Distributed Multimedia Indexing and Optimal Resources Utilization: An Implementation Based on Metadata, Context and Usage

Mihaela Brut, Dana Codreanu, Ana-Maria Manzat, Florence Sedes
Universite de Toulouse – IRIT – UMR 5505
31062 Toulouse, France
{Mihaela Brut, Dana.Codreanu, Ana-Maria.Manzat, Florence.Sedes}@irit.fr

***ABSTRACT*:** *Effective and flexible solutions for enabling reduced resources consumption and handling formats heterogeneity are essential in multimedia management systems. The reduction of the resources consumption for indexing can be achieved in two ways: by limiting the multimedia content transfer over the network, and by employing only the most appropriate algorithms, only over the relevant content. Multiple formats are developed for the description of resources, contents and applications, which are not necessary interoperable. A solution for coping with this heterogeneity is to create an integrative model for the description of any resources. We present in this paper the solution implemented in the context of the LINDO project for indexing multimedia content within a distributed system. Such solution addresses these two points by reducing as much as possible the resources consumption through (1) a distributed indexing technique that avoids multimedia transfer; (2) a flexible mechanism for selecting the indexing algorithms to be employed on each remote server, according to the multimedia content characteristics, its acquisition context and user queries history. Our proposal is based on generic description models associated to multimedia content, indexing algorithms and servers.*

Keywords: Multimedia Indexing, Metadata, Indexing algorithm, Multimedia resource description

Received: 14 August 2011, Revised 30 September 2011, Accepted 8 October 2011

1. Introduction

A huge quantity and diversity of multimedia contents is generated in multiple domains (video surveillance, medical records, libraries), with different acquisition contexts. The management of this data raises two major problems: (a) the heterogeneity of description formats and (b) the big resources consumption.

The *heterogeneity of description formats* concerns two aspects:

- Indexing algorithms diversity and heterogeneity: there is a large palette of indexing algorithms having different performances, purposes and constraints. In a multimedia information system it is not desirable to execute all available indexing algorithms on all multimedia content; these will (1) overload the system and (2) produce metadata that might never be used.

- Multimedia metadata heterogeneity*:* the huge number of existing formats, most of them developed for a certain application context by different organizations using different schemas and languages raises a serious interoperability issue.

From an operational point of view the problem of *resources consumption* is how to efficiently manage the huge volume of content and the diversity of the content and of the content processing techniques in the context of a large scale information

system in order to provide users with a high quality multimedia experience.

This paper presents a solution to these two problems, by taking into account the multimedia acquisition context and the recurrent usage of the system. The idea is not to provide another information retrieval model or indexation engine but to integrate existing models and engines for indexing and retrieval multimedia contents. Our solution provides:

- A *multimedia metadata generic approach.* We propose an integrating model for the metadata associated to content and for the indexing algorithms descriptions;

- A *dynamic indexation process* that is accomplished in two steps: (1) at the acquisition moment and (2) at the query execution moment. The indexing algorithms are selected in function of the system usage, some execution context elements and user's access rights.

- A *technique for improving the dynamic indexing process* by introducing new filtering criteria for the indexing algorithms selection, such as the acquisition context, and the users' queries. This technique reduces the indexing amount, and also dynamically establishes the most relevant indexing algorithms according to the changes that occur in the acquisition context (e.g. weather or luminosity variations) and the users' queries history.

These solutions are implemented within a *generic framework*, developed in the context of the LINDO[1] (Large scale distributed INDexation of multimedia Objects) ITEA2 project, which is intended to guide the formalization and the development of a distributed multimedia information system, while favoring a reduced resources consumption, in terms of data transfers over the network, storage and CPU utilization and a management of the diversity and the heterogeneity of the resources involved in the multimedia system (e.g., contents, algorithms, hardware, software).

In the remainder, the paper presents a synthesis of the existing works related to the problems addressed. The Section 2.1 offers a state of the art of the metadata standards and vocabularies. The Section 2.2 provides an overview of the solutions adopted by some industrial projects that address the multimedia management problems. Then, the LINDO generic architecture is detailed in Section 3. The description models developed within the project are presented in Section 4. How these descriptions are used in order to develop the dynamic selection of indexing algorithms is explained in Section 5. In Section 6 a solution for the optimization of the indexing process is proposed. A concrete implementation of the architecture and some utilization examples are presented in Section 7 Conclusions and further work directions are exposed in the end of the paper.

2. Related Works

Inside a distributed multimedia information system, the indexing process is managed through an indexing engine that includes multiple indexing algorithms. The indexing could be accomplished by applying to each multimedia content all these algorithms, or just a customized subset. In addition, a customized indexing could be realized by combining specific algorithms into some defined combinations. Also, the indexing could be accomplished in real time or off-line, at a central server level or distributed at the remote server's level. The metadata associated to algorithms or content can be attached to the resource or it can be stored separately in a distributed or centralized database. It can be generated: manually (user annotation), semi-automatically (annotation tools plus user refinement) or automatically (indexing algorithms).

In the following, we present a state of the art of some popular metadata standards and vocabularies and the criteria that we used to compare them. Further, for some representative projects related to distributed multimedia information systems, we briefly present: the main objective, the adopted architecture and indexing mechanism and the solution used for the metadata management. This allows us to do some comparisons with LINDO approach in order to emphasize its advantages.

2.1 Standards and Vocabularies for Multimedia Metadata Management

Metadata, defined as: 'data about data', aims at facilitating the access, management and sharing of large sets of structured/ unstructured data. Many researchers redefined the term in accordance with their purposes. The sense that we use throughout these paper is: digital data that describes resources, digital or non-digital [17].

[1]http://www.lindo-itea.eu

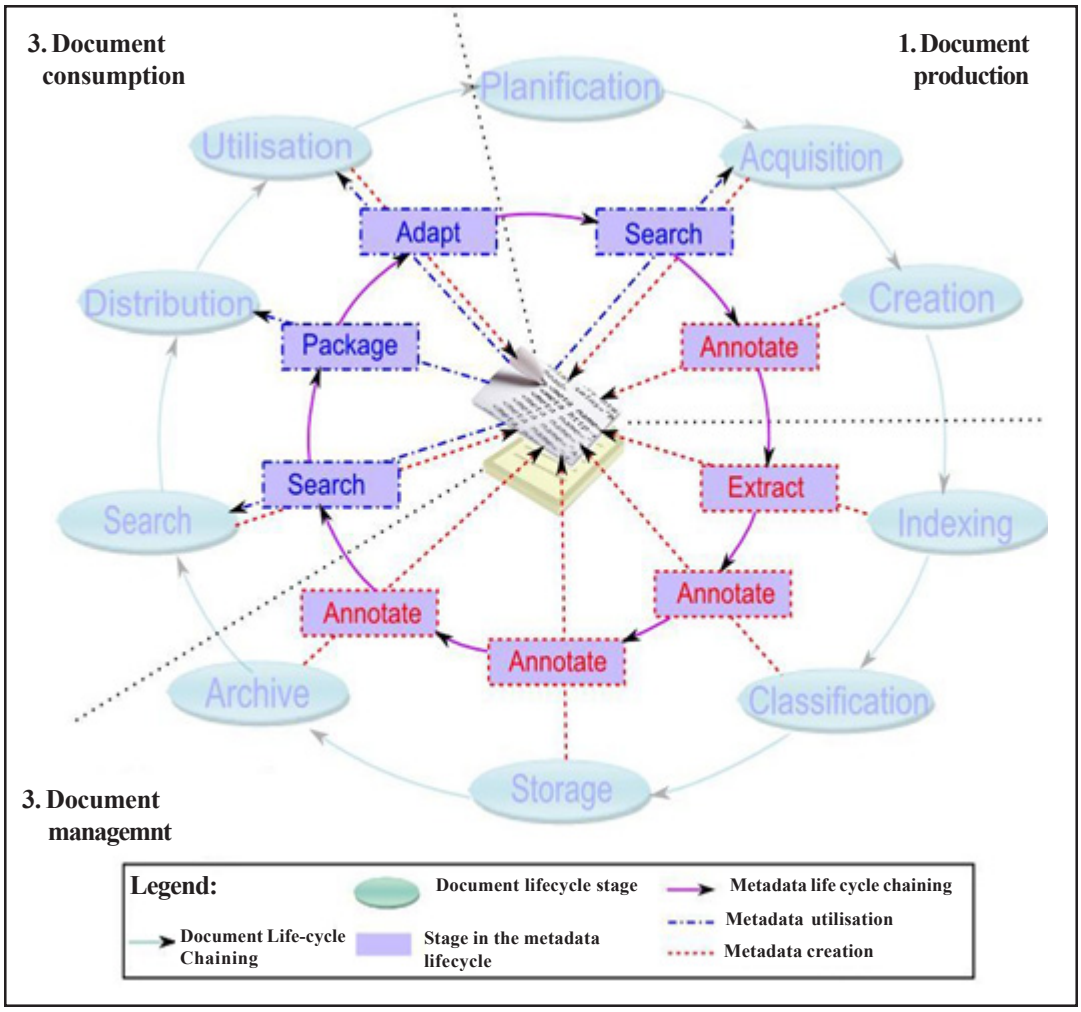

Figure 1. Multimedia and metadata lifecycle

Within multimedia information systems, metadata is playing a central role for the access, retrieval and delivery of multimedia objects. In the following, we present the main advantages in using metadata, and then we provide some metadata classifications in function of the most important criteria. After that, we utilize a part of these criteria in order to do o comparison of some representative metadata standards.

2.1.1 Metadata Utility

Metadata is an essential element for multimedia because it can be the support for Multimedia life-cycle management, Multimedia's content structure description and organization, Multimedia content description and Controlled vocabulary [18].

The information resulted during object's lifetime (across processes dealing with content acquisition and creation, production, indexing, search, distribution etc.) can be captured as metadata. The Figure 1 illustrates the multimedia document life cycle in relation to the metadata document life cycle. The metadata has a central role in the life cycle of the multimedia document. Each stage of this life cycle (i.e., the exterior circles in the Figure 1) interacts with or creates metadata. The interior circle of the Figure 1 represents these interactions (e.g., when creating a document it can be annotated with some information like creation date, author etc.; these annotations are captured in the metadata document). The red boxes (Annotate, Extract) represent processes that produce metadata and the blue ones (Search, Adapt, Package) represent processes that use metadata. During some steps of multimedia document life cycle, metadata can be used and enriched (e.g., Utilization).

Let us take the example of this article. The first thing we did when planning this article was to decide its subject and we

determined what we needed in order to write it. Once this **planning** stage was over we started to search the information that we might have needed for the creation of the document, such as the related work articles that we referenced. At this **acquisition** stage, a search is executed on the metadata that is associated to the different articles. Once the articles were retrieved the writing of the article began (**creation** process). During this step some administrative metadata are created, such as the author, the creation date, the key words, and also the structure metadata. When the article is ready, the **indexation** phase is reached and some supplementary metadata are extracted from the content of the article, e.g., the articles referenced. After the indexation, the document is classified. This **classification** is generating also some classification metadata, such as the subject, the category. The **storage** and the **archive** stages also generate metadata of the article, e.g., the usage rights. Because this document is stored with the others document in the system, it can be at its turn **searched**, and thus its metadata are used in order to see if this document is relevant to a query. At the **distribution** stage, the metadata are used for the packaging of the document and of all the other media that are needed in order to properly use the document. When the article is displayed to a user, the metadata are used in order to adapt it to the user's preferences and context. This utilization of the article ends the multimedia document life cycle. As we could see the metadata has a real central role in the creation, the management and the consumption of the multimedia document, by being produced and used at each stage of this life cycle.

