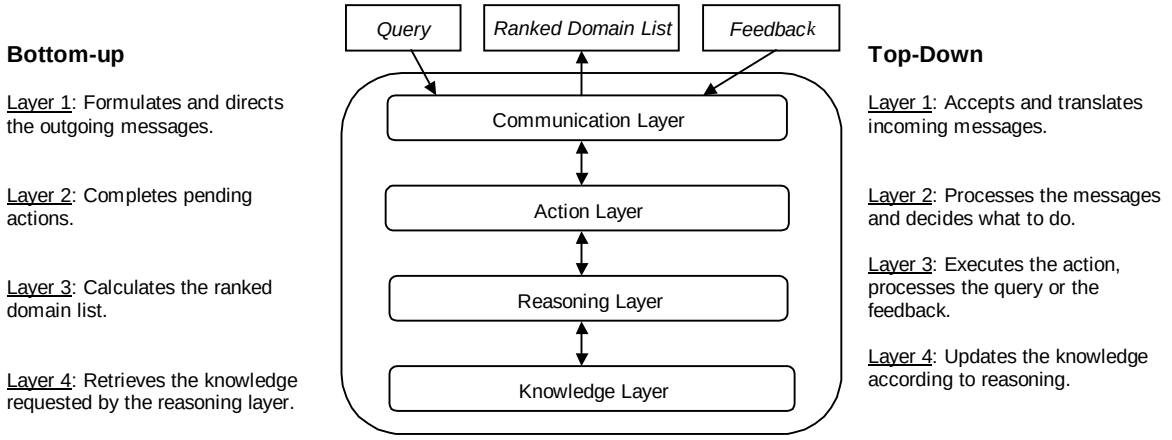

Fig. 2. Information Source Locator Agent model

- *Communication Layer:* it is responsible for communicating the ISLA with QCAs. This layer formulates messages for QCAs and translates the incoming messages into ISLA semantic. That is, it receives the queries or the information obtained from the domains that have been visited and translates them to the ISLA semantic. Furthermore, it receives the ranked domain list from the action layer, formulates the message and directs it to the QCA.
- *Action Layer:* in this layer is determined what to do next according to the messages that have been received and when the reasoning layer generates the ranked domain list the action layer informs it to the communication layer.

- *Reasoning Layer:* this layer is composed by two components.
    - ♦ *Knowledge Retrieval Component:* it interprets the queries formulated in natural language, makes the classification process and generates the ranked list of domains to visit.
    - ♦ *Learning Component:* it is responsible for keeping the KB updated.
- *Knowledge Layer:* this layer manages ISLA's knowledge about the information managed by each domain, the results of the searches carried out by the QCAs (cases) and the tools necessary to interpret the natural language.

The purpose of this paper is to describe the analysis and design of the two lowest layers: Knowledge and Reasoning.

3.1. Knowledge Layer

The ISLA Knowledge Layer is organized in such a way that facilitates the analysis of a query formulated in natural language and the comparison of the query with the knowledge that manages each domain. Therefore it is necessary to store a representation of this knowledge in a Knowledge Base (KB). It should also facilitate the learning process, for which it will store the perceptions the ISLA receives about how the domains evolve. To achieve this, we propose that the Knowledge Layer have to be formed by a KB with the following elements (see Fig. 3):

- Domains representations.
- Morphological variants.
- N-grams.
- Stop-words.
- Separators.
- A cases base.

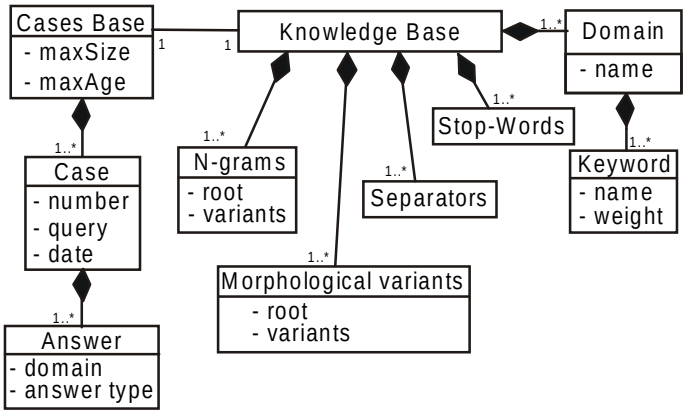

Fig. 3. Classes' diagram of the Knowledge Layer.

Domain representation. Stegmayer et al. [17] have proposed a knowledge retrieval mechanism for a dynamic DSS, following the ideas of the existing methodologies for information retrieval from big document collections. It is based on a vectorial representation of the knowledge managed in each domain. That is, for each domain d_j it should identify a set of keywords Kd_j representative of the knowledge available in each domain. Each keyword can be either a word or a n-grams (sentence or expression) (For example: Data Base; due date; just in time; etc.). 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 knowledge.

