A Digital Libraries System based on Multi-level Agents

Kamel Hamard[1], Jian-Yun Nie[1],
Gregor v. Bochmann[2],
Robert Godin[3], Brigitte Kerhervé[3],
T. Radhakrishnan[4], Rajjan Shinghal[4],
James Turner[5],
Fadi Berouti[6], F.P. Ferrie[6]

1. Dept. IRO, Université de Montréal
2. SITE, University of Ottawa
3. Dept. Informatique, Univ. du Québec à Montréal
4. Dept. Of Computer science, Concordia University
5. Dept. de bibliothéconomie et science dÆinformation, Université de Montréal
6. Center for Intelligent Machines, McGill University

Abstract
In this paper, we describe an agent-based architecture for digital library (DL) systems and its implementation. This architecture is inspired from Harvest and UMDL, but several extensions have been made. The most important extension concerns the building of multi-level indexing and cataloguing. Search agents are either local or global. A global search agent interacts with other agents of the system, and manages a set of local search agents. We extended the Z39.50 standard in order to support the visual characteristics of images and we also integrated agents for multilingual retrieval. This work shows that the agent-based architecture is flexible enough to integrate various kinds of agents and services in a single system.

1. Introduction

In recent years, many studies have been carried out on Digital Libraries (DL). These studies have focused on the following points:

- the description of digital objects
- organization and processing of multimedia data
- user interface
- scalability of the system
- interoperability
- extensibility

UMDL and Harvest are two examples of such systems. Although not specifically designed for DL, Harvest [Bowman 94] proposes an interesting architecture for distributed DL system. In this system, four types of components are included: client looking for information, information provider providing source information, information gatherer collecting information through the network and information broker matching a userÆs query with a set of documents. The main characteristics of this system are its flexibility for integrating new components and its efficiency for information processing in a distributed environment. UMDL [Birmingham 95] uses an architecture based on agents. An agent is autonomous in the sense that it is able to organize itself

1 Contact persons: K. Hamard and J-Y. Nie, DIRO, Université de Montréal, c.p. 6128, succursale Centre-ville, Montreal, Quebec, H3C 3J7 Canada. email : {hamard, nie}@iro.umontreal.ca

the necessary processing and to ask help from other agents if necessary. The flexibility, scalability and extensibility of the system are further increased. This makes the architecture one of the most attractive for DL.

Nevertheless, several aspects seem to be neglected in the previous studies.

1) An information site, once encapsulated within a service, is directly integrated into the system, and becomes accessible from any other component. There is no further structuring among different services and access control is centralized. As a result, a system is composed of a bag of services at the same level. In a system where the number of services is limited, this does not raise particular problems. However, if a system integrates a great number of services, this single-level architecture becomes difficult to manage.
2) Although the multilingual problem has been mentioned in several systems, no practical solution has been integrated in DL systems.
3) Advanced multimedia processing is another problem that has been studied but not integrated in a large scale DL.

In this paper, we focus on the above problems. We propose an architecture inspired from Harvest and UMDL, but we enhanced it in order to deal with the above problems. An operational prototype has been successfully built.

To address the single-level architecture problem, we propose a multi-level cataloguing of information sites. Apart from traditional indices created from documents within an information site, we also create a higher level catalogue of information sites which stores characteristics of each information site. The control of information sites is shared among a set of global search agents. According to the query of the user, only the most appropriate global search agents will be called. This will reduce much useless calls and network traffic.

The architecture based on agents allows us to add translation services easily into the system, offering multiple possibilities of query translation. Our concern on multimedia data has been concentrated on images and texts. We deal with the indexing of multimedia documents and their retrieval with multimedia queries.

This paper is organized as follows. In the next section, we present the main concepts of our system. In section 3 we describe the different agents. We present several interesting implementation approaches in section 4, before some conclusions.

2. General view of the system

The architecture is inspired from Harvest [Bowman 94] and UMDL [Birmingham 95], and centered on the notion of agent. In our case, an agent is an independent and autonomous entity that provides some service to other components of the system. The autonomy of an agent is important in order to make the whole system extensible and flexible. This fact has been made clear in UMDL system [Birmingham 95].