2.1.2 Types of metadata

There are many ways to classify metadata because of the large number of functions that they can accomplish and the big number of elements and domains that it allows to describe. In the following, we present some classifications found in the literature according to:

2.1.2.1 The concerned domain or activity[11]

- *Domain specific metadata*: Digital library (e.g., Dublin Core, METS), E-learning (e.g., LOM, GEM), E-Government (e.g., e-GMS), Broadcasting (e.g., TV-Anytime, P/meta), Movie Industry (e.g., SMPTE), Video Streaming (e.g., Youtube), Art collections (e.g., VRA Core), etc.
- *Generic metadata* (e.g., MPEG-7, MPEG-21, MPEG-A, WSMO, WSDL)

2.1.2.2 The functions that they support [19]

- *Descriptive metadata*: used for discovery and interpretation of the resource (e.g., Dublin Core);
- *Administrative metadata*: is used for managing the resources and contains information like technical metadata, rights management metadata or preservation metadata (which contains information needed to archive and preserve a resource) (e.g., METS);
- *Structural metadata*: describes the logical or physical relationships between the parts of a compound object (e.g., a book consists of a sequence of chapters, a TV serial is composed of a sequence of episodes) (e.g., METS);

2.1.2.3 The elements that they enable to describe [20], [21]

- *Identification metadata:* contains information like: ID, titles etc.
- *Production metadata:* offers information related to the creation of the content (e.g., location and time of the capture, producer);
- *Rights metadata:* refers to the rights holders of the content and at the permitted use, modification, distribution etc. (e.g., a link to a license like Creative Commons, a detailed description of the rights concerning every segment of a content);
- *Publication metadata:* describes previous use of content (e.g., broadcasting, syndication) and related information (e.g., contracts, revenues);
- *Process-related metadata:* contains information describing the steps in the lyfe-cycle of a content (e.g., capture, digitization, encoding, editing);
- *Content-related metadata:* describes the structure of the content (e.g., shots, scenes) and related information (e.g., objects detected);
- *Context-related metadata:* comprises information about the context in which the content was captured and used;
- *Relational/enrichment metadata:* describes links to other related external sources (e.g., multimedia content, services, textual information);

2.1.2.4 The dimensions of the knowledge contained in the standard [22]

- *Services dimension*

- *Content dimension*
- *Hardware dimension*
- *User dimension*
- *Community dimension*
- *User generated metadata dimension*

2.1.2.5 Openness
The metadata description can be developed and published by a nonprofit organization, meaning that it can be used free with no constraints (*open* standard or description) or it can be *proprietary* (it can be used only with a license)

2.1.3 Multimedia Metadata Standards
In the last years the technological development determined an explosion of the multimedia content volume and diversity. Implicitly there is a growing need for analyzing, retrieving and delivering content. In order to support these three functions and to provide to users a better multimedia experience, the need for multimedia metadata formats emerged (compared with the beginnings of multimedia when the need was for coding standards).

Multimedia standards and descriptions represent a key element in enhancing interoperability between systems, favoring technical evolution and the business growth (e.g. MP3, JPEG, ID3). However they generate a problem of interoperability between standards also and we are interested in interoperability in a metadata context. While the coding standards concern a low signal processing level, metadata aims to provide a more semantically description of the multimedia content and to span over the entire multimedia life cycle in terms (see section 2.1.1). Thereof it is very difficult to have a generic metadata standard, independent of the application type. This is the reason why many organizations developed their own standards in concordance with their needs. In the following we present a table that illustrates a synthesis of the main characteristics of some popular multimedia metadata standards. The Table 1 contains a general presentation of some representative descriptions used in multimedia domain. In Table 2, we selected some criteria that we judged the most relevant for a more pursued comparison of metadata.

All this information could serve in the analysis phase of a multimedia information system development to answer to the question 'For some requirements and constraints what format(s) is (are) better to use?'

	Standardization Body	Publication Year	Description of	Application Domain	Level	Purpose	Schema definition	Producibility	Applications (extentions) language	Openness
Dublin Core (DC)	ISO/NISO	1998	Any multimedia content	Any	High	Describe and catalogue most resources	XMLS, RDF/S	Mainly manual	Dublin Core Application Profiles[2]	Open
MARC	Network Development & MARC Standards Office	1999	Any digital library object	(digital) Libraries	High	bibliographic information about textual materials, computer files, maps, music,visual materials, and mixed materials	XMLS, MARC 21	Manual	MARC 21 to MARCXML Conversion, MARCXML to DC Conversion, MARCXML Validation	Open
VRA Core	VRA Data Standards Comitee	1996	Image	Art collections	High	description of images of art works	XMLS	Mainly manual	-	Open
TV-Anytime	ETSI	2003	Video	(digital) Broadcasting	High and low	description of Radio and Television programs	XMLS	Manual and automatic	myTV project[3], Java API[4]	Open

[2]http://dublincore.org/usage/documents/profile-guidelines/

[3]http://www.hitech-projects.com/euprojects/mytv/

[4]http://www.bbc.co.uk/opensource/projects/tv_anytime_api/

	Standardization Body	Publication Year	Description of	Application Domain	Level	Purpose	Schema definition	Producibility	Applications (extentions) language	Openness
METS	Digital Library Federation	2001	(digital) Libraries	Any digital object	High and low	encoding metadata necessary for the management and exchange of digital library objects	XMLS	Mainly manual	METS Java Toolkit[5]	Open
MPEG-7	ISO/IEC	2001	Any	Any multi media content	High and low	provides a rich set of standardized tools to describe multimedia content	XMLS	Manual and automatic	IBM Annotation Tool[6]	Proprietary
Youtube	-	-	Video streaming	Videos	High	description of video content	XMLS	Manual and automatic	Youtube Data API[7]	Proprietary
MPEG-A	ISO/IEC	2007	Any	multimedia applications	High and low	standardized specifications for multimedia applications and related software implementation	Requirements list	Manual	-	Proprietary
MPEG-21	ISO/IEC	2001	Any	Any digital item	High and low	framework describing two concepts: Digital Item (any digital resource) and User	XMLS, DIDL	Manual and automatic	MPEG-21 Reference Software[8]	Proprietary
SMPTE	SMPTE	2004	Movie industry	Audio-visual	High and low	interoperability and traceability within the processes in the movie chain production	Tabel, XSD	Manual and automatic	-	Proprietary
WSMO	W3C	2005	Any	Web Services	High	semantic description of Web services in order to enable their automatic discovery, selection, comp-osition, mediation, execution, etc.	Meta Object Facility (MOF)	Manual and automatic	WSMO Studio[9]	Open
WSDL	W3C	2001	Any	Web Services	High and low	description of Web services and their network bindings	XSD, XMLS	Manual and automatic	Web Services Description Language Tool[10]	Open
	-	2007	Any	Any mm content	High	formal semantics for multimedia annotations by mapping MPEG-7 elements to an ontology	OWL	Manual and automatic	Java API[11]	Open

Table 1. Multimedia description formats overview

[5] http://hul.harvard.edu/mets/

[6] http://www.research.ibm.com/VideoAnnEx

[7] https://developers.google.com/youtube/

[8] http://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_IEC_21000-_2008_Reference_Software/

[9] http://www.wsmostudio.org/

[10] http://msdn.microsoft.com/en-us/library/7h3ystb6%28v=vs.71%29.aspx

[11] http://multimedia.semanticweb.org/COMM/api/

Comparison criteria	Dublin Core	EXIF	MARC	VRA Core	TV - Anytime	METS	MPEG - 7	You tube	MPEG-A	MPEG - 21	SMPTE	WSMO	WSDL	COMM
Descriptions associated to														
Content														
Hardware	x	x	x	x	x	x	x	x	x	x	x	-	-	x
User	-	-	-	x	-	-	-	-	-	-	-	-	-	-
User generated metadata	-	-	-	-	-	-	-	-	-	x	-	-	-	-
Application	-	-	-	-	-	x	x	x	-	-	x	-	-	x
Target users	-	-	-	-	-	-	-	-	x	x	-	x	x	-
Professionals														
Untrained users	x	x	x	x	-	x	x	x	x	x	x	x	x	x
Metadata Type	x	x	-	x	x	-	x	x	-	x	-	-	-	-
Identification														
Creation and localization	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Rights	x	x	x	x	x	x	x	x	x	-	x	x	x	x
Content description	-	-	x	x	x	x	-	x	-	x	x	-	-	x
Technical description	-	-	x	x	x	x	x	x	x	-	-	-	-	x
Historical	-	x	-	-	-	x	x	-	x	-	x	-	-	x
External information	-	-	-	-	-	-	-	x	-	-	x	-	-	-
Extensibility	-	-	-	x	-	x	-	x	x	x	-	x	x	x
Controlled vocabulary	-	-	-	x	-	x	x	x	x	x	x	x	x	x
Serialization	-	-	x	x	x	x	x	-	-	x	x	x	x	x
XML	x	-	x	x	x	x	x	x	-	x	-	-	x	-
Binary	-	x	-	-	-	-	x	-	-	x	x	-	-	-
Storage														
Internal	-	x	-	-	-	-	-	-	-	-	x	-	-	-
External	x	-	x	x	x	x	x	x	x	x	-	x	x	x

Table 2. Multimedia Metadata Standards Comparison

In the last part of the state of the art we present an overview of some representative projects that have as purpose to develop multimedia information systems. We show how these projects address the problems exposed in the introduction of this paper.

2.2 Multimedia Information Management Systems

The management of multimedia content is an important research topic in the last years. Many projects were developed in order to deal with the different problems related to this topic: storage, indexation and retrieval. In the following we present some representative projects by emphasizing their architectural, indexation, and metadata management solutions.

The CANDELA project[12] (Content Analysis and Network DELivery Architectures) proposes a generic distributed architecture

[12] http://www.hitech-projects.com/euprojects/candela

for video content analysis and retrieval [3]. Multiple domain specific instantiations are realized (e.g., personal mobile multimedia management [4], video surveillance [5]. The indexation is uniformly accomplished in all remote servers and managed at the central server level and the resulting MPEG-7 based metadata can be distributed over the network.
However, the indexation algorithms are a priori selected and pre-installed.

Within its peer-to-peer architecture, the SAPIR project [6], [7] (Search on Audio-visual content using Peer-to-peer Information Retrieval) employs three specialized indexing servers (for images, texts and audio-visual contents) where each distributed peer sends its ingested content in order to be indexed. Both the multimedia content and the metadata issued from indexation are stored on the same peer, while the user query is executed over the distributed peers. SAPIR project does not adopt a uniform metadata model, but keeps the original metadata formats obtained during the indexation process. The equivalences among different metadata formats are handled during the retrieval process, based on a probabilistic-based algorithmic solution.

The WebLab project[13] proposes an integration infrastructure that enables the management of indexation algorithms as web services in order to be used in the development of multimedia processing applications [8]. These indexing services are handled manually through a graphical interface. For each specific application a fixed set of indexing tools is run. The obtained XML based metadata is stored in a centralized database.

The VITALAS project[14] (Video & image Indexing and retrieval in the Large Scale) capitalizes the WebLab infrastructure in a distributed multimedia environment [9]. The architecture enables the integration of partner's indexation modules as web services. The multimedia content is indexed off-line, at acquisition time, on different indexing servers. No selection of indexing algorithms based on user query is done.

The MUSCLE network of excellence[15] (Multimedia Understanding through Semantics, Computation and LEarning) developed a centralized framework, where the indexing algorithms are managed at the central server level, in order to be used for distributed indexation. Within the project a large set of different multimedia indexing algorithms has been developed (e.g. object recognition, content analysis, unusual behavior detection, movie summarization, human detection, speech recognition). The project didn't focused on the metadata management so no model or storage and use solution is proposed.

The KLIMT project (Knowledge InterMediation Technologies)[10] proposes a Service Oriented Architecture middleware for easy integration of heterogeneous content processing applications over a distributed network. The indexing algorithms are considered as web services. The user query is limited to pre-defined patterns that match a set of rules for the algorithms' execution sequence. After such a sequence selection, the content is analyzed and the metadata is stored in a centralized database. KLIMT project adopts an XML-based solution for managing multimedia metadata. [23] have proposed an indexing algorithm model that describes the input and the output of the algorithm. Moreover, they have proposed a chaining algorithm based on the input/ouput data types. Nevertheless, the purpose of the project isn't the metadata so no generic model or management technique was proposed.

In [2], a distributed image search engine based on mobile software agents is proposed. In order to deal with the metadata generated by the indexing algorithms, the authors propose two architectures: one centralizing the index and the other one distributing the indexes on the remote servers. The indexation is accomplished with a fixed set of indexing algorithms that are implemented as mobile agents. These agents migrate from one site to another in order to extract different multimedia features. Thus the multimedia contents are not transferred over the network. . In the case of this work, the metadata that is extracted from the content is modeled as features vector. The main features took into consideration are the low level ones (e.g., color histogram, texture).

The CAM4HOME project[16] proposes a metadata based contend delivery framework that allows end users and commercial content providers to create and deliver rich multimedia experiences. In order to accomplish this objective a platform based on a Service Oriented Architecture was developed. The concept behind the platform is Collaborative Aggregated Multimedia (CAM) which refers to composition of different multimedia objects and related services into a bundle. For example, three commercial

[13] http://weblab-project.org/

[14] http://vitalas.ercim.org

[15] http://www.muscle-noe.org

[16] http://www.cam4home-itea.org

actors, one who provides online poker game services, one who provides poker related videos and one who provides poker related websites can aggregate their services and contents and offer a better experience to users through the CAM platform. The CAM Metadata Framework aggregates content, user, application, service related metadata (from different sources and in different formats) in a unique RDF based model developed within the project. The adaptive and personalized delivery of the content is based on that metadata. The indexation techniques are not part of the scope of the project.

