Published in Proceedings of 2nd IEEE Metadata Conference, Silver Spring, Ma, septembrer1997.

Models for Metadata or Metamodels for Data?

Brigitte Kerhervé
Université du Québec à Montréal
Département Informatique
CP 8888, succursale centre ville
Montréal, Québec, Canada, H3C 3P8
Kerherve.Brigitte@uqam.ca

Olivier Gerbé
Université de Montréal
Département Informatique
CP 6128, succursale centre ville
Montréal, Québec, Canada, H3C 3J7
and DMR Consulting Group
1200 McGill College
Montréal, Québec, Canada, H3B 4G7
gerbe@iro.umontreal.ca

ABSTRACT

In this paper we examine modeling and metamodeling for metadata. We illustrate the need of metamodeling through three examples where metadata sets are used to efficiently process internal functions. We show that several levels of metamodeling are required in order to efficiently support essential functions provided by these systems. We argue that extensible metadata managers should support (i) several modeling levels, (ii) homogeneous manipulation of these different levels and (iii) extensibility of the corresponding models and metamodels. We introduce a pyramid of modeling levels for extensible metadata managers and we propose conceptual graphs as a homogeneous modeling formalism for the different levels.

1.0 INTRODUCTION

Metadata, generally defined as data about data, aim at facilitating access, management and sharing of large sets of structured and/or unstructured data. A lot of research efforts have been dedicated to the specification of metadata for data-intensive applications such as earth sciences, multimedia systems [1] or data mining systems. These efforts generally leaded to the development of application-oriented metadata sets and corresponding standards. Examples of such results are emerging standards for geographic information systems or library management systems.

We are now entering a new era for metadata specification and management. It clearly appears that efforts conducted for metadata sets specification is not sufficient and that research should now go in the direction of integration, federation and inter-operation of existing propositions and standards. Research in that direction includes issues in the design and development of:

- extensible metadata models that can be adapted to specific application domains;
- tools allowing integration and interoperation of metadata sets coming from different sources and represented using different standards;
- extensible metadata managers offering basic services to support essential functions such as

access, transfer, discovery or analysis for the development of specific applications.

These directions absolutely require important work on metadata modeling and more specifically on modeling levels and corresponding metamodels. Proposals for metadata models are requested to describe the content and the semantic of data in given applications. Among the different metadata sets proposed for applications such as geographic information systems or distributed multimedia systems, we can identify a common subset of metadata that is useful for both applications. Defining a core model for metadata is then a way to describe this common subset. But it is not sufficient. When we address the issue of extensibility of this metadata core model, we are obliged to work on an upper modeling level: on the concepts used to define the metadata model, that is the metamodel for metadata. Then, while addressing extensibility and interoperability, defining and manipulating different modeling levels is mandatory.

In the framework of a research project conducted at University of Québec at Montréal (UQAM) and University of Montréal, we are currently designing an extensible metadata manager, based on a core model for metadata and providing services for the development of distributed multimedia systems. While designing the corresponding metadata model, we were faced with modeling and metamodeling issues.

In this paper we examine modeling and metamodeling for metadata. We identify several modeling requirements for metadata managers. More specifically we show that extensible metadata managers should support (i) several modeling levels, (ii) homogeneous manipulation of these different levels and (iii) extensibility of the corresponding models and metamodels. We also point out that these requirements should guide the choice of an adapted knowledge representation formalism or data model formalism and we briefly introduce conceptual graphs as such a formalism.

We illustrate the need of metamodeling through three examples where metadata sets are used to efficiently process internal functions. The first example is the query optimization function in relational database systems where metadata describing the structure and the physical organization of data are used to efficiently process users queries. The second example deals with quality of service management in distributed multimedia systems where metadata describing the quality of multimedia documents are used to access and deliver documents according to user specified constraints. The last one presents knowledge management for corporate memories where metadata associated to processes and methods are used to generate documentation. In these different cases we show that several levels of metamodeling are required in order to efficiently support essential functions provided by these systems. We also point out that when extensibility is required for metadata sets and essential functions, operations on the different modeling levels should be processed.