The architecture is as illustrated in Fig. 1. We notice that at the bottom, a set of databases is integrated into the system. However, they are first encapsulated with a *Local Search Agent* (LSA) before being accessible from other components of the system. A database, together with its LSA is called an *information site*. At a higher level, a set of local search agents is managed by a *Global Search Agent* (GSA). Global search agents are those accessible from other GSAs and Query Agents (QA). Once a query is submitted from a *Query Agent* (QA), the system first selects several GSAs that are the most appropriate to answer the userÆs query according to its requirements. These latter then contact their LSAs to look for relevant documents. The documents found from different databases will be merged before being transmitted to a *Presentation Agent* (PA) which will choose an appropriate presentation strategy to show the results.

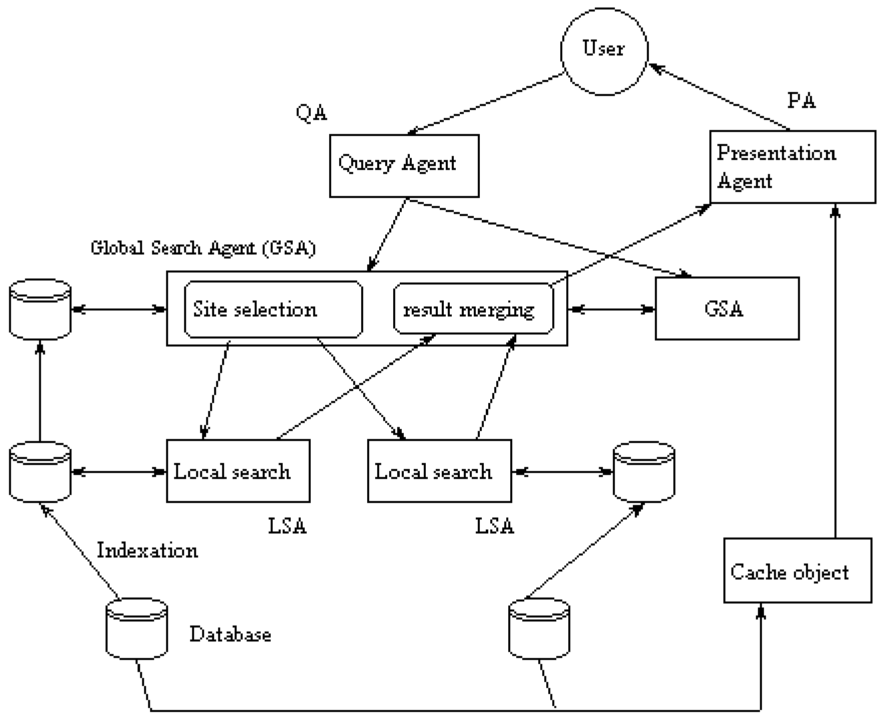

Figure 1: Architecture of the system

We now describe the key concepts in this architecture.

2.1 Multi-level indexing

The distinction between local and global search agents lies in the fact that a local search agent only has a view about the documents stored in the local database. It does not know the other search agents in the system. Its role is to perform a detailed search among the documents stored locally. A global search agent (GSA), on the other hand, does not have detailed knowledge about the documents stored in a local database, but only a synthetic view of them. For example, it knows the main domains (chemistry, medicine, à) of a local database. The idea of distinguishing levels of search agents is very intuitive. In fact, when a human being looks for documents in libraries, he first identifies the libraries where he has the highest chance to find relevant documents. Then a detailed search is proceeded in the identified libraries.

The role of a GSA is precisely to guide the searching to the most appropriate databases. This solution offers several advantages: 1) it avoids useless call to LSAs which are not connected with interesting databases; 2) it allows us to optimize the entire searching by considering the state of

LSAs. For example, if a search agent is busy, the query may be sent to another candidate LSA. This allows us to speed up query processing because useless waiting for answer may be avoided.
To each search agent, a catalogue (either local or global) is created. A local catalogue is much similar to what is offered in other DL and information retrieval systems. It stores indices of documents in the corresponding database. The question now is what a global catalogue stores and how this information may be obtained.

The types of information stored in a global catalogue depends on the kinds of searching users may want to perform. One may think of the following kinds of information:

- The specialization domains of the information site, as well as the richness with respect to each domain;
- The document media type;
- The document languages;
- The physical location (distance) of the LSA as well as performance characteristics of the site or the network link;
- The document quality.