A global overview of these projects is provided in Table 3.

System	Architecture	Indexation	Metadata management
SAPIR	Peer to peer	Uniform and fixed indexation at each peer	No uniform metadata model
CANDELA	Generic distributed with central control	Uniform and fixed indexation at the same server as the content storage	Distributed MPEG-7 based metadata database
VITALAS	Service oriented architecture	Variable set of indexing algorithms, running on some indexation servers	A XML based metadata repository
KLIMT	SOA with a central control	Realized at the query moment, with a variable set of IA, on dedicated servers	XML metadata solution based on SOAP messaging
WebLab	SOA with central control	Realized at the acquisition moment, with a fixed set of IA, on dedicated servers	XML based centralized metadata database
[2]	Distributed architecture based on mobile agents	Accomplished by a fixed set of mobile IA on the content server	Features vector
CAM4HOME	Service oriented architecture with central control	Not in the purpose of the project	The CAM4HOME metadata model
MUSCLE	Centralized architecture	Variable set of algorithms managed at the central server	Not in the purpose of the project

Table 3. Comparative study of multimedia management projects

After a comparative study on the concrete LINDO system needs, we decided to adopt a distributed architecture with a centralized management because it enables:

- To have a centralized management of indexing algorithms and also to deploy them on demand on different remote servers;
- To extract simultaneously multiple and diverse metadata by executing different indexation algorithms, in parallel, on different remote servers;
- To dispatch user queries only on some specific remote servers that might contain some needed information;
- To simultaneously process queries on the central server and on remote servers.

The indexation process adopted in the LINDO project is managed at the central server level and accomplished at the remote servers levels:

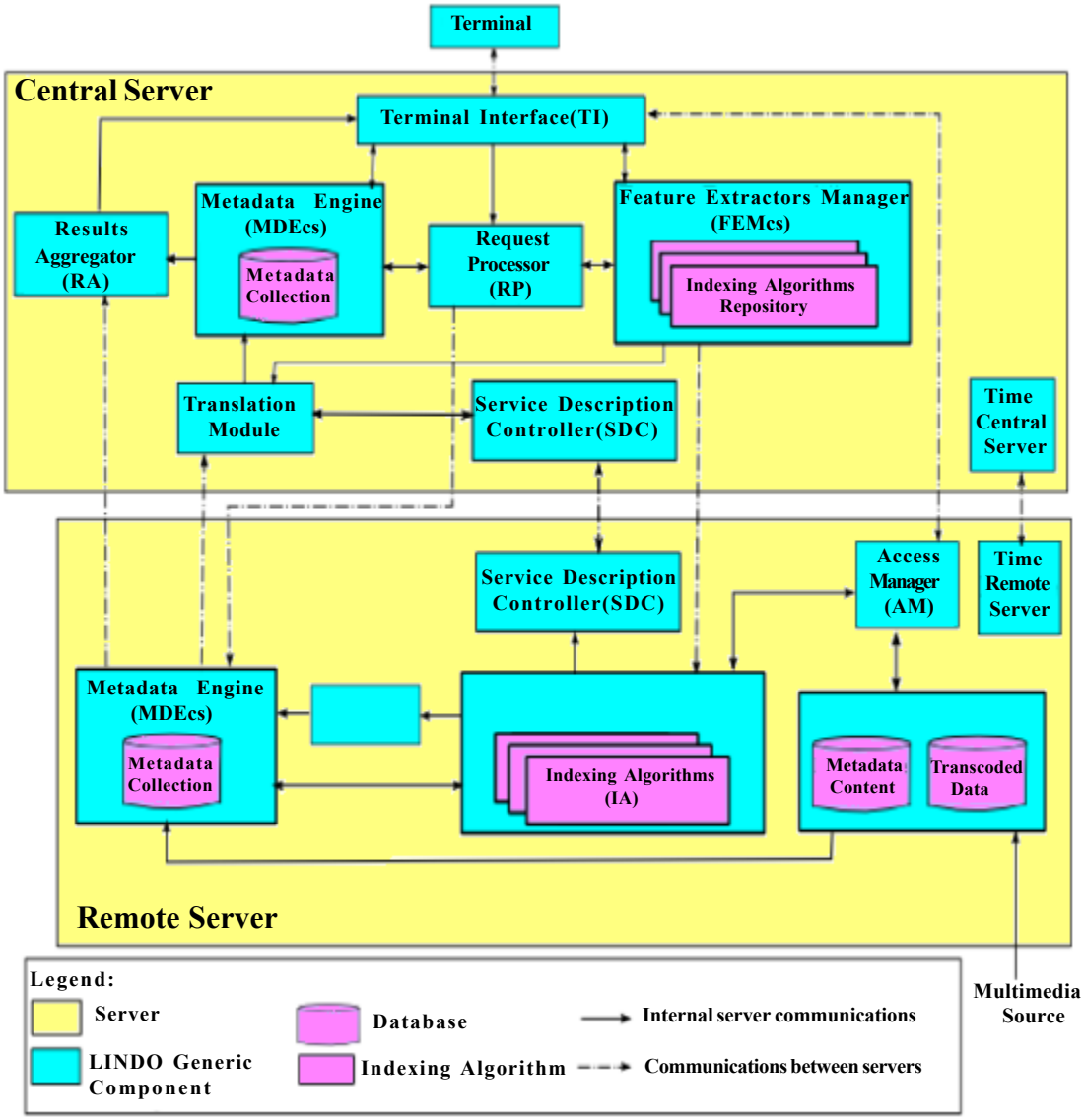

Figure 2. LINDO Architecture

- A generic interface was defined for indexing algorithms in order to uniformly handle them, and to enable the integration of new algorithms at any time into the LINDO architecture;

- Two indexation processes were defined: (1) an implicit indexation that is a priori executed over the multimedia contents from each remote server; (2) an explicit indexation that could be executed, on demand, on a specific remote server.

LINDO project aims to integrate any metadata models, such as standard metadata formats and uniform integrated models. In the following section, we will present in detail the architecture by emphasizing its generic property.

3. The LINDO generic architecture

The main goal of LINDO project was to specify a generic architecture that could guide the design of distributed multimedia information systems, while favoring a reduced resources consumption, in terms of data transfers over the network, storage and CPU consumption. Actually, the idea was not to define yet another information retrieval model or indexation engine, but rather to encapsulate any existing frameworks for indexing and retrieving multimedia contents, like those employed and/or defined by

the projects presented in Section 2.

Therefore, the choices made for the design of this architecture improve resources consumption:

For limiting data transfers, we chose to implement a distributed indexing technique. Instead of transferring content over the network to specialized indexing servers, the indexing algorithms are deployed on the remote servers where the content is stored. Also, the user queries are not sent to all remote servers but only to a filtered subset (according to the servers characteristics).

For limiting the databases dimension, we chose not to store several versions of each multimedia content, but rather to apply a conversion mechanism at the access moment. Also through the explicit indexing we don't execute all indexing algorithms over all content. Thus, we extract only the necessary metadata. A limitation of this approach is the increased query processing time. This problem can be addressed by classical query optimization methods (e.g., parallelization etc) that are not in the scope of this paper.

The limitation of the CPU consumption is accomplished through the implicit and explicit indexing. Thus, we chose not to execute all the available algorithms at the acquisition time but rather to execute only subset of generic algorithms (e.g., key frames extraction). At the query time, if necessary, other indexing algorithms could be executed.

Taking into consideration all these constraints, we have developed the LINDO generic architecture, illustrated in Figure 2, which enables each server to adopt uniform as well as differentiated indexations of multimedia contents. For that purpose, our architectural solution is divided into two main components: (1) *remote servers* which acquire, index and store multimedia contents, and (2) *a central server* which has a global view of the overall system and orchestrates the indexing and query processes.

This architectural solution makes LINDO-based systems scalable because it has the two following advantages:

- Each remote server is independent, i.e., it has its own specificities inside the distributed system, depending on different contexts, such as its location, its capacities or its application domain. For instance, some remote servers may index in real time acquired multimedia contents, while others may proceed to an off-line indexation.

- The central server can send relevant indexation routines or queries to relevant remote servers, while the system is running.

The whole LINDO generic architecture is illustrated in Figure 2. The upper part of the figure concerns the central server, while the lower part details a remote server scheme.

In the following, we will present and motivate each remote server module detailed in the lower part of Figure 2 (Section 3.1), as well as the central server ones (Section 3.2), detailed in the upper part of figure. The interactions between the modules will be presented in Section 5.1.

3.1 The remote servers components

A remote server stores and indexes all acquired multimedia contents, and could provide as well answers to some queries. For that purpose several modules have been defined and composed together:

- *Storage Manager* (*SM*) stores the acquired multimedia contents, in real time or off-line. Through the *Transcode* module, a multimedia content could be converted into several formats, with different qualities and encodings, which enables an end-user to download different encodings of one desired content.

- *Access Manager* (*AM*) provides methods for accessing the multimedia contents stored into the SM. This module can also select different parts of one multimedia content. For instance, given two timestamps, it can select the corresponding clip from a video. This feature is useful when an end-user wants to see a specific event.

- *Feature Extractors Manager* (*FEMrs*) is in charge of managing and executing a set of indexing algorithms over the acquired multimedia contents. At any time, new algorithms can be uploaded into this module, updated or removed, if needed. It can permanently run some algorithms over all the acquired contents or it can execute them on demand only on certain multimedia contents. Another important functionality of this module is the automatic selection of the indexing algorithms that obtain the best performances in the current execution context (see Section 6.1). The management of the indexing algorithms is based on a

technical and semantic description of these algorithms (see Section 4.2).

- *Filtering module* compresses the outputs of an indexing algorithm that may contain redundant or useless metadata.

- *Metadata Engine* (*MDErs*) collects and aggregates all extracted metadata about multimedia contents. The metadata stored into this module can be queried in order to retrieve some desired information. In the context of the project a metadata model was defined, with the goal of integrating some popular metadata standards and models (see Section 4.3 for more details). The utilization of this model is not mandatory.

- *Time Client* is in charge with the time synchronization between the remote servers and the central server.

- *Service Description Controller* (*SDCrs*) stores the description of the remote server, such as its location, its capacities, the software installed and the media acquisition context. In addition to this information, this module contains also the complete information about what indexing algorithms were executed and on which multimedia contents (see Section 4.1).

3.2 The central server components

The central server has a global view of the whole distributed system. It can control the remote indexation processes, as well as answering or forwarding user queries to the remote servers that may contain relevant results. One major difference between the central server and a remote server is that the central server does not store or index multimedia contents. Actually, the central server can deploy, on the remote servers, indexing algorithms on demand. It is also in charge with the filtering of the multimedia content that has to be indexed, based on the user query. It manages also the users that employ the system. Each user has associated a role, a localization, a material context and a set of rights (i.e., the actions he can accomplish, the content he can access). Thus, a central server is composed of the following components:

- *Terminal Interface* (*TI*) provides the interface between the user and the system. In the TI a user can specify some queries, in natural language or as keywords, and where the query results will be displayed. Inside the TI, several functions have been developed in order to visualize metadata collections, install, deploy and remotely execute some indexing algorithms. A login module integrated into the TI has the goal to identify each user and to display a different visual interface, according to the role of each user.

- *Metadata Engine* (*MDEcs*) contains information related to the remote servers. It can contain some extracted metadata about distributed multimedia contents, some contextual information about the whole system, specific remote server acquisition contexts, the remote servers' descriptions, such as their locations, their capacities, etc., but also some supplementary background knowledge[17].

- *Service Description Controller* (*SDCcs*) collects all remote server descriptions, and aggregates them to the metadata collection in the MDEcs.

- *Feature Extractors Manager* (*FEMcs*) manages the entire set of indexing algorithms used in the system. It can deploy, update and remove any indexing algorithm, on any remote server, at any time. Furthermore, this module can execute and stop at any time an indexing algorithm.

- *Request Processor* (*RP*) processes the query in order to extract the demanded multimedia features, and executes these queries on the central server metadata engine (MDEcs) or forwards the queries on specific remote server metadata engines (MDErs). Based on the user query and on the information contained by the MDEcs, this module decides if new indexation has to be accomplished on some remote servers, and, if it is the case, it selects the best indexing algorithms that should be employed. Before this selection the RP checks of the user's role allows him to run explicit algorithms.

- *Results Aggregator* (*RA*) aggregates the results received from the metadata engines, the MDErss and MDEcs. Actually, from a user query, it groups all the current available answers and, if necessary it filters the results according to the user's rights, and then sends them to the Terminal Interface module for displaying.