The paper is organized as follows. Section 2 gives an overview of research issues in metadata management to provide extensibility and interoperability. Section 3 introduces our three examples and the corresponding metadata sets and models. Section 4 presents requirements for metadata modeling and metamodeling. Section 5 proposes a pyramid of modeling levels for extensible metadata managers and introduces conceptual graphs as a modeling formalism. Section 5 concludes and presents our future work.

2.0 RESEARCH ISSUES IN METADATA MANAGEMENT

Metadata is extensively used in systems and applications to mainly gain efficiency in access, transfer, share or process large amounts of data. Defining the corresponding metadata set is the first step towards implementing efficient systems or applications. In the past several years, different proposals have been made for specific needs encountered in applications and systems. Examples of such proposals are FGDC for geographic information systems[2], Dublin Core for Digital Libraries[3], CDIF for modeling tools[4] or MDIS for CASE systems[5]. Answering the needs of these applications and systems required similar research work on (i) data modeling, (ii) system implementation and (iii) tools development. While these proposals and standards were initially designed for specific communities, the open environment provided by the internet and the growing need to share information require from now on to focus on interoperability between data models, systems and tools. The challenge is to propose approaches for metadata management allowing the integration as well as the extension of existing proposals. A possible way to proceed is to adopt a bottom-up strategy offering tools for integration, federation and inter-operation of metadata sets and corresponding procedures. Some proposals have already been made in that sense, the

Dublin Core is an existing proposal of metadata common subset in the field of digital libraries. Nevertheless such a metadata subset is generally very reducing. An alternative solution is to adopt a top-down approach where a generic metadata set is set-up and tools are provided to adapt it to the needs of systems or applications. Figure 1 depicts these two alternatives.

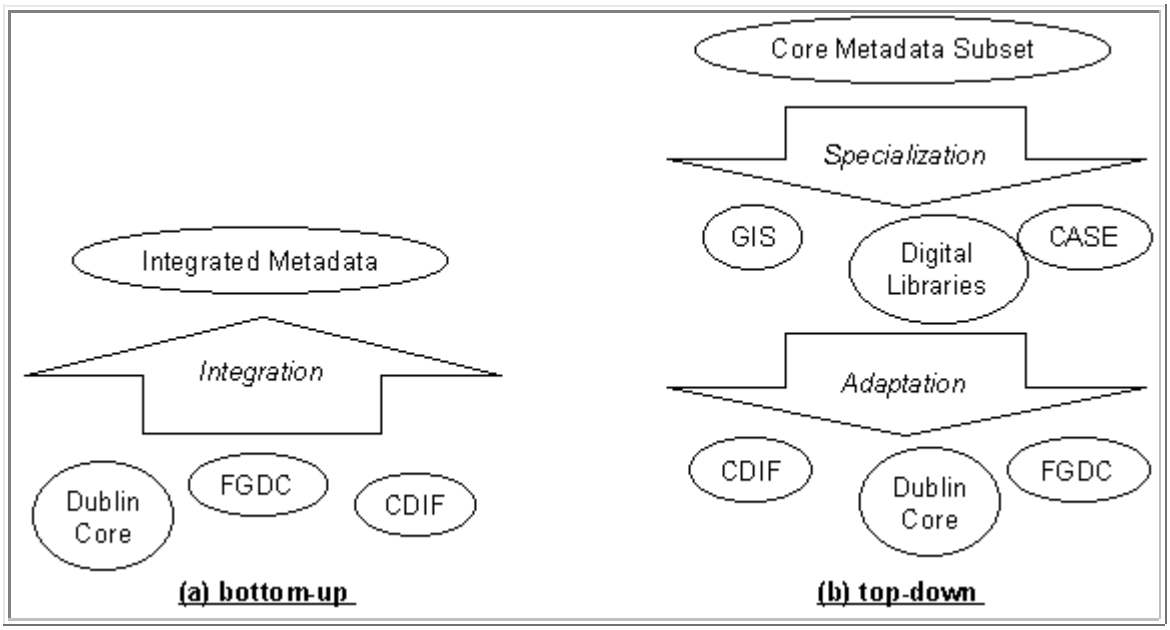

Figure 1. Alternatives for Metadata Sets Integration and Inter-Operation.