Several kinds of information are easy to obtain. For example, there are known means to detect automatically the physical location of an information site, the medium used in a document (text, image, etc). Some characteristics may also be detected using metadata included in documents such as language and medium. The automatic detection of language is no longer a problem. Several automatic language identifiers have been developed in recent years. For example, [Isabelle 97] uses statistical information to guess the language and coding of a text at a very high accuracy (over 95% if the text is at least a line long). Such a tool may be used to survey the languages used at an information site.

The problem of determining automatically the domains of documents is much more difficult. It is very similar to document classification. This problem has been the subject of various researches in information retrieval. Until now there is no approach totally satisfactory. However, our goal is much less: we do not need a refined class hierarchy for documents. We only intend to identify several main domains for the entire document collection. For example, we are interested to know if a document collection is about chemistry, medicine, computer science, or economy. Our hypothesis is that an information site in DL usually stores documents in a limited number of domains. There are, of course, information sites where documents of any area are mixed up. However, we think that such a case does not occur very often in DL. As a matter of fact, many suggest that DL should follow the organization model of current libraries. This lead us to think that future DL will have many specialized information sites for which it is possible to guess the (large) domain(s) of the documents.

The problem of automatic detection of document domains has been investigated in several IR studies, and recently in DL context [Banerjee 94]. In this latter, the author suggested to detect the domains using a large thesaurus. In a thesaurus, each domain is divided into several sub-domains, and the latter are further divided. A few levels of domains are thus created. A possible characterization of a domain is created by all the concepts or terms included in that domain. Using this characterization, it is then possible to determine if a document is part of the domain by comparing the terms of the domain with those in the documents. The work in [Banerjee 94] has been very encouraging.

Recently, we also investigated this problem with the help of a large terminology base û La Banque de Terminologie du Québec (BTQ) developed by the Office de la langue française of Quebec gouvernment. It contains over 1 million terms classified according to specialization (about 160 general domains). At the top level, the general domains include ôcomputer scienceö, ôpublic buildingö, ôchemistryö, and so on. Our preliminary tests using the BTQ for domain detection have been highly encouraging. A set of scientific abstracts has been used as test material. The result shows that in most cases, the detection is correct. The method is being further refined, and in the near future, a precise evaluation will be reported.

2.2. Multimedia retrieval

Multimedia document retrieval may concern documents that are written each in several media, or documents that are written each in a single medium. Our research has been focused on the second type of documents. So we do not deal with the detection of medium for chunks of document.

In order to retrieve documents in different media, we first have to establish indexing mechanisms for each medium. In our case, a text indexing and three image indexing mechanisms have been integrated. The text indexing is a classical one in information retrieval. The three image indexing methods emphasize each on different visual characteristics of images.

The first index is based on the absolute intensity values of the pixels. The second and third type of index is based on shape characteristic. The research made on the field on image indexing conclude that it is very difficult to construct indexing techniques for all type of images. In this research work, images are classified into two groups: the first group contain all images that are well framed and the second group contain images with complex scenes and the indexing techniques is based on appearance-based methods. In the first group, we know where the objects are located in the image. The shape characteristic is based on the zero crossing calculated from the Marc/Hildreth operator.

In the second group, the shape characteristic is calculated from intermediate representation of the image. The intermediate images are obtained from the Fourier Transform of the complex scenes.

Specialized local search agents and global search agents for images are set up which cares about image indexing and retrieval. At the user interface level, the user has the possibility to submit a query in text (a Boolean expression of keywords). He also has the possibility to submit a query by specifying an example image. He should also choose one of the image indexing methods. The image will be indexed, then compared with images in the image database. The user may also combine the two kinds of specification for a single query. For example, he may specify the author of the image, the type of the object in the image (e.g. house) using a textual specification, and an example image for visual aspects (e.g. a special shape of house). The two kinds of criteria will be merged in a linear way to produce a final similarity of the document to the query (i.e. $S = \alpha S_{text} + (1-\alpha) S_{image}$).

2.3. Multilingual problem

Although this problem has been underlined in many studies, it is not integrated in DL systems. The purpose of our study on this aspect is not to solve all the problems underlying query translation, but to show that such an agent may be easily integrated in our system.

Several approaches may be used for query translation: using a machine translation system [Systran], a bilingual dictionary, or a probabilistic (statistical) translation model based on parallel texts. Recently, we have investigated this problem and compared the three approaches [Nie 98]. In our current system, two simple translation agents (for French-English) have been integrated. Both are based on the bilingual terminology base BTQ. In our future work, we will integrate other translation methods such as the one based on parallel texts and the machine translation systems available in the Internet [Systran].