- *Translation module* homogenizes the data stored into the central server metadata engine. Indeed, many different models can be used by remote servers for storing the metadata, obtained after the multimedia contents indexation, or describing their characteristics. Hence, this module unifies all descriptions in order to provide one global view of the system.

- *Time Server* provides a unique system time that is used for synchronizing all remote server times.

[17] [26] propose also to store on a central server a descriptive hierarchy of the metadata collected by the system

The number of remote servers that can be used by a LINDO compatible system is not limited. A LINDO system supports any domain of application and it enables to integrate different *metadata models*. Thus, the proposed architecture is scalable and generic, and it offers an efficient *distributed and dynamic indexing algorithms management*. The adopted indexation management is based on descriptions for remote servers, indexing algorithms and multimedia contents.

Before the presentation of the indexation and query workflows (Section 5), we describe these adopted descriptions in the next section.

4. LINDO adopted descriptions

Apart from the architecture, the LINDO project led to the proposal of descriptions for the indexing algorithms, the remote servers and also of the multimedia metadata. When establishing these descriptions we took into consideration a wide variety of characteristics that describe the different resources. These descriptions are key elements in the LINDO system. They assure the integration of any indexing algorithm in the system and the description of any multimedia content. The data and system workflows defined on the LINDO architecture are based on these descriptions. In the following we present de description of the remote servers (Section 4.1), the description of the indexing algorithms (Section 4.2) and the multimedia metadata model (Section 4.3).

4.1 Remote server

Each remote server has associated a description that captures its physical characteristics, a description of its context (e.g., domain, location) and also the installed indexing algorithms. An important part of this description is dedicated to the modeling of the acquisition context. Thus, the remote server description contains the following classes:

- **RemoteServer:** general information about the server (e.g., its name, its description, its domain)

- **TechnicalProperties:** elements that describe technically the server (e.g., the operating system, the CPU); it is important to have these elements in order to match them to the constraints of the indexing algorithms (e.g., there are algorithms that only run on Linux);

- **IndexingAlgorithm: a** list of all the algorithms installed on the remote server and some description elements for each of them (e.g., their type: implicit or explicit (see Section 5 for details));

- **Device:** all devices associated with the server (e.g., Camera, Microphone, Sensor); all information captured by these devices are stored and processed by the remote server. This element is not mandatory for the description of the remote server, because it is possible to have a remote server that is used for managing multimedia content generated through editing software ;

- **Localization:** contains the information on the localization at a certain moment. This localization contains a textual description and the GPS coordinates. A server can have different locations over time, if it is located on a bus, in the context of a video surveillance application, but it can also be situated in a fix location. The localization is also associated to a device, because we can consider that a device is locate in a certain place, orientated towards a certain direction.

- **AcquisitionContext:** describes the acquisition context, at a certain moment, that can be associated to a remote server. The major characteristics that are taken into account are the weather conditions, the luminosity conditions and the temperature.

An example of a description is provided in the next table. This description is associated with a remote server located in the Austerlitz train station in Paris, and which manages the content form video surveillance cameras located in the parking in front of the station.

4.2 Indexing algorithm

In order to select the relevant indexing algorithms for a need and a context, it is necessary to describe the indexing algorithms characteristics. To cope with the need that we have specified a generic indexing algorithm model (Figure 4). In order to give an illustration, we present an example of algorithm that detects pedestrians in images. We divide its description considering the model's components. A more explicit version of the model can be found at the following URL: http://www.irit.fr/~Dana.Codreanu/images/model2.jpg.

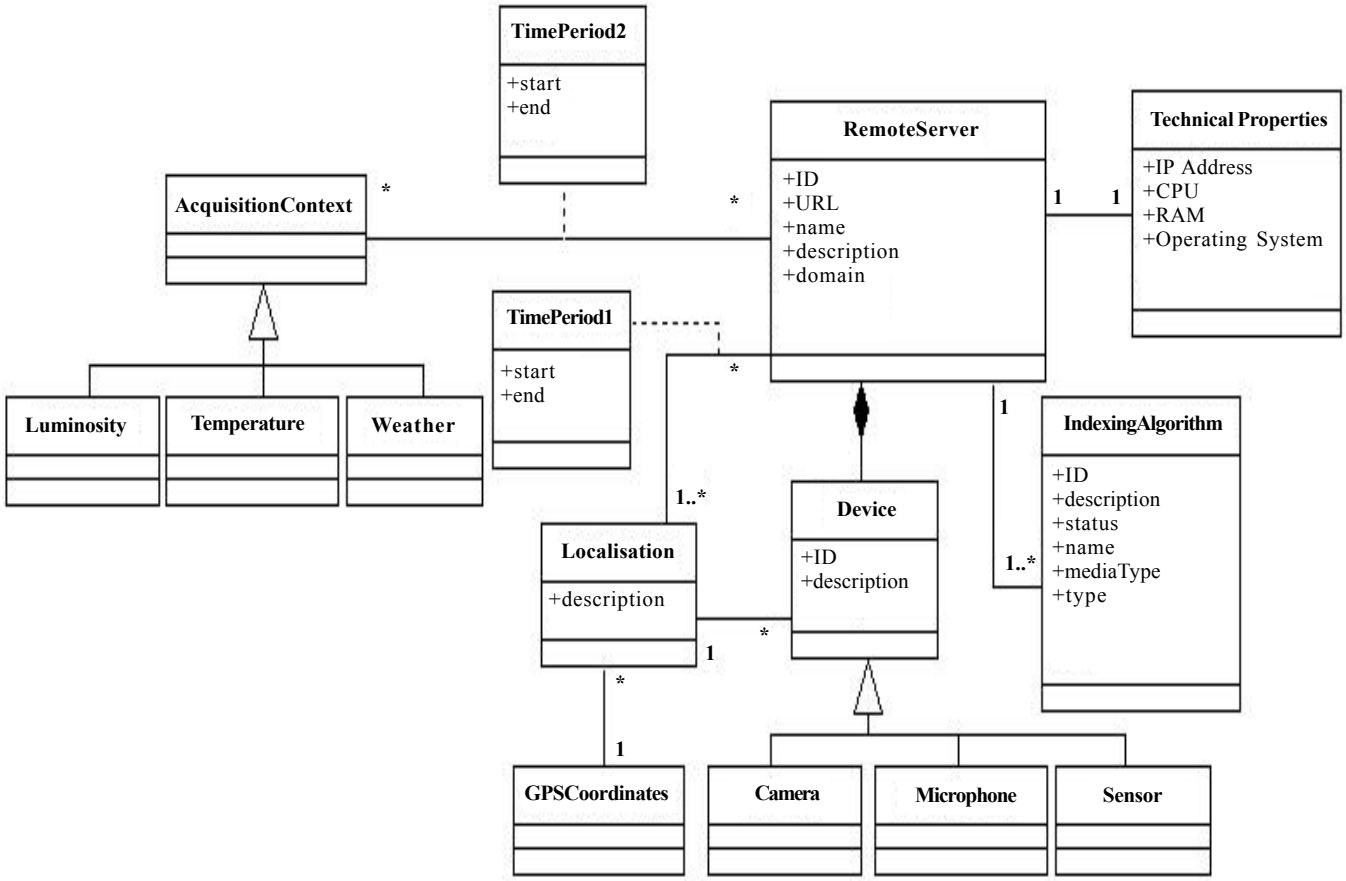

Figure 3. Remote Server Description Model

Let us comment the proposed model:

• **AlgorithmModel**: identifies some general characteristics of an indexing algorithm, such as its name, author, location (e.g., *AlgoName* = Pedestrian Recognition, Author = IRIT, Script = C++, URL= www.irit.fr/pedestrianRecognition.zip, *MediaType* = Image, *Complexity* = O (n)). Such a characterization of indexing algorithms is useful in order to to identify them, to localize them (in a multimedia system we could refer to indexing algorithms that are stored on different sites) and to compare them (e.g., compare their efficiency). The *MediaType* represents the type of media the algorithm could treat (e.g., image, video).

• **InputParam:** contains the indexing algorithm input parameters descriptions, like the category of media that has to be treated (e.g., news videos, audio speech etc.), the different algorithm settings. . . (e.g., *Used Features* = {Color Histogram, Spatial Location, etc.}, *ConfidenceThreshold* = 0). These are the parameters that allow a configuration of the algorithm.

• **OutputObj**: determines the different algorithm outputs, e.g., the produced metadata formats and the description of the extracted features (e.g., *OutputObjectType* = metadata, *Format* = Generic Format, *OutputObjectDescription* = detects the pedestrians and their position). The description of the output object is used to select the algorithms which extract the desired elements.

• **ExecutionConstraints:** specifies a set of constraints (e.g., Operating System, CPU, RAM) that have to be satisfied in order to correctly execute th*e indexing algor*ithm. For i*nstanc*e, suppose the fol*lowing constraints for* the algorithm that we gave as example: *ProcessingConstraints* = the algorithm detects only standing pedestrians, *MultimediaConstraints* = accepted data format is jpg, *PlatformConstraint* = {OS = Linux, CPU >=2,5GHz}. These elements are used in order to verify if an algorithm can be executed in a given context. They can be useful in a debugging process (if the obtained results are not satisfying and the preconditions a*re satisfied then by* looking at these constraints we could find the cau*se of the fault).*

```
<RemoteServer id = "rs2" name="RServer-2">
    <localisation>parking St. Lazare train station, Paris, France </localisation>
    <description>Manages content from cameras located in the parking in front of the
      station </description>
    <devices>
             <camera id="c1Paris">
                  <description>located in the north corner</description>
             </camera>
    </devices>
    <acquisitionContext>
         <weather>
            <period start="2011-07-14T08:07:00" end="2011-07-14T11:33:00">cloudy</
                period>
            <period start="2011-07-14T11:34:00" end="2011-07-14T19:14:00">sunny</
                period>
         </weather>
         <luminosity>
               <period start="2011-07-14T08:07:00" end="2011-07-14T11:33:00">75</
                  period>
               <period start="2011-07-14T11:34:00" end="2011-07-14T19:14:00">100</
                  period>
         </luminosity>
         </acquisitionContext>
         <indexingAlgorithms>
               <indexingAlgorithm id="ia2rs1" mediaType="video" name="person
                  detection">
               <description>Detects persons in outdoor area and their predominant
                  color</description>
         </indexingAlgorithm>
    </indexingAlgorithms>
</RemoteServer>
```

Table 4. Example of Remote Server Description

• ***Per*formance:** contains a set of mea*sures related to a* set of data tests (e.g., for a set of Flickr[18] photos we have a precision of 85% and a recall of 80% and for a set of captures from a room we have a precision of 50% and a recall of 40%). This is useful for instance to compare some algorithms, or if we know the category of desired media (given as input parameter) we could choose the algorithm with the best performance for that media category (e.g., a journal photo or a capture from a video).

In some situations, it could be necessary to execute other algorithms before the one that we gave as example (e.g., an algorithm that separates the visual and audio contents of a video, an algorithm that extracts the key images of a video content). Through the relation isChained, one may identify that several indexing algorithms can be executed sequentially.

We have proposed an XML serialization of the model illustrated in Figure 4. Some instantiations (descriptions of real algorithms developed within the LINDO project or within some research teams) are available online[19]. Furthermore, we have developed a Java applet[20] which allows creating and storing some indexing algorithms descriptions.

The following XML fragment represents an example of such algorithm description.