The second approach seems more promising for several reasons. First, since a significant overlap has been identified among the current existing proposals, a common subset of metadata should be defined as the common denominator between applications and systems. Second, common basic functions as well as specific tools are traditionally implemented for metadata management, they can be defined as basic services to be provided. Third, emerging applications in Global Information Infrastructure will definitively lead to new proposals for metadata sets; these proposals should be rapidly integrated to existing frameworks. Research work is presently done in that direction, and some of them such as Dublin Core or MDIS initiatives follow this approach but nevertheless, they are defined for a specific application domain. In particular if we detail these two propositions, we find similar elements in the metadata set such as *relation* in Dublin Core and *relationship* in MDIS. We claim that the upper modeling level depicted in Figure 1b is required to integrate application domains. This level has to be defined through a clear identification and classification of metadata elements[6] to produce a metadata core model.

Associated to this metadata core model should be defined a set of basic operations as well as the corresponding tools allowing to extend and adapt this core metadata model. While the core metadata model and the associated operators can be defined as a metadata registry[7], the tools for extension and adaptation lead to specify extensible metadata managers. Extensible metadata managers can be defined as complex systems offering basic services to define and support essential functions such as access, transfer, discovery or analysis for the development of specific applications. These functions are based on an extensive use of metadata. We can consider the development of an image management system where the focus in on querying according to the content. This system is based on metadata extracted from images, automatically or manually. Some of the metadata are well known metadata such as the image format, color or resolution that are part of the metadata core model, while others are more sophisticated and depend on the technology and the algorithms used to extract information from images. In that case, the core metadata model should be extended to incorporate these new metadata. We here address the issue of extensibility, that is to be able to add new concepts to the existing model, to integrate them in the basic functions and to share them with other systems.

Designing extensible metadata managers requires to address modeling and meta-modeling issues in order to provide interoperability among existing metadata sets as weel as extensibility to existing propositions.

Our approach is to deeply understand metadata management issues in existing systems in order to propose a core metadata model allowing interoperability and extensibility.

3.0 EXAMPLES

Metadata management has been identified as an important issue while designing and implementing applications using large volumes or complex data. Nevertheless, metadata management has also been introduced in different management systems such as database systems, distributed systems or knowledge management systems without explicitly referring to the concept of metadata. In such systems, a clear distinction is made between the data elements and the data structures and these two modeling levels are manipulated with different tools. But it is important to have a coherent view on these different levels. Thus such systems should be revisited with a metadata management perspective in order to be able to address fundamental issues for metadata managers with the experience gained in the development of these systems.

This section illustrates the use of metadata in three different systems: relational database systems, quality of service managers and corporate memories management systems. In the first case, metadata describing the structure and the organization of data elements are used to efficiently process user's queries. The second one is concerned with distributed multimedia systems where quality of service metadata are used to access and deliver documents according to user specified constraints. Last, we present knowledge management for corporate memories where metadata associated to processes and methods are used to generate documentation. In these different systems we show that metadata are generated from different modeling levels and thus, it is very important to clearly identify on which level the systems works.

3.1 RELATIONAL DATABASE SYSTEMS

Relational database management systems (DBMS) are widely recognized as a major technology for data management. These systems are based on the relational model introduced in 1970[8] and offer basic functions to efficiently store and access large amounts of data, as well as sophisticated languages for data definition and manipulation. Research efforts in that field has lead to the implementation of efficient systems supporting the SQL standard[9] as the definition and manipulation language. Commercial products implement sophisticated algorithms to efficiently process essential functions such as query optimization, concurrency control or reliability. The data model supported by relational database systems is based on the unique and simple concept of *relation*, and implemented as *tables* containing *tuples*. The data of an application are stored in a *database*, that is a set of tables manipulated with the SQL language. As an homogeneous way of defining and manipulating databases, relational DBMS use the concept of meta-database that is, a database describing other databases. The meta-database is itself defined as a set of tables. Figure 2 shows the different levels that are found in traditional relational DBMS.