3. Components of the system

From an architectural point of view, the system is organized as a pyramid of three layers (Fig. 2). At the highest level, there is a unique *Supervisor Agent* (SA). Global Search Agents (GSA) are at the middle layer, and Local Search Agents (LSA) are at the lowest layer. In the top-down direction, each layer represents a decreasing *globality*.

Components in the pyramid correspond to the permanent components of the system. Outside the pyramid, a Query Agent (QA) and a Presentation Agent (PA) may be created dynamically whenever a user is connected and initiates a search operation. These agents are essentially connected to the middle layer (although they should first request the SA for available GSAs in the system). Other agents can be involved in query processing. For example, the figure 2 shows that both the user (via the GUI) and the QA can use the Translation Agent service (TA).

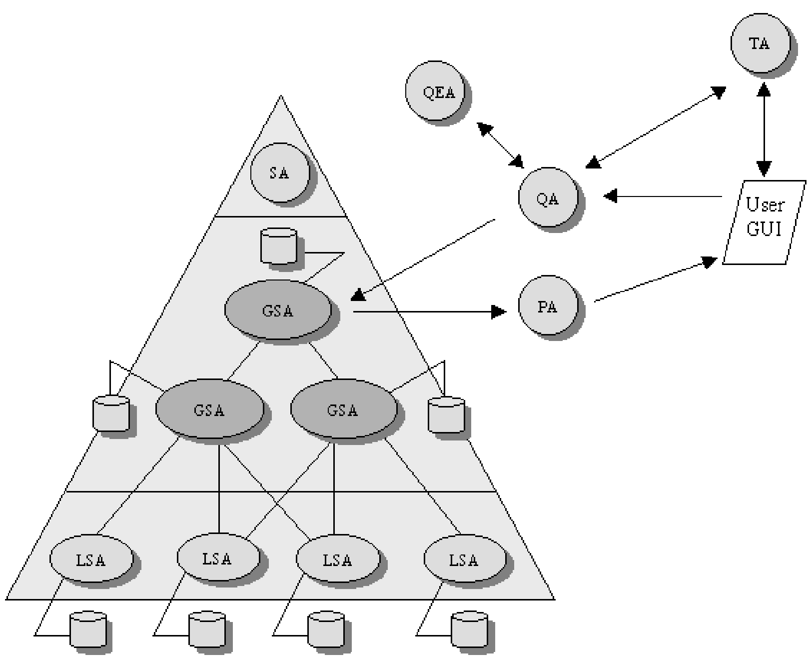

Figure 2: pyramidal organization of agents

We now describe in details of the five different agents that compose our system: Supervisor Agent (SA), Global Search Agent (GSA), Local Search Agent (LSA), Query Agent (QA) and Presentation Agent (PA).

3.1. Supervisor Agent (SA)

This agent is unique and is the first service established in the system. It offers essentially two types of operations: registration and search. The first operation updates the record for available services in the system, and the second operation allows an agent to look for other agents of certain kind. The basic roles of the SA are the following:

- it keeps a registration record for each service;
- it supervises the operations of the integrated components;
- it allows any agent to identify all the other available services in the system.

When a new service is created and integrated into the system, the first operation is to register with the SA. At the same time, the new service is required to provide descriptions of the service, for example, the type of the service, the location, the security information, and so on. Once the service is registered, it may be called by other agents. In our current system the SA keeps track of the following information lists: GSAL (GSA List), LSAL (LSA List), QEAL (Query Expansion Agent List), TAL (Translation Agent List). Figure 3 shows the information managed by the SA.