[18] http://www.flickr.com

[19] http://www.irit.fr/~Dana.Codreanu/LINDO/listingAlgos.php

[20] http://www.irit.fr/~Dana.Codreanu/LINDO/launch.html

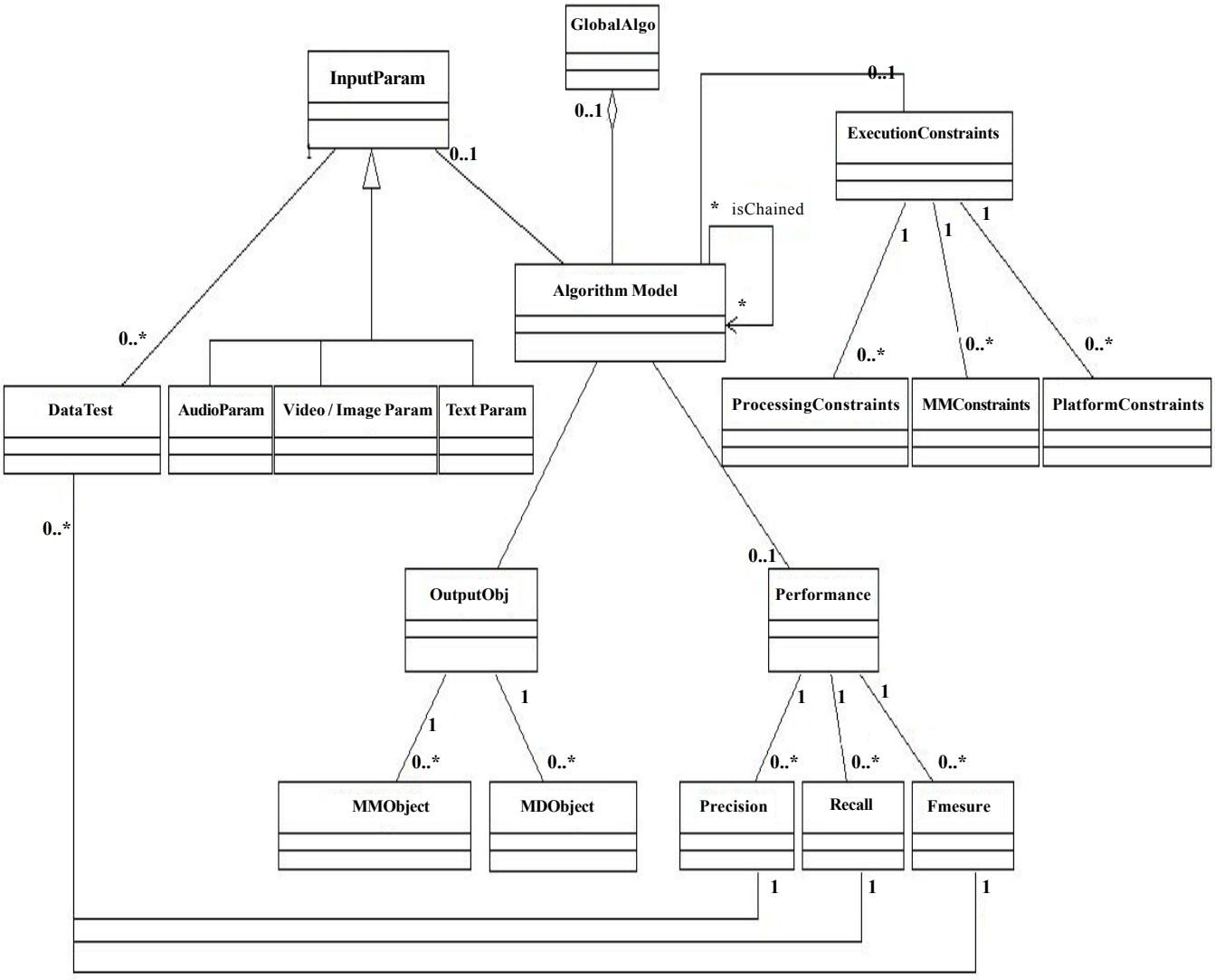

Figure 4. Algorithm description model

4.3 Multimedia content

After a study of the existing multimedia metadata standards we chose the ones that better corresponded to the project needs. Based on these standards and on their requirements stated by the project's partners we defined a multimedia metadata model that can be associated to any multimedia content, from text documents to video surveillance and broadcast content. An overview of this model in presented in Figure 5.

Our model is structured on several levels. We associate a MultimediaMetadata to each multimedia content, which includes all the metadata elements related to that content. The study of the metadata standards and of their classification concluded with the identification of a group of metadata elements that describe the document as a whole, and that are mainly related to the administrative side of the metadata. For this reason we created a GeneralInformation class that is associated with the MultimediaMetadata. This element contains for example the author, the creation date, the size, the subject, the key-words, the description.

The multimedia content can be composed by several types of media, combined through a certain structure. In order to capture this characteristic of the multimedia document, we added a supplementary level, by saying that the MultimediaMetadata is composed by multiple media. From a logical point of view, we can state that the general metadata elements can be attached also to a media.

```
<AlgorithmModel AlgoName= "Person Detection" MediaType="Video">
             <InputParameters>
                              <InputParamFileFormat>xml </InputParamFileFormat>
             </InputParameters>
            <OutputObject Type="Metadata">
                             <MetadataObject>
                                      <MetadataObjectDescription> person and
color detection algorithm</MetadataObjectDescription>
                             </MetadataObject>
            </OutputObject>
            <ExecutionConstraints>
                    <MMConstraints>
                              <weather>windy, cloudy</weather>
                              <luminosity min="50" max="80" />
                              <DataFormat>MPEG, AVI</DataFormat>
                  </MMConstraints>
                  <PlatformConstraints>
                  <OS>Windows</OS>
             </PlatformConstraints>
       </ExecutionConstraints>
</AlgorithmModel>
```

Table 5. Example of Algorithm Description

```
<document src = "stream1">
    <generalInformation>
            <filename>stream1</filename>
            <title>St. Lazare train station, Paris</title>
            <subject>video surveillance</subject>
            <creationDate>2010-07-28T00:00:00</creationDate>
    </generalInformation>
    <video capturedBy = "cam1_Paris">
           <object type = "Person" id ="0">
                 <localisation confidence = "100">
                      <period start_time = "2010-07-28T11:07:35" end_time = "2010-
                       07-28T11:08:55"/>
                      <area>control room</area>
                 </localisation>
                 <property name = "color">red</property>
           </object>
    </video>
</document>
```

Table 6. Multimedia metadata example

In order to determine what metadata structure to consider for each media type, we studied the existing algorithms that extract information from each media type. We had this approach because we wanted to include with predilection in our model the elements that are really extracted. Thus, for the text documents, we took into consideration the segmentation of the document in chapters, sections and paragraphs. For the audio content we took into consideration its possible segmentation is speakers, topics, episodes and sections, as it can be realized by through the Transcriber[21] software.

In the last years there are many research groups that are interested in the identification of a certain concept in the multimedia

[21] http://trans.sourceforge.net/en/presentation.php

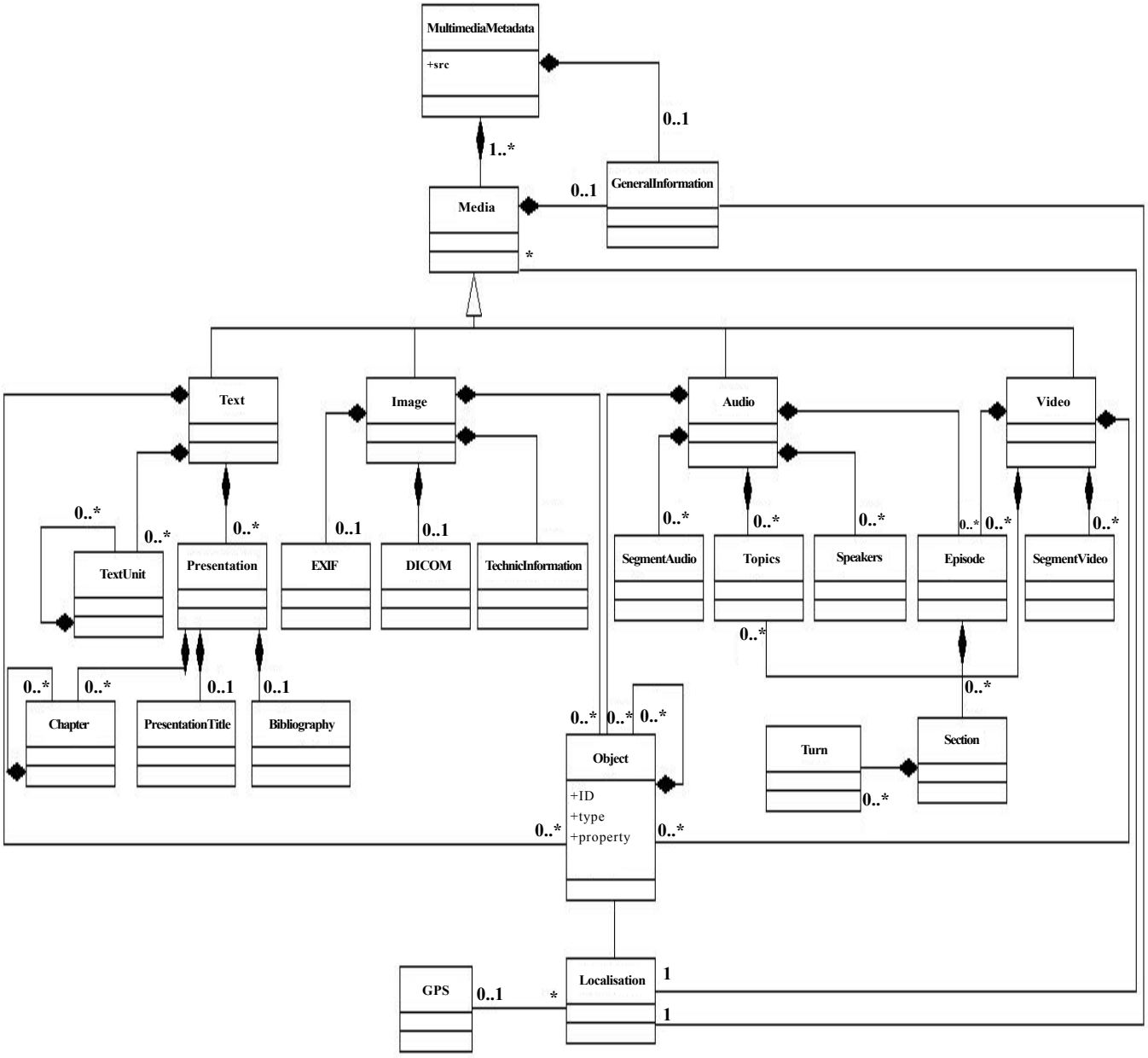

Figure 5. Multimedia metadata model

content (e.g., the LSCOM concepts[22]). We provided in our model the possibility of including these concepts in the description of the multimedia content through the Object element. The Object, the Media and the GeneralInformation have associated a Localisation in order to establish the place where the document was created, or the place where the Object is located in the media. The video surveillance context of our project motivated us to add the GPS coordinates to the Localisation element.

In the next table we present an example of metadata which was automatically obtained in the experimental phase of the project. This XML metadata is associated to a video took by a video surveillance camera in Montparnasse Station in Paris. This video was created on the 28 of june 2010 by the camera with the ID = "*cam1_Paris*". In this video a person was detected in the control room from 11;07:35 until 11:08:55. The algorithm that extracted this person from the video content extracted also the predominant color of his/her cloths. In our case this person is dressed in red (the object has a property color with the value red).

[22] http://www.lscom.org/ontology/index.html

All these descriptions are used in the realization of the indexing and query processes. In the next section we present how these processes are adressed by the LINDO framework. In the Section 5.1 we provide a general presentation of the indexing and query workflows. Then, in Section 5.2 we describe in details the way the indexing algorithms are selected for accomplishing the indexing.

5. Implicit and Explicit Indexing Processes

5.1 General Presentation of System Workflows

The LINDO architecture, presented in Section 3, enables the specification of several workflows that can be used for the implementation of different use cases. Moreover, in order to facilitate server resource consumption, multimedia content indexing is realized at ingest time - i.e., *implicit indexation* - and on demand - i.e., *explicit indexation*. Indeed, it avoids executing all possible indexing algorithms at once. Figure 6 illustrates the indexing and querying workflows proposed by the LINDO framework.

When a new multimedia content is ingested by a remote server, its SM first stores it. Afterwards, the FEMrs module starts the *implicit indexation process* (steps 2 to 19). This process consists in the execution of a predefined set of indexing algorithms, i.e., implicit indexing algorithms (iIAs), on the acquired multimedia content in order to extract some metadata that will be further queried. For that purpose, the FEMrs retrieves the multimedia content from the SM module through the AM module. Mainly in function of the media types and of the acquisition context, the FEMrs selects and executes the implicit indexing algorithms (steps 8 and 9). Once the execution of an indexing algorithm is achieved, the obtained metadata is forwarded to the Filtering module. The filtered metadata is then stored by the MDErs in its metadata collection. Each time the metadata collection is changing, the MDErs sends a concise version of the added metadata to the Translation Module on the central server. This concise version of the extracted metadata, that we call summary, can be obtained by using the algorithm proposed in [14]. This summary can be created using different strategies, according to each application domaine (e.g., for video surveillance it can consist in statistics based on the metadata obtained in the last hour of recording, for broadcast, it can consist in the titles and participants for each broadcast content (article, show, etc.)). Once the translation is done, the metadata are finally sent to the MDEcs in order to be stored and used for the querying process.