This representation can be provided by experts from each domain, using an appropriate methodology as the one proposed by Stegmayer et. al [17]; or it can be obtained through the analysis of documents that are available in the domain and selected by the expert to represent the available knowledge. For this last case, it is necessary to use document indexation techniques ([5], [1]). The first alternative, although requiring much effort from the experts, allows representing part of the tacit knowledge. The assistant system that supports the capture of this initial representation of each domain will not be described in this work.

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 the domain representations, the keywords included in it should have the same gender, number, mode, etc. We call this word as root. For example, considering *sale* as a root then its variants could be *sales, salesman, salesmen, saleswoman, saleswomen, sell, selling,* etc. 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.

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 *database* is the root, its variant could be *databases.*

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, propositions, interjections, auxiliary verbs, articles, etc.

Separators. They are all the symbols that can separate words as space, point, colon, point and colon, double point, cross, bar, etc.

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 search results, i.e., which domain d_R provided the answer to query q_i, and in which position this domain was placed in the ranked 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. Reasoning Layer

The Reasoning layer offers two services to the Actions layer: making ranking and processing feedback. These services are carried out by the Knowledge Retrieval Component and for the Learning Component respectively. To simplify the access to this layer in such a way that the layer above it should not know the internal components of the Reasoning layer, the facade pattern is used.

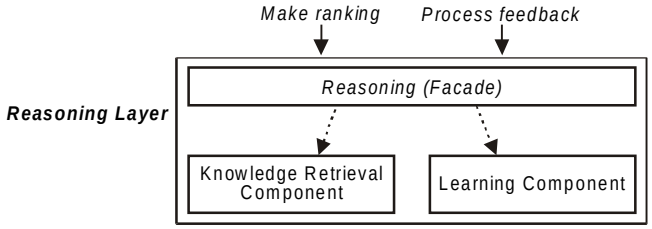

Fig. 4. Reasoning layer architecture.

3.2.1. 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 formulates the query in natural language. 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 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 [17]:

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

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, and the other domains are assigned to the D_N set (Non potential domains).

Given a query q_i, the RSV value of each domain classified into the D_P set can be used to rank these domains. That is, the domains $d_j \in D_P$ are arranged according to the descending values RSV forming a ranking for a query. Starting from the RSV differences, the QCAs decide the search strategy to be used.

To carry out these tasks we propose to model this component using the "pipes and filters" architectural style [7]. 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 ranked list of domains is the output data from the last. In this case, there are five filters showed in Fig. 5.

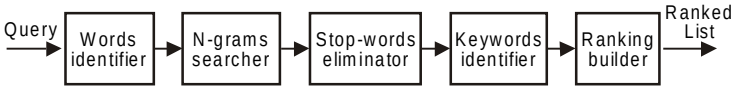

Fig. 5. Knowledge retrieval component architecture

Words Identifier. This filter receives a string as input and provides a list whose elements are the words present in the input string. To do this, it uses the separators set contained in the KB.

N-grams Searcher. This filter receives a list of words as input, it identifies n-grams and provides a list of words and/or n-grams. This filter has access to the n-grams set of the KB.

Stop-words Eliminator. This filter receives a list of words and/or n-grams. It eliminates the connectors from the list. To carry out this filtering, it has access to the stop-words set of the KB.

Keywords Identifier. This filter receives as input a list of words and n-grams and provides a list of keywords (root) corresponding to each word or n-grams. If the keyword is not found, that word or

n-grams is incorporated in the output list just as it entered. To carry out this filtering, it has access to the morphological variants and n-grams sets contained in the KB.

Ranking Builder. It receives a keyword list and provides a list formed by tuples, domain name and RSV of the same one, for all domains with RSV>0, ordered in decreasing way for RSV. To do that, this filter has access to the KB to obtain the name of all the organization's domains and the keywords weight in each domain representation. Also, it knows the equation (1) to compute the RSV. This filter has an internal architecture in layers that we do not explain in this paper for space reasons.

3.2.2 Learning Component

The initial representation of each domain is obtained from the explicit knowledge that is either captured from available work documents in the domain or defined by the domain experts. In both cases, 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 ranked list and others 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.
- Not all keywords characterizing the domain are included.

It is necessary to highlight that these errors can arise during the capture process of the initial representation of each domain, 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 Cases 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 [12]. 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 [11]:

- *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.

Retrieval. To carry out a convenient recovery of cases, they should be previously classified.

Firstly, cases must be classified as *positive* or *negative* according to the search results:

Positive cases are those where the domain that answered d_R has the first position in the ranking.

Negative cases are those where 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 is classified in three matrices, which establish work memory. The objects of this classification are to detect the need of modifying the weights of some keywords in the representation of a domain and/or incorporate new words into domain taxonomy. All matrices have the same format, although they store different information (Fig. 6).