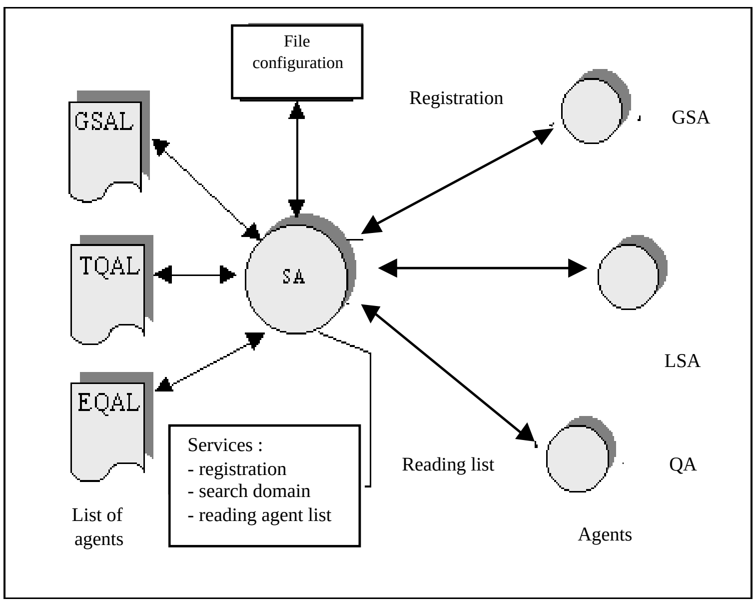

Figure 3: Description of the SA.

Once a local search agent (LSA) is registered with the SA service, a GSA is to be assigned to manage it. The GSA to be assigned is determined according to the LSA’s characteristics. The LSA may request to be associated with a particular GSA. Once the GSA is determined, the LSA will register again with the GSA and the latter will update its global catalogue.

If an anomaly or dysfunction occurs to a component, the SA will put it out of service temporarily. When a service is retracted, the SA will also communicate with the related agents to tell them to update their information.

3.2. Global Search Agent (GSA)

There are a number of GSAs in the system, each interacting with several LSA. Some GSA may be specialized in certain domains (e.g. chemistry); some others may be specialized for a particular medium (e.g. for image indexing or retrieval).

Once a GSA is contacted by a QA, according to the information stored in the global catalogue, it identifies the appropriate LSA. The documents found by the LSA will be merged by the GSA. At this stage, it is also possible to detect document duplication. This problem has been dealt with in [Shivakumar 95]. This problem is not dealt with in our current prototype. The strategy used in our current system is a simple merging according to the value of (normalized) similarity of the documents to the query.

3.3. Local Search Agent (LSA)

This agent performs a real document retrieval on a local document base. A LSA may be connected to one or several GSAs. The second case occurs if GSAs have been classified according to different aspects. As for example, there may be one GSA for image retrieval, and another for the domain of medicine. Then a LSA specialized in medical images may be connected to both GSAs.
Apart from retrieval, an LSA may also perform several other operations. For example, it may compress a document before the latter being sent to the user (Presentation Agent).

3.4. Query Agent (QA)

A query agent (QA) works in tandem with a user interface. Once a query is created by the user, it is first transmitted to the QA for further processing. A QA executes two different types of processing:

- appropriate GSAs identification

When a query is formulated by the user, the first operation consists in contacting the SA to request the available GSAs according to the criteria provided by the QA. For example, the QA may request GSAs for documents in the domain of ôchemistryö. Then the SA identifies the corresponding set of GSAs.

- Query transformation

A userÆs query may be directly sent to the GSAs for document retrieval, or it may be transformed. For example, the user may choose to use a translation service (by a Translation Agent û TA) or a query expansion service (by a Query Expansion Agent û QEA) to add related terms to the original query.

The processed query will be finally transformed into the Z39.40 format.

3.5. Presentation Agent (PA)

A PA should decide the presentation strategy of the retrieval results according to the choice of the user or according to the user environment. For example, the retrieval results may be presented as a simple list of references or a set of icons. Depending on the environment of the user interface, some transformations may be necessary before the documents may be shown on the screen. For example, if the userÆs environment is unable to display image in a certain format, another format needs to be created dynamically. In the current system, the presentation agent is simple: it transforms the original (text or image) documents into HTML in order to be displayed in an Internet browser. More sophisticated presentation agents are under development.

The five agent types we have previously presented are the basic agents for our system. Several optional agents have been, or will be, added to the system such as:

- Translation Agent (TA): to translate a query from one language to another;
- Query Expansion Agent (QEA): to expand a query by related terms;
- User profile Management Agent (UPMA): to manage the userÆs profile.

A user profile may also help in query expansion. For example, it may provide information about the working domains of the user (that could be considered as the default domains for the query). This may help to choose the relevant terms to be used in query expansion. It also provide helpful hinds for the selection of appropriate information sites or GSAs for the userÆs queries.

4. Implementation strategies

The current system is integrated with the WWW environment. Different communication protocols have been used at different levels of the system.