The query workflow begins with the user query specification by employing the TI. The user formulates the query through a graphic interface that enables him to specify five query components: the query itself (as free text), the location (free text), time span (calendar-based), domain (checkbox list) and the media type (video, image, audio or text). The query free text is then analyzed in order to obtain a set of features. For more details about query processing please refer to [16]. Each user query (the set of features, the localization, the time span, the domain and the media type) is analyzed in order to translate it into the formal language used by the MDEcs and the MDErs. After this transformation, the RP selects, based on the information stored in the MDEcs, the remote servers that might have results for the query. The RP sends the query to the MDErs of the selected remote servers (steps 22 to 24), where the query is executed and the results are transferred to the RA module in order to be ranked and finally sent to the TI for displaying (steps 31 and 32). The RA takes into consideration in one hansd the user's context and role (accessing rights) and on the other hand the permitted use and distribution rights of the content. After matching these two the RA decides if the content is displayed as it is or it has to be modified (e.g., resize, apply a blur region).

At this step of the query process it is possible that not all the remote servers were selected for executing the query. This does not necessarily mean that they do not have results for the user query. It is possible that their multimedia contents were not indexed with the right indexing algorithms. For this reason, the RP module selects: (1) the remote servers that should be reindexed, (2) the supplementary indexing algorithms, i.e., explicit indexing algorithms (eIAs), that could be deployed on these remote servers in order to produce additional metadata that could provide answers to the user query (step 25) and (3) the multimedia contents on which the indexing algorithms should be executed. The user is informed that supplementary indexing is performed in order to retrieve other results to his query and that he will be announced when the results are available. Before this selection the RP Checks of the user's role allows him to run explicit algorithms as these algorithms analyze more in detail the multimedia contents and usually they employ a lot of resources.

At this step, the RP decides, according to the user query, on what multimedia content these explicit indexing algorithms have to be executed. The FEMcs is in charge with the deployment of these explicit algorithms on the concerned remote servers, if necessary (step 28). Form this point the FEMrs takes the control of the indexing, which is accomplished as for the implicit indexation (steps 29, 30, and 33 to 40). When the explicit indexation is finished, the MDEcs informs the RP that other results are

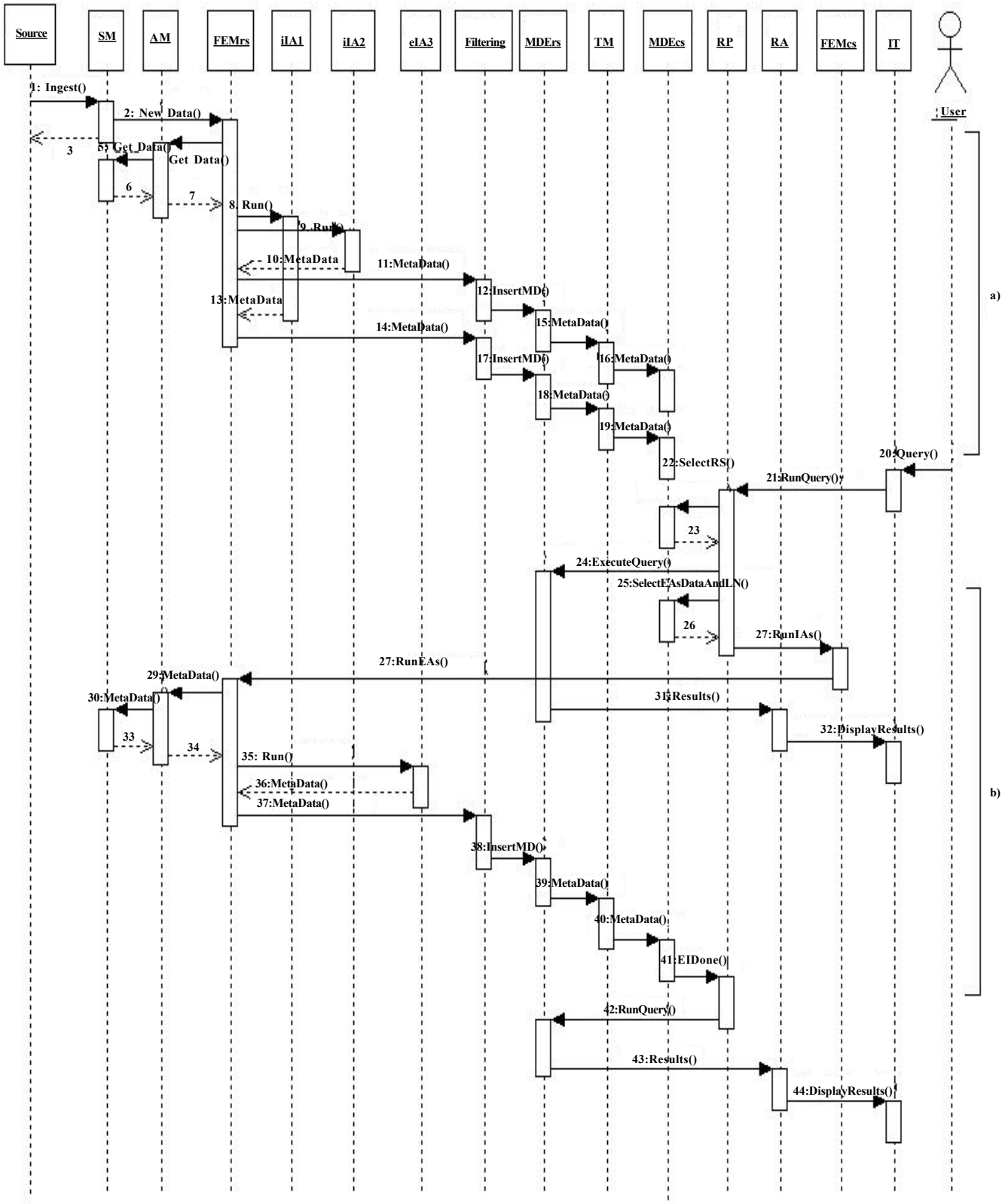

Figure 6. Sequence diagram of a) implicit indexation and b) explicit indexation

available (step 41). At this moment, the RP module sends the user's query to the MDErs of remote servers where the explicit

indexation was accomplished. The results obtained are sent to the RA module which informs the user that supplementary results are available to his query (steps 43, 44).

In the next section we will detail the defined mechanisms for the selection of the indexing algorithms according to the user query.

5.2 The Selection Mechanism of Indexing Algorithms

Our goal is to select an appropriate set of algorithms for indexing multimedia contents. The input elements of our algorithm are:

- a query (as a set of features), i.e., Q= f1, f2, . . . , fm;
- a set of indexing algorithms descriptions which correspond to the model that we presented in Section 4.2;
- an execution context, for example the description of each server of the multimedia information system, containing their operating system, the type and frequency of their processors, their characteristics (e.g., if a server is able to process texts, images or videos), their network performances, etc. In the following, we consider two elements for describing the context: the operating system and the processor frequency (we use the notation Context= {OS, CPU});
- a set of constraints (e.g. Multimedia type, Localization).

Our algorithm selection mechanism follows three steps. In the following sections we present these steps and we illustrate them through an example. Let us suppose a query R= {person, car, color, parking} and the following constraints: (C1) we search the algorithms that can process videos, (C2) we need to run the algorithms on a Windows platform with a 2,5 GHz processor. For this exemple we have a collection of 18 indexing algorithms (Ai) that extract several features. In the following we present the list of algorithms descriptions with their context elements (the media type that can be processed by each algorithm, the operating system, and the minimum processor frequency needed)

A1 : Acoustic Segmentation : Audio, OS = “Windows”, Processor = “2.2GHz” ;

A2 : Car detection in a parking : Video, OS = “Windows”, Processor = “2GHz” ;

A3 : Cars color recognition : Video, OS = “Windows”, Processor = “2GHz” ;

A4 : Color recognition : Video, OS = “Windows”, Processor = “2GHz” ;

A5 : Demultiplexing : AudioVisual, OS = “Windows”, Processor = “2GHz” ;

A6 : Detecting Persons : Video, OS = “Linux”,Processor = “2.7GHz” ;

A7 : Face Recognition : Image, OS = “Linux”, Processor = “2GHz” ;

A8 : Feature Extractor : Video, OS = “Windows”, Processor = “2.5GHz” ;

A9 : Key Images Extractor : Video, OS = “Windows”, Processor = “2GHz” ;

A10 : Language Detection : Text, OS = “Windows”, Processor = “2GHz” ;

A11 : Named Entity Recognition : Text, OS = “Windows”, Processor = “2GHz” ;

A12 : Pedestrian Recognition : Image, OS = “Windows”, Processor = “2GHz” ;

A13 : Person detection : Video, OS = “Windows”, Processor = “SGHz” ;

A14 : Person detection in moving cars : Video, OS =“Windows”, Processor = “2GHz” ;

A15 : Scenes Resuming Images Extraction : Video, OS=”Linux”, Processor = “2GHz” ;

A16 : Shot change detection : Video, OS = “Windows”, Processor = “2.7GHz” ;

A17 : Speech Diarization : Audio, OS = “Linux”, Processor = “2.5GHz” ;

A18 : Topic Segmentation : Text, OS = “Windows”, Processor = “2GHz” ;

5.2.1 Step 1 : Computation of the algorithm lists which extract the elements of the query

a) For each feature of the query, a list of indexing algorithms (AlgoList) which extract the feature is built (each feature is searched in the field Output Object Description of the description model);

b) The algorithms that do not comply with the context conditions, like for example the OS or the CPU (these values are compared with the values of the fields from the class Platform Constraints of the model) are eliminated.

A list that contains all the distinct algorithms that extract the features of the query is created (AlgoSolution).

Algorithm 1: The steps 1, 2 and 3a of the algorithm selection.
Input: A list of features (f_i) from the query R, a list with the context elements values C
Output: L is the algorithm list that extracts the query elements and that satisfies the context

1. ***for*** each term f_i of R ***do***
2. ***for*** each algorithm A_j of the data base ***do***
3. ***if*** OutputObjetDescription(A_j) contains f_i ***then***
4. ***if*** OS(C) = PlatformConstraintsOS(A_j) ***and*** CPU(C) >= PlatformConstraintsCPU(A_j) ***then***
5. $AlgoList_i \leftarrow AlgoList_i$ or $\{A_j\}$
6. AlgoSolution $\leftarrow$ AlgoSolution or $\{A_j\}$
7. ***end***
8. ***end***
9. ***end***
10. ***end***
11. ***for*** each algorithm A_i of AlgoSolution ***do***
12. $InvertedList_i$ = the query features which are extracted by A_i;
13. $Score_i$ = number of query features extracted by A_i;
14. ***end***
15. Construction of the list of lists AlgoInfo={{A_i; $Score_i$; $InvertedList_i$}};
16. Sort(AlgoInfo);
17. Search(AlgoInfo,0);

In order to exemplify this first step (lines 1 to 10 in the Algorithm 1), we execute the query R, mentioned before on the algorithms collection. After the execution of the first step the following lists are created:

$AlgoList1 = \{A_6, A_{13}, A_{14}\}$;

$AlgoList2 = \{A_2, A_3, A_4, A_{14}\}$;

$AlgoList3 = \{A_3, A_4, A_8\}$;

$AlgoList4 = \{A_2, A_3, A_4, A_{14}\}$;

$AlgoSolution = \{A_2, A_3, A_4, A_6, A_8, A_{13}, A_{14}\}$.

The A6 algorithm will be eliminated because it can be executed only on Linux OS.

5.2.2 Step 2: Computation of the inverted lists

A score is associated for each algorithm in AlgoSolution. This score may depend on the number of extracted features (size of the inverted list). We could compute other scores for the algorithms, for example by associating different priorities to query features. This step corresponds to lines 11 to 15 of Algorithm 1.

In our example we consider the score of an algorithm as the size of the inverted list. Thus, we obtain the following inverted lists:

$InvertedListA_2$ = {car, parking}; $ScoreA_2 = 2$;

$InvertedListA_3$ = {car, color, parking}; $ScoreA_3 = 3$;

$InvertedListA_4$ = {color, parking, car}; $ScoreA_4 = 3$;

$InvertedListA_8$ = {color}; $ScoreA_8 = 1$;

$InvertedListA_{13}$ = {person}; $ScoreA_{13} = 1$;

$InvertedListA_{14}$ = {person, car, parking}; $ScoreA_{14} = 3$.