The first field in all matrices, called keywords, stores all possible keywords combinations that belong to each Kqi. For each domain dj, the following fields store the name of the cases i that present the characteristics described next.

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

Fig. 6. Classification matrix format

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 keywords lacking in 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.

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.

Let $T=\{t_k\}$ be the keywords set shared in β cases (α=#T) (first field of the matrix);

Let d_{1i} the first domain ranked for case *i*:

$$\Delta w = \frac{1}{\beta}\sum_{i=1}^{\beta}\left(RSV(d_{1i},q_i) - RSV(d_R,q_i)\right) \tag{2}$$

To maintain the original weights relation, the increase in the weight of the keyword t_k in the d_R representation, δ_{tk}, should be:

$$\delta_{tk} = \Delta w \frac{w_{tk}^{d_R}}{\sum_{t_i \in T} w_{ti}^{d_R}} \qquad \forall\ t_k \in T \tag{3}$$

Then, the new weight of t_k in the d_R representation should be:

$$\left(w_{tk}^{dR}\right)_{new} = min\left\{1; \left(w_{tk}^{dR} + \delta_{tk}\right)\right\} \qquad \forall t_k \in T \tag{4}$$

maintaining the condition $0 \leq w_t^d \leq 1$.

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 great RSV than 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.

Let d_{Ri} be the domain that answering the query q_i:

$$\Delta w = \frac{1}{\beta}\sum_{i=1}^{\beta}\left(RSV(d_J, q_i) - RSV(d_{Ri}, q_i)\right) \tag{5}$$

the diminution in the weight of the keyword t_k in the d_J representation, δ_{tk}, should be:

$$\delta_{tk} = \Delta w \frac{w_{tk}^{dJ}}{\sum_{t_i \in T} w_{ti}^{dJ}} \qquad \forall\ t_k \in T \tag{6}$$

Then, the new weight of t_k in the d_J representation should be:

$$\left(w_{tk}^{dJ}\right)_{new} = max\left\{0; \left(w_{tk}^{dJ} - \delta_{tk}\right)\right\} \qquad \forall t_k \in T \tag{7}$$

Rule 3: 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) α keywords do not belong to Kd_R

(information gathered from Matrix 3),

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

$$\Delta w = \frac{1}{\beta}\sum_{i=1}^{\beta}\left(RSV(d_{1i}, q_i) - RSV(d_R, q_i)\right) \tag{8}$$

$$\delta_{tk} = \frac{\Delta w}{\alpha} \qquad \forall\ t_k \in T \tag{9}$$

$$\left(w_{tk}^{dR}\right)_{new} = min\{1; \delta_{tk}\} \qquad \forall t_k \in T \tag{10}$$

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 great RSV than 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 it imposes the execution of stronger restrictions, we suggest that the weights variation assures that the β cases become positive. Then,

$$\Delta w = \max_i\left\{RSV(d_J, q_i) - RSV(d_R, q_i)\right\} \tag{11}$$

$$\delta_{t_k}^{d_R} = 1/2\,\Delta w \frac{w_{tk}^{d_R}}{\sum_{t_i \in T} w_{ti}^{d_R}} \qquad \forall\ t_k \in T \tag{12}$$

$$\delta_{t_k}^{d_J} = 1/2\,\Delta w \frac{w_{tk}^{d_J}}{\sum_{t_i \in T} w_{ti}^{d_J}} \qquad \forall\ t_k \in T \tag{13}$$

$$\left(w_{tk}^{dR}\right)_{new} = \min\left\{1; \left(w_{tk}^{dR} + \delta_{t_k}^{d_R}\right)\right\} \qquad \forall t_k \in T \tag{14}$$

$$\left(w_{tk}^{dJ}\right)_{new} = \max\left\{0; \left(w_{tk}^{dJ} - \delta_{t_k}^{d_J}\right)\right\} \qquad \forall t_k \in T \tag{15}$$

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.

Once we have identified for each hypothesis which keywords will have their weights changed, which domain they belong to and which the value of the new weights are, 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.

$$\eta = \frac{1}{n}\sum_{i=1}^{n} \frac{RSV(d_{Ri}, q_i)}{RSV(d_{Ri}, q_i) + \sum_{p \in Db}\left(RSV(d_p, q_i) - RSV(d_{Ri}, q_i)\right)} \tag{16}$$

where n is the number of cases in the Cases 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.

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 registers in memory the generated hypotheses, so that every time that a new case is received it can analyze which had been the result in the previous states. In this way, it can have the possibility to undo or to re-do changes.