- TCP/IP is the general protocol used for communication between different components (between a client and a server).
- An extended version of Z39.50 [NISO 95] is used as the communication standard between agents. The extension is made for images according to the Dublin Core metadata [Dublin Core].

There are both image retrieval agents and text retrieval agents in our system. The protocol Z39.50 has been originally designed for textual documents. We extended the protocol to take into account several image features. A new attribute, *Image index* is created. A set of unique references has been defined to represent visual characteristics of images (e.g. color, size, etc).

For implementation, we used a distributed object system û HORB [Hirano]. All the services of the system may be shared due to the creation of a common bus, as illustrated in Fig 4.

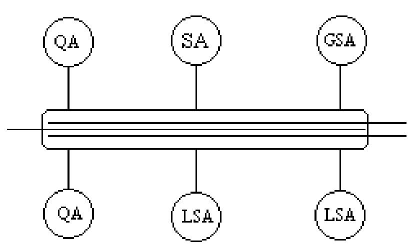

Figure 4: Agents connected by a HORB bus.

A particular point we want to stress is mobility. Data transfer usually takes considerable amount of time in comparison with processing time of an agent. In order to reduce data transfer, some services are migrated toward data. For example, for a presentation agent, the amount of data to be presented is very big. If the service is created at a different site from the data, all the documents found should be transferred to the presentation agent. However, only a small part of the documents will be viewed by the user. So a major part of the data transfer is useless. If we migrate the presentation agent to the data site, then all the data transfer from the database to the presentation agent is local. The distant transfer to the user interface only concerns the documents that the user actually wants to view and some neighbor documents (in order to construct a local document cache to accelerate the presentation). This approach may greatly increase the efficiency of query processing and reduce much useless data transfer through the networks.

There are also cases where mobility is a necessity rather than a choice. This case occurs typically when a server (e.g. a database server) wants to use a service that is not available locally. In this case, it has to use a service migrated from another server. As an example, we can mention data compression and encryption. So mobility is also a means to extend the capability of a service of a particular server or agent.

5. Processing of a query

Let us now give a complete example of query processing in this section (Fig. 5).

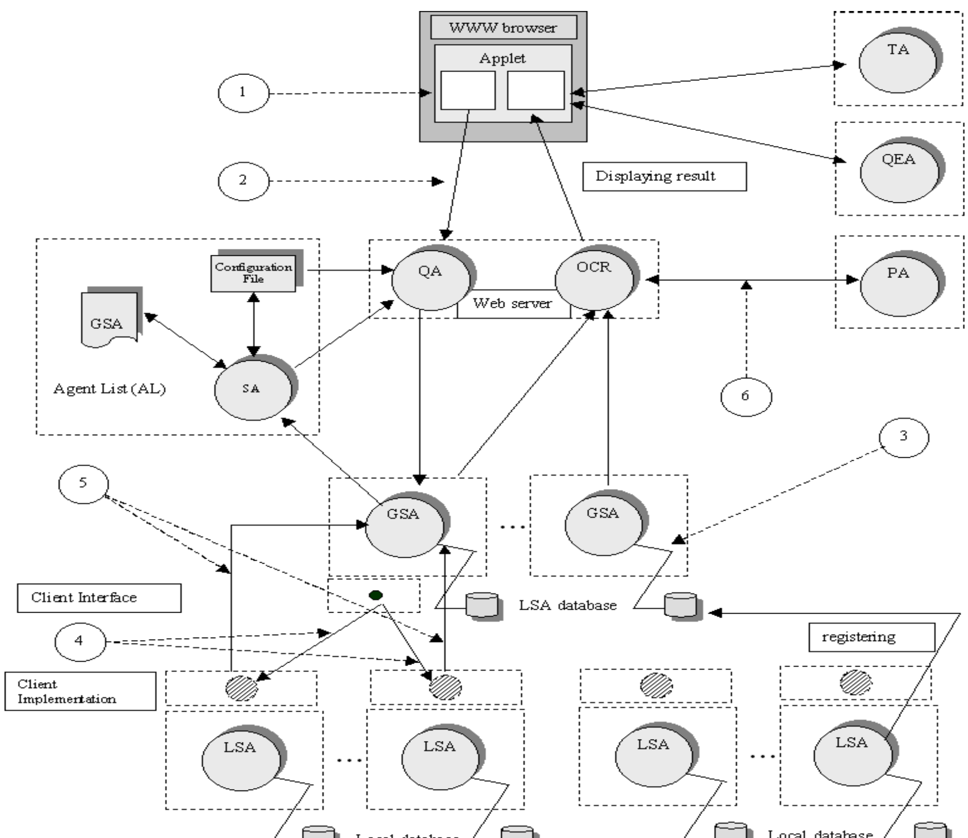

Figure 5. Processing flow of the example.