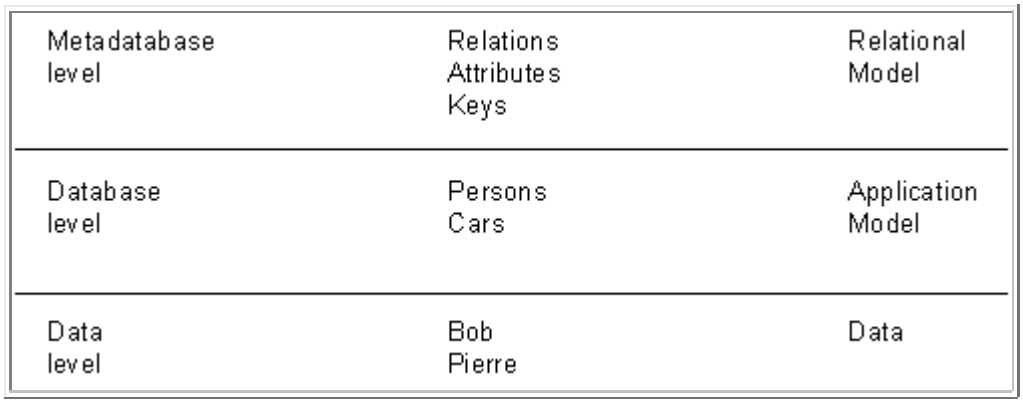

Figure 2. Modeling Levels in Relational DBMS.

In this figure, for clarity reasons, we defined the meta-database as only composed of the tables TABLES, ATTRIBUTES, KEYS and INDEX. Other tables generally compose the meta-database of commercial relational DBMS products. These four tables contain data describing the application database. In particular the table TABLES describe the name of the tables that constitute the application database, ATTRIBUTES,

KEYS and INDEX describe respectively the attributes, the keys and the index of the application database. The meta-database is populated while the database administrator creates the database in giving the set of tables that compose it.

The meta-database is the kernel of the database system. For executing almost all functions, the DBMS accesses the meta-database to find pertinent information on the tables. The meta-database can the be considered as the set of metadata required to efficienlty process internal functions. As an example let us consider the SQL query : "SELECT Name, Age FROM Person WHERE Town= "Montreal"". Before executing this query on the table PERSON, the DBMS check in the meta-database the existence of the table PERSON and the attributes Name, Age and Town. In that case, the DBMS uses information on the structure of the database, that is information constituting the database schema. To efficiently access the tuples of the PERSON table, the DBMS uses other tables of the meta-database such as INDEX or CLUSTER if it exists. We can the make a distinction between several parts of the meta-database: metadata describing the structure of the application database (the schema) and metadata describing the content of the database and the physical properties of the database such as index or clusters. Relational DBMS use the first subset as metadata for semantic controls and operations while the second subset is used for optimization and efficient execution.

If we consider relational DBMS from a metadata management perspective, we can consider the meta-database as the set of metadata used for internal mechanisms implementation. Operations on the meta-database include traditional operations to generate and record the set of metadata, to organize and efficiently access it, to modify metadata. It also clearly appears that evolutions to the internal mechanisms of the database system require extension to the metadata, that is to the meta-database. As an example, we can see the integration of distributed database functionality's in a relational DBMS through the addition of several tables such as FRAGMENT and LOCALIZATION into the meta-database. Associated to these new tables are defined operations to access and efficiently manage the extended metadata set.

Figure 3 shows the distinction between database, meta-database and the different parts of the meta-database and illustrates how the meta-database is used to process user's queries. With the help of this example, we see that metadata management is the kernel of database systems and that it could be revisited with a metadata management perspective, more specifically to enhance relational database functionality's. In particular in the context of interoperability of different DBMS such as relational, object or hierarchical, it becomes necessary to support other models to define the metada-database .