5.2.3 Step 3: Computation of the result lists

a) The algorithm list is sorted by the scores. For example, if we want first the algorithms which extract many features, the list will be sorted descendant. If the contrary situation, the list will be sorted ascendant;

For the example that the algorithms are descendant order: AlgoInfo= {{A3, InvertedListA_3, 3}, {A4, InvertedListA_4, 3}, {A14, InvertedListA_{14}, 3}, {A2, InvertedListA_2; 2}, {A8, InvertedListA_8, 1}, {A13, InvertedListA_{13}, 1}}.

b) A research procedure identifies the algorithms associations that allow identifying all or a part of the query elements. In the following, we detail the method Search (line 17) which corresponds to the step 3.b. The Search method implements a backtrack algorithm (see Algorithm 2) which allows to find a combination of indexing algorithms that satisfy the user needs and the execution constraints. This procedure stops when the algorithms in S2 extract all query features. In this case, S2 is added to the result L (lines 1 to 3). At each step of the algorithm, the mechanism will consider those that have a score maximum (lines 10 to 15). Each best candidate is added to the S2 list and it is deleted from S1 (so it won't be considered in the next steps of the recursion). In the same time, the mechanism recalculates the S1 algorithms scores (lines 20 to 22). The method Search is called once again with the new updated input parameters and this until an association of algorithms that respond to the query is found (line 27). The procedure could be optimized by using back marking or forward checking techniques.

Algorithm 2: Search
Input: List of selected algorithms with the Inverted List and score attached (sorted by score) and the number of elements (features) extracted by the current selected algorithms
Output: Combinations of algorithms that extract all query elements

```
1.  if NumberOfExtractedElements = cardinality(R) then
2.          L.add(S2);
3.  end
4.  else if S1= Empty then
5.       Lincomplete.add(S2);
6.  end
7.  else
8.       copyS1= clone(S1);   /* Duplication of S1and S2 */
9.       copyS2= clone(S2);
10.       listBestAlgos = the algorithms A_i for which Score_i = Score_max;
11.      copyS1= clone(S1);
12.  for each algorithm A_i of listBestAlgos do
13.      copyS1.remove(A_i);
14.      copyS2.add(A_i);
15.      for each algorithm A_j of copyS1 do
16.          Score(A_j) = Cardinality(InvertedList(A_j )- ListExtractedElements);
17.          if Score(A_j) = 0 then
18.              copyS2.add(A_j );
19.          end
20.      end
21.     LExtractedEl ← LExtractedEl or InvertedList(Ai);
22.     Search (copyS1, copyS2, NumberOfExtractedElements + Score(Ai));
23.    copyS1= clone(S1);
24.    copyS2 = clone(S2);
25. end
```

After the execution of the Search method and after eliminating the duplicate values, we have the next solution: {{{A3, A13}, {A3, A14}}, {{A4, A13}, {A4, A14}}, {A14, A8}}.

The quality of the detections realized by the indexing algorithms is tributary to the quality of the multimedia content that is

indexed and to the acquisition context of this content. In order to by-pass these limitations, we propose an algorithm that take into account the acquisition context in order to dynamic update the implicit and explicit algorithms.

6. Dynamic Mechanism to update the Implicit and Explicit Indexation Algorithms

The improvement of the indexing algorithm selection can be accomplished in two ways: first by taking into account the environmental conditions in which the multimedia content was created; and second by modifying the implicit algorithm set according to the utilization frequency of the indexing algorithms in the explicit indexation phase. These tow mechanisms are further detailed.

6.1 Considering Environmental Conditions (for Establishing Implicit/Explicit Algorithms)

As could be noticed, the semantic description of an algorithm provides information about the constraints that enable to obtain the optimal results further to the algorithm execution. Such constraints include the quality of the multimedia content that is indexed, the weather conditions (sunny, cloudy, windy, rainy, snowy), the luminosity conditions (the degree of luminosity: night, day, murky, luminous, clear), location (indoor, outdoor), the language of the multimedia content (for speech detection), etc.

For the *implicit indexation*, a set of indexing algorithms is employed in order to obtain a specified set of multimedia features. For example, in a parking place, the implicit algorithms could concern person detection, car detection and registration plate detection. If for each of such detections, only one algorithm is employed, it is possible that the quality of detection would be affected by the changes encountered in the above mentioned environmental conditions.

For this reason, we propose for each multimedia feature that is intended to be detected implicitly on a remote server, multiple indexing algorithms to be included in the FEMrs, each one of them having different execution constraints. Based on this algorithms collection managed by FEMrs, we propose to dynamically change the implicit algorithms: when some environmental changes are encountered, the FEMrs will replace the current implicit indexing algorithm that detects a certain feature (e.g. human presence) with another algorithm that detects the same feature, but has a better performance in the current conditions.

For the parking example, we consider the weather and luminosity conditions. These changes are easy to capture based on some sensors, and are stored in the description of the remote server, managed by the SDCrs module. Multiple "*person detection*" algorithms are available on this remote server, each of them having best performance in a certain weather condition (sunny, cloudy, windy, rainy, snowy) and into a certain range of luminosity intensity (from 0% to 100%).

Let us suppose that at the current moment the weather is sunny and on the remote server a "*person detection*" algorithm is running (i.e., it is considered as an implicit indexing algorithm), with the best performance for "*sunny*" weather and "*minimum* 80% *degree of luminosity*". If a drastically weather change occurs, such as some dark clouds appear and it starts to rain, the sensors will transmit the modifications of the two considered parameters. Based on the semantic descriptions of algorithms, FEMrs module will select another "*person detection*" algorithm to be executed, which has a good performance on a rainy weather, and in the conditions of a 40% luminosity degree.

As well, the environmental conditions are considered during the *explicit indexation*. Further to query analysis, there will be known the multimedia features that should be supplementary detected, as well as the multimedia sub-collection that should be indexed (e.g., corresponding to a certain time period, and to a subset of remote servers). Based on the mechanism presented in the sub-section 5.2, all the possible algorithms combinations are established that could extract the set of the multimedia features corresponding to this explicit indexation. Further, for each remote server selected for explicit indexation, the summary of its environmental parameters is consulted on the central server by the FEMcs. It splits the time period required by this indexing in some time sub-intervals, corresponding to changes encountered in the environmental parameters. For each sub-interval, FEMcs selects the algorithms combination that best suits the values of these parameters. The selected algorithms combinations are sent to the remote servers, where the explicit indexation is accomplished in a differentiate manner for each time sub-interval.

6.2 Considering the User's Queries history

On each remote server, a fixed set of multimedia features is a priori established for being extracted during the implicit indexation (according to the characteristics of the remote server). Other multimedia features could be extracted normally during the explicit indexation, based on the user query.

We propose to consider the recurrent user queries during a time period in order to set up new implicit algorithms among the most demanded explicit algorithms. For this purpose, we associate a weight with each multimedia feature. It is possible that some features occur recurrently in the user queries during a certain time period. The weight of each feature increases with each query that includes it. Our proposal consists in temporary considering a certain feature in the process of implicit indexation (more precisely, the algorithm that detects it) from the moment when its weight reaches a certain threshold. Of course, multiple algorithms could exist for detecting a certain multimedia feature, and the selection of the most suited one to the current context is accomplished as described in the previous sub-section. At the beginning of each time period (e.g., hour, day, week, etc.), the weight of all supplementary multimedia features are set up to zero, this avoiding that some temporary demanded features to be extracted permanently.

If we consider the parking example, it is possible that during a week multiple user queries concerning "*snatched bag*" occur and determine the explicit execution on a certain remote server of the algorithm that detects "*snatched bag*". Consequently, the weight of the multimedia feature "*snatched bag*" increases. When this weight reaches the considered threshold, the algorithm that detects "*snatched bag*" becomes implicit. At the beginning of a new week, the weight of the feature "*snatched bag*" is set on zero, while the algorithm still remains implicit. If during the current week, only few queries concern "*snatched bag*", the weight will remain low and the algorithm will not be kept among the implicit algorithms for the next week.

In order to evaluate our proposal the LINDO architecture was instantiated. In the next section we present the use case implemented for the final demonstration of the project. In the section 7.1 we illustrate the architecture as it was physically and logically created. Then, in section 7.2 we focus on some utilization examples that illustrate the functioning of the system based on some user queries.

7. Illustrative use case

7.1 Architecture instantiation

In the development of the testing system architecture, we considered multiple remote servers, located in different countries that instantiate modules of the proposed architecture, and that concern different domains (video surveillance and broadcast).

The topology of the LINDO testing system, presented in Figure 7, is composed of a central server, located in Paris, and three remote servers, two of them are located in Paris (RS2 and RS3) and the other one in Madrid (RS1). Two remote servers are dedicated to video surveillance (RS1 and RS2) and they store, index and query video contents acquired in real time. The third remote server (RS3) manages multimedia contents for the broadcast domain.

In the following, we detail the particularities of each architecture module instantiation on each one of these servers, either remote or central.

The generic architecture of a remote server was instantiated for each one of the three remote servers. The instantiations maintained the architecture's modules, while adopting different implementations of their functionalities, according to each parteners' preferences and skills:

- For the *SM* module, a proprietary software developed in C language by one of the partners was adopted for a video surveillance remote server as well as for the broadcast remote server; for the other video surveillance remote server, a software produced by another partner was employed;
- Similarly, two different implementations (in Java and C#) of the *FEM* module were adopted for the two video surveillance remote servers, while the broadcast and the central server employed the Java implementation;
- The same *MDE* module was integrated in all the three remote servers. This module was developed in Java and uses the XML native Oracle Berkley DB XML[23] database for storing the metadata;
- The *Filtering* module was not included in the broadcast remote server;
- A Java implementation of the *SDC* was instantiated on each remote server and on the central server as well.

[23] http://www.oracle.com/technology/products/berkeley-db/xml/index.html

The characteristics of each remote server in terms of multimedia content and implicit indexing algorithms are presented in the following.

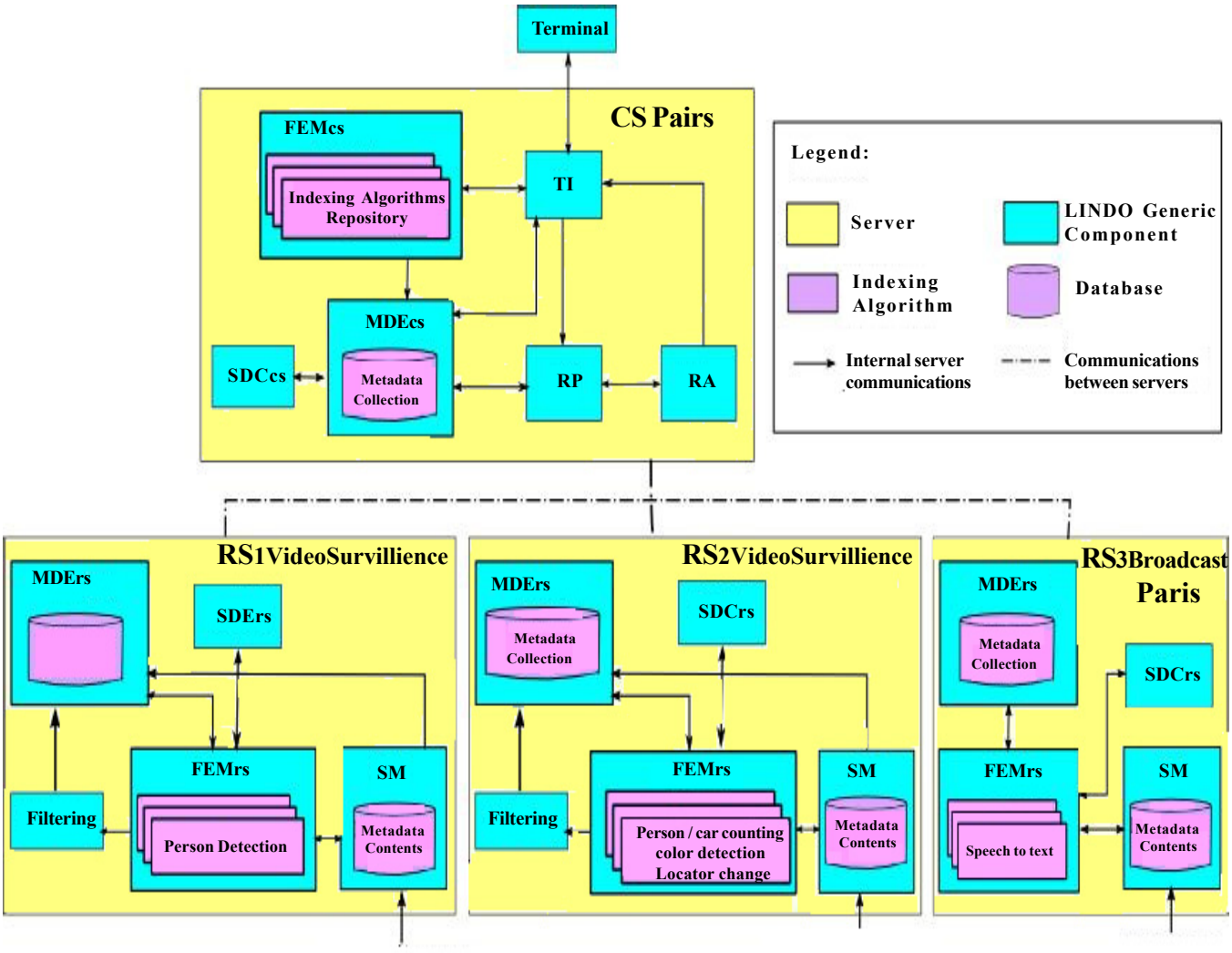

Figure 7. The LINDO testing system topology

The video surveillance remote server (RS2), installed in Paris:

• Manages multimedia contents acquired from two video surveillance cameras situated in Saint Lazare train station and watching the main hall and parking.