Step 1: Query submission by the user

The interface allows the user to specify a query in two manners: by a text, or by an example image. The example image is one of the typical images stored in a default example database, or the user may choose any other image that he knows (typically, the user may specify one of the previously retrieved images as an example image). The default example images are typical for different visual characteristics (e.g. luminosity, shape, and so on). The design of the interface follows the model of Willow [Willow]. Figures 6 and 7 show respectively a query specification by texts and by an example image in the case of a retrieval within an architectural photography database.

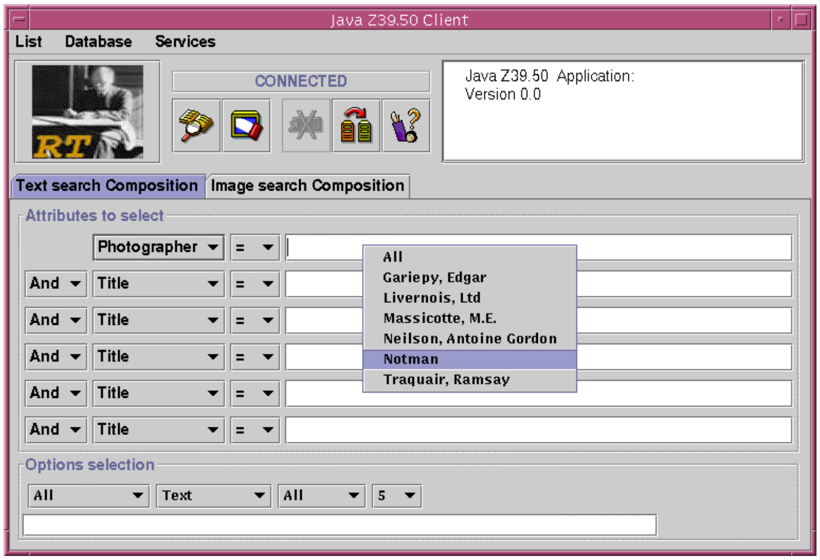

Figure 6: The textual part interface.

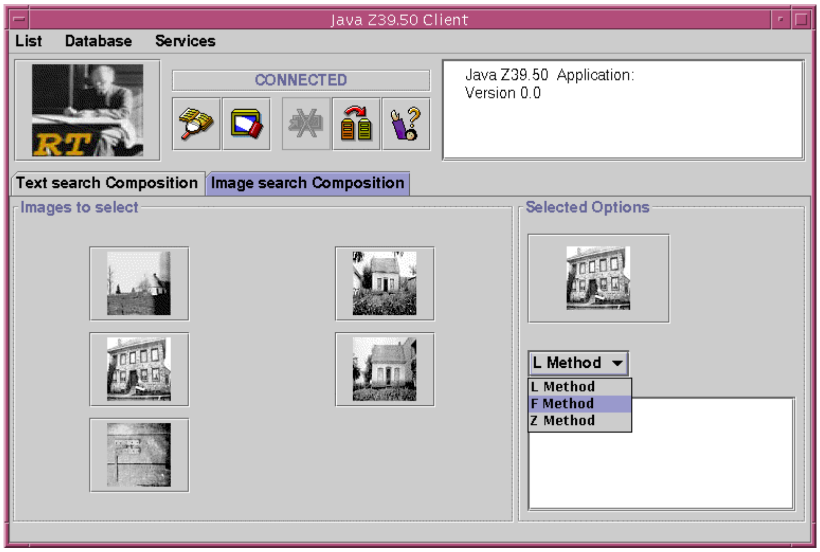

Figure 7: the visual part interface

A query object is created from the user specification. This object describes all the aspects that the user specified in a format compatible to Z39.50. The query object is then sent to the QA.

Below is an example of query object (OBQ) sent to a QA:

```
And ( Notman [1,1003], And ( img_1 [ media = ref_image , index =
ref_index ])
```

Here the user wants to retrieve the photos taken by `Notman` and respecting the visual characteristic of the image `img_1` which is composed of two parts: the first specifies the type of medium (in this case, `image`); the second specifies the index of the example image.