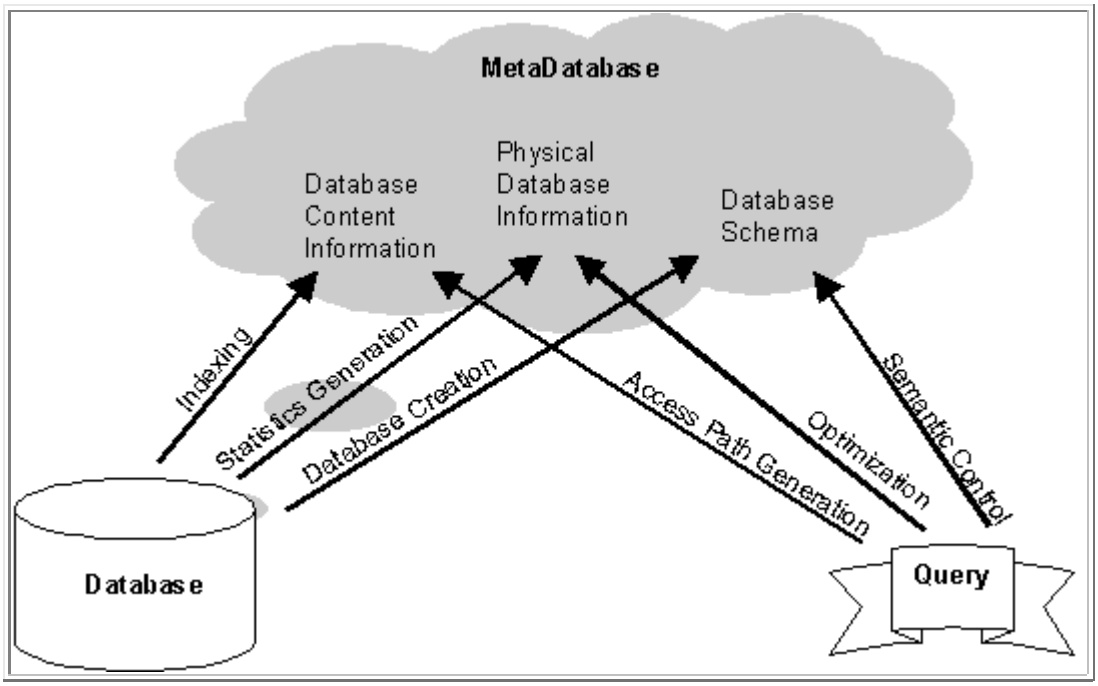

Figure 3. Metadata Management in Relational DBMS.

QoS MANAGERS

In the framework of a research project funded by the Canadian Institute for Telecommunication Research (CITR) we study enabling technologies for distributed multimedia systems and applications such as news on demand or teleteaching systems[10]. More specifically, we focus on quality of service (QoS) negotiation and adaptation in distributed multimedia systems. We aim at proposing approaches and algorithms to satisfy QoS constraints specified by the users.

The concept of Quality of Service was originally introduced in computer communications to mainly characterize data transmission performance[11]. QoS management is an essential function aiming to control and guarantee the level of quality that the distributed multimedia system is able to offer to the user. During a specification step, the user specifies his requirements which may concern system performance such as the delay needed to transfer objects, the quality of information provided, e.g. image quality: black and white or color, as well as financial costs attached to document delivery such as the costs charged to obtain a research article. The system then transparently operates to deliver the requested level of quality and for that purpose adapt the user's requirements to the various constraints supported by the system components: client machines, database systems, server machines and transport system.

In the framework of our project, we have defined the metadata set required for QoS management[12]. The metadata set includes three categories of QoS information, respectively associated to the user, to the system components and to the multimedia document. A user profile describes user preferences in terms of (1) QoS settings for video, audio, still images and text, (2) cost he is willing to pay for a given quality, and (3) time constraints, such as the maximum delivery time. QoS parameters associated to system components describe the distributed multimedia environment and its technical characteristics such as memory size, available formats, screen quality or system performance. It also includes parameters which are dynamically evaluated such as: throughput, transfer delay and guarantee. The multimedia documents have intrinsic QoS characteristics used during the different steps of QoS negotiation.

In the multimedia document model we defined, a multimedia document is composed of several monomedia objects synchronized with each other and possibly shared by different multimedia documents. A multimedia document also includes a *Price* and information allowing the expression of search conditions on the multimedia database, that are *Registration* and *Description* attributes, traditional descriptors such as keywords, author, date etc.... A monomedia object is defined in a particular medium: a text, a still image, an audio sequence, a graphic, a video sequence or an audio-video sequence and stored using given format and quality. The quality of a given monomedia document is defined by static parameters depending on the kind of monomedia medium or referring to its physical localization. The