4. Conclusion

The non-habitual decision process requires gathering explicit knowledge stored in some domain of the organization and/or tacit knowledge that bellows to people of the organization. In this work we present a multi-agent system that allows establishing contact between domains that require information or knowledge and domains that should generate it. To this end, we propose to acquire and to store the knowledge about what the tacit knowledge of the organization is and where it is. This knowledge is managed by the ISLA, and it is used to help the decision makers to find the required information. The system 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 developed for a particular organization and it is dynamic due to it has a learning mechanism to update the knowledge about the information that can be generated in each domain.

In this work we have shown the preliminary analysis and design of ISLA, the intelligent agent responsible for the efficiency of system. A test prototype of this agent is being implemented at this moment, taking our faculty as organization for the analysis. The different departments (students, library, etc.) are considered as system domains. The learning process has shown a good yield, but even more time of operation is required to evaluate it correctly. Starting from the performance analysis we are beginning a new cycle in the system analysis and design process.

In a future work we will extend the learning mechanism to consider partial answers and answers from multiples domains.

References

[1] Baeza-Yates, R. And B. Ribiero-Neto. Modern Information Retrieval. Addison-Wesley, Wokingham, UK, (1999).

[2] Brézillon P. & Pomerol J-Ch. "Contextual knowledge sharing and cooperation in intelligent assistant systems". *Le Travail Humain*, **62**, 3, 223-246 (1999).

[3] Buschmann, F., R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-Oriented Software Architecture: A System of Patterns*. Wiley and Sons. (1996).

[4] Cabral, L., Caliusco, M., Nissio, H., Villarreal, P., Galli, M.R., Taverna, M. and Chiotti, O.,"Use of Agents Technology to Support Distributed Decision Processes", *First World Conference on Production and Operations Management*, POM. Sevilla, Spain. August, (2000).

[5] Crestani, F. Exploiting the Similarity of Non-Matching Terms at Retrieval Time. *Information Retrieval*, **2**, 25-45, (2000).

[6] Gamma, E., R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison- Wesley, (1994).

[7] Garlan, D., Shaw, M., "An Introduction to Software Architecture". *In Advances in Software Engineering and Knowledge Engineering*, I (Ambriola V, Tortora G, Eds.) World Scientific Publishing Company, (1993).

[8] Iglesias, C., Garijo, M., and Gonzalez, J. "A Survey of Agent-Oriented Methodologies". *In Intelligent Agents V - Proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages (ATAL-98), Lecture Notes in Artificial Intelligence*. Springer-Verlag, Heidelberg (1999).

[9] Kendall, E. A., M.T. Malkoun and C.H. Jiang, "Multiagent System Design Based on Object Oriented Patterns", *The Report on Object Oriented Analysis and Design in conjunction with The Journal of Object Oriented Programming*, June, (1997).

[10] Kendall, E. A., P.V. Murali Krishna, Chirag V. Pathak, C.B. Suresh, "Patterns of Intelligent and Mobile Agents," *Agents '98*, May, (1998).

[11] Kolodner, J.,"*Cases based Reasoning"*, Morgan Kaufman Publishers, USA, (1993).

[12] Lopez de Mantaras, R. And Plaza, E., "Case-Based Reasoning: An overview". *AI Communications Journal*, **10**,1, 21-29 (1997).

[13] Nonaka, I. "A Dynamic Theory of Organizational Knowledge Creation. Organizational Science". **5**,1, 14-37 (1994).

[14] Rabarijaona, A., Dieng, R., Corby, O. And Ouaddari, R. "Building and searching an XML-based corporate memory". *IEE Intelligent System.* , 56-63 (May/June, 2000).

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

[16] Shen, W. and Norrie, D.H. ,"Agent-Based Systems for Intelligent Manufacturing: A Sate-of-the-Art Survey" *Knowledge and Information Systems*, **2**,1, 129-156(1999).

[17] 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*, **5**, 2, 30-43 (2001) <http://journal.info.unlp.edu.ar>

[18] Stegmayer, G., Caliusco, M., Chiotti, O. and Galli, M., "Knowledge Component of a Multiagent Distributed Decision Support System". *CLEI Electronic Journal*, **5**, 1, (2002). <http://www.dcc.uchile.cl/~rbaeza/clei>

[19] Turban, E. and Aronson, J., "Decision Support Systems and Intelligent Systems", Prentice Hall, USA, (1998).

[20] Tveit, A. "A survey of Agent-Oriented Software Engineering". *Proc. of the First NTNU Computer Science Graduate Student Conference*. Norwegian University of Science and Technology, May 2001.

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

[22] Villarreal, P., Alesso, M., Rocco, S., Galli, M.R. and Chiotti, O. "Approaches for the Analysis and Design of Multi-Agent Systems". *To be presented in 31 JAIIO - Jornadas Argentinas de Informatica e Investigacisn Operativa*. Santa Fe –Argentina Septiembre(2002).