Step 2: the query object (OBQ) is sent to QA

Once the QA receives an OBQ, it contacts the SA to know the list of the GSAs available, and decides which GSAs to contact. The GSA to be contacted for the above example query is a GSA which is specialized in architectural photography.

Step 3: the GSA receives the OBQ and contacts the LSA for the search.

When the GSA receives the OBQ, it determines the appropriate LSAs under its control to be contacted. For this example, two LSA will be called: one for text retrieval, and the other for image retrieval (on the same document base).

Step4: the LSA receives the query object and contact the Z39.50 database server for the search.

The Z39.50 server makes a search in the database and return the results to the LSA, which then transmits them to the GSA.

Step 5: The GSA receives lists of results from LSAs.

At this stage, a linear combination is used to combine the lists from different LSAs (i.e. $S = \alpha S_{LSA1} + \alpha S_{LSA2}$ with an equal α for all the LSAs currently). Ideally, we would want to determine the coefficient α according to the quality of the documents managed by a GSA and its richness with respect to each domain or medium. However, the assessment of quality is a difficult issue that is not yet incorporated in our current system.

Once a set of documents identified, it is sent to the object cache agent in order to be browsed by the user.

Step 6: The object cache uses the PA to format the document required by the user before displaying it.

The presentation agent receives the list of document references. If the user choose to view a document, the document is transformed into HTML format, then shown in the user interface.

5. Conclusions

In this paper we proposed an agent-based architecture based distributed multimedia DL. The organization principle is similar to those of Harvest [Bowman 94] and UMDL [Birmingham 95], but we extended it with respect to the following aspects:

- Search agents are distinguished into local and global ones. This distinction makes it easier to manage the search agents in the system (especially when the number of local search agents grows up). It also allows us to avoid many useless calls to LSAs for query processing, and reduce network traffic load.
- We integrated several simple agents for query translation and multimedia retrieval. The integration of these agents showed that the architecture is flexible enough to integrate various agents easily.

We used Z39.50 as a communication protocol within the system. This facilitated much the system integration. However, we have had to extend the protocol with respect to images. This direction has also been taken by several other research groups.

The research is still ongoing. We are integrating more (sophisticated) agents into the system. An agent for user profile management will be integrated to maintain a profile for the user. This profile will be useful in determining the domain(s) in which the userÆs query is, and in expanding the userÆs query. A more sophisticated presentation agent will also be added. A user will be able to choose among a set of presentation strategies.

Acknowledgement:
The authors would like to thank Mrs.K.Pathy for her participation in this project. We would also like to thank the Office de la Langue Française for providing us with the BTQ.

References:
[Banerjee 94] Sujata Banerjee and Vibhu O. Mittal., On the Use of Linguistic Ontologies for Accessing and Indexing Distributed Digital Libraries. D-lib Magazine 1994.

[Birmingham 95] William P. Birmingham, An Agent-Based Architecture for Digital Libraries. D-lib Magazine, July 1995.

[Bowman 94] C. M. Bowman et al.. The Harvest Information Discovery and Access. Technical Report CU-CS-732-94, Department of Computer Science, University of Colorado, Boulder July 1994.

[Dublin Core] http://linnea.helsinki.fi/meta/oclc/index.htm

[Hirano] HIRANO Satoshi, HORB - a distributed Java package. Tutorial document. http://ring.etl.go.jp/openlab/horb

[Isabelle 97] P. Isabelle, G. Foster, P. Plamondon, The SILC project: A system for language and coding identification, http://www-rali.iro.umontreal.ca/ProjetSILC.en.html, 1997.

[Nie 98] J.Y. Nie, P. Isabelle, P. Plamondon, G. Foster, Using a probabilistic translation model for cross-language information retrieval, *Sixth workshop on Very Large Corpora*, Montreal, pp. 18-27, 1998.

[NISO 95] National Information Standards Organization. Z39.50 : Information Retrieval (Z39.50) : application Service Definition and Protocol Specification, 1995.

[Shivakumar 95] N. Shivakumar and H. Garcia-Molina, SCAM: Acopy detection mechanism for digital document, Digital LibraryÆ95, 1995, http://csdl.tamu.edu/csdl/DL95/papers/shivakumar.ps

[Systran] http://babelfish.altavista,digital.com/

[Willow] http://www1.cac.washington.edu/willow/home.html