• Stores audio and video contents acquired in real time in the *SM* module, which in this case is the software developed in C language.

• Contains implicit indexing algorithms managed by the *FEMrs*. The indexing algorithms (executed on Windows and Linux environments) for video content are in charge with person and car counting and intrusion detection for indoor and outdoor environments. For audio content, the speaker change detection is available.

• Handles the metadata provided by the indexing algorithms in a uniform XML data format presented in section 4.3, as well as the descriptions of the installed indexing algorithms that respect the model described in section 4.2. All this information is stored by the *MDErs*.

• The *SDC* module contains an XML based description of the specific characteristics and context for this remote server as explained in section 4.1.

The second video surveillance remote server (RS1)**,** installed in Madrid:

- Manages and stores video contents acquired in real time from a video surveillance camera situated at the entrance into a security control room, using a software produced by a local partner;
- Contains an indexing algorithm for person and color detection in indoor environments. The description of this algorithm and its output follow the same formats as the algorithms installed on the Paris remote server;
- The *SDC* stores locally the description of the remote server.

The third remote server (RS3), designed for the broadcast domain:

- Stores video content resulted from BBC journals using the same software for the SM as the video surveillance remote server from Paris;
- Contains a speech-to-text indexing algorithm based on Microsoft technology, which processes the audio stream of video files. The output of this algorithm follows the metadata format defined in the project.

The **Central Server** complies with the architecture presented in Figure 1 and has the following characteristics:

- FEMcs manages the indexing algorithm global collection where, alongside with all the implicit indexing algorithms installed on the remote servers, a supplementary set of explicit indexing algorithms are installed (abandoned luggage detection [24], shape detection, color detection, shout detection, etc. [25])
- MDEcs manages multiple data: descriptions of each remote server, abstracts of the multimedia metadata from each remote server, descriptions of indexing algorithms.
- Contains also the TI, the RP and the RA modules.

The system is used by security agents that work in the train stations and policemen. In order to assure the secure access to the multimedia content and to respect the privacy laws, each user has different rights with respect to the access to the content and to the indexing algorithms. For example a security agent is not allowed to execute explicit algorithms or to see some multimedia content. In some real time emergency situations it is important for the agent to have access to resources that are necessary to him in order to solve the emergency. In these situations the access rights are by-passed, without damaging the security of the system.

7.2. System utilization examples

Let's suppose that a security agent enters the following queries:

Query 1: show me all *audiovisual content* related to *red bag* forgotten in *Saint Lazare train station, Paris*, on *Wednesday,* 29 *of February, between 10 a.m and 2 p.m.* on his PDA, in the train station. The query is analyzed and based on the location and on the requested multimedia type, the two remote servers from Paris are selected. Now the query is executed on the metadata summaries on the CS corresponding to the two selected RS. Responses are found on both servers and they are sent to the RA for ranking. At this moment the RA checks if the security agent has the right to access all multimedia results. In our case the agent is not allowed to listen to the audios from the video surveillance server. Thus, the audio parts of the content from RS2 from Paris are blocked.

Query 2: show me all *videos with women in red*, on *Wednesday,* 29 *of February*, on his PDA, in the train station. Because there is no location specified in the query, all RS are selected. After analyzing the metadata summaries no results are found. At this moment, the RP checks the rights of the agent in the current situation. He does not have the right to run explicit algorithms and his context does not describe an emergency situation. So, the explicit indexation is not accomplished and a message is displayed to the user in order to let him know that no results were found for his current context.

Query 3: show me all *audiovisual content with lost child dressed in blue* in *Saint Lazare train station, Paris*, on *Wednesday,* 29 *of February, between* 10 *a.m and* 2 *p.m.,* from the control room The query is analyzed and based on the location and on the requested multimedia type, the two remote servers from Paris are selected. Now the query is executed on the metadata summaries on the CS corresponding to the two selected RS. No results are found. The RP checks the rights of the agent in the current

situation. Normally he does not have the right to run explicit algorithms, but his current context, he is in the control room and his query is considered to belong to an emergency situation, unlocks temporarily this restriction. So, the RP selects the explicit indexing algorithms based on the user query as describes in section 5.2. More precisely the algorithm that detects persons is selected. This algorithm is executed on the video content that corresponds to the specified time period. After the execution of the algorithm the query is executed on the remote servers selected before and the results are transmitted to the RA. Because of this situation the audio content restriction is not taken into consideration and all results are shown to the user.

8. Conclusion and future works

In the context of the general process of distributed multimedia content management, this paper addressed concrete twofold problem concerning the following two deadlocks: (a) description formats heterogeneity and (b) big resources consumption. The solutions proposed for these two problems consist in: (1) adopting generic descriptions formats, (2) implementing a dynamic indexing process accomplished at acquisition time and query execution time (3) integrating remote site characteristics and context, as well as the user queries history in the dynamic selection of indexing algorithms. These solutions were implemented and validated in the LINDO project. Some system utilization examples are described in order to illustrate the functioning of the system.

A limitation of this approach is the increased query processing time. This problem can be addressed by classical query optimization methods (e.g., parallelization) that are not in the scope of this paper.

As future work, we will improve the indexing algorithm selection process by taking into account multiple characteristics from their descriptions (e.g., their complexity, execution time). In the case of complex indexing needs, it is often necessary to execute into a certain order a chain of algorithms (e.g., French words detection algorithm must be executed only after a positive detection accomplished by the speech detection algorithm). Therefore, we intend also to include some pre-conditions and post-conditions specifications in the algorithms descriptions and to exploit them in order to automatically determine such chains specific to a complex indexing need.

9. Acknowledgement

This work has been supported by the EUREKA project LINDO (ITEA2-06011).

References

[1] Burnett, I. S., Davis, S. J., Drury, G. M. (2005). MPEG-21 digital item declaration and Identification-principles and compression, *Multimedia, IEEE Transactions on*, 7 (3) 400-407.

[2] Roth, V., Peters, J., Pinsdorf, U. (2006). A distributed content-based search engine based on mobile code and web service technology, *Scalable Computing: Practice and Experience*, 7 (4) 101-117.

[3] Petkovic, M., Jonker, W. (2003). Content-Based Video Retrieval: A Database Perspective, *ser. Multimedia Systems and Applications*. Berlin: Springer Verlag, 25.

[4] Pietarila, P., Westermann, U., Jarvinen, S., Korva, J., Lahti, J., Lothman, H. (2005). Candela-storage, analysis, and retrieval of video content in distributed systems: Personal mobile multimedia management, *In*: *Proceedings of the IEEE International Conference on Multimedia and Expo* (*ICME*). IEEE Computer Society, p. 1557-1560.

[5] Merkus, P., Desurmont, X., Jaspers, E., Wijnhoven, R., Caignart, O., Delaigle, J.-F., Favoreel, W. (2004). Candela - integrated storage, analysis and distribution of video content for intelligent information systems, *in European Workshop on the Integration of Knowledge, Semantics and Digital Media Technology* (*EWIMT*)

[6] Agosti, M., Buccio, E. D., Nunzio, G. M. D., Ferro, N., Melucci, M., Miotto, R., Orio, N. (2007). Distributed information retrieval and automatic identification of music works in SAPIR. *In*: Proceedings of the 15th Italian Symposium on Advanced Database Systems (SEBD), p. 479-482.

[7] Michal, B., Fabrizio, F., Claudio, L., David, N., Raffaele, P., Fausto, R., Jan, S., Pavel, Z. (2010). Building a web-scale image similarity search system. *Multimedia Tools and Applications*. 47, 599-629

[8] Giroux, P., Brunessaux, S., Brunessaux, S., Doucy, J., Dupont, G., Grilheres, B., Mombrun, Y., Saval, A. (2008). Weblab : An integration infrastructure to ease the development of multimedia processing applications. *In*: The 21st Conference on Software and Systems Engineering and their Applications.

[9] Viaud, M.-L., Thièvre, J., Goëau, H., Saulnier, A., Buisson, O. (2008). Interactive components for visual exploration of multimedia archives. *In*: Proceedings of the 7th ACM International Conference on Image and Video Retrieval (CIVR). ACM, p. 609-616.

[10] Conan, V., Ferran, I., Joly, P., Vasserot, C. (2003). KLIMT: Intermediations Technologies and Multimedia Indexing. *In*: *T*hird International Workshop on Content-Based Multimedia Indexing (CBMI), p.11-18.

[11] Atarashi, R.S., Kishigami, J., Sugimoto, S. (2003). Metadata and new challenges. In: Proceedings of Applications and the Internet Workshops Symposium on , 4 (31), 395- 398, 27.

[12] Brut, M., Sèdes, F., Manzat, A. -M. (2009). A web services orchestration solution for semantic multimedia indexing and retrieval. *In*: *T*he 2nd Workshop on Frontiers in Complex, Intelligent and Software Intensive Systems. IEEE Computer Society, p. 1187-1192

[13] Hinds, N., Ravishankar, C. V. (1998). Managing metadata for distributed information servers. *In*: *Proceedings of the Thirty-First Hawaii International Conference on System Sciences*, p. 513-522.

[14] Laborie, S., Manzat, A.-M., Sèdes, F. (2009). Managing and querying efficiently distributed semantic multimedia metadata collections. *IEEE MultiMedia Special Issue on Multimedia-Metadata and Semantic Management*, 16 (4) 12-21

[15] Brut, M., Laborie, S., Manzat, A.-M., Sèdes, F. (2009). A Framework for Automatizing and Optimizing the Selection of Indexing Algorithms, *In*: The 3rd Conference on Metadata and Semantics Research (MTSR), p. 48-59

[16] Brut, M., Codreanu, D., Dumitrescu, S., Manzat, A.-M., Sedes, F. (2011). A distributed architecture for flexible multimedia management and retrieval. *In*: *Proceedings of Database and Expert Systems Applications* (*DEXA*), ser. LNCS, v. 6861. Springer Berlin / Heidelberg, p. 249-263.

[17] Haslhofer, B., Klas, W. (2010). A survey of techniques for achieving metadata interoperability. *ACM Comput. Surv*, 42, p. 7:1-7:37.

[18] Smith, J. R., Schirling, P. (2006). Metadata standards roundup, *IEEE MultiMedia*, 13, 84-88

[19] NISO (2004). Understanding Metadata. National Standard Organization (NISO) Press.

[20] Bailer, W., Schaullauer, P. (2008). Metadata in the audiovisual media production process. *In*: Multimedia Semantics- The Role of Metadata V. 101 of Studies in Computational Intelligence.

[21] Schallauer, P., Bailer, W., Troncy, R., Kaiser, F. (2011). Multimedia Metadata Standards. *In:* Multimedia Semantics- Metadata, Analysis and Interaction of Wiley.

[22] Samir, A. (2011). Multimedia metadata integration system. PhD thesis at Université Lille1 - Sciences et Technologies .

[23] Haidar, B., Joly, P., Haidar, S. (2005). A graph based approach to automatically chain distributed multimedia indexing services. *In*: Proceedings of the *Ni*nth IASTED International Conference on Internet and Multimedia Systems and Applications, p. 336 – 342.

[24] Gasteratos, A., Vincze, M., Tsotsos, J., Miezianko, R., Pokrajac, D. (2008). Detecting and Recognizing Abandoned Objects in Crowded Environments. *In*: Computer Vision Systems, LNCS, Springer, p. 241 - 250

[25] Snoek, C. G., Worring, M. (2005).Multimodal video indexing: A review of the state of the art. *In*: Multimedia Tools and Applications, 25, 5- 35.

[26] Hinds, N., Ravishankar, C. V. (1998). Managing metadata for distributed information servers, *In*: Proceedings of the Thirty-First Hawaii International Conference on System Sciences, p. 513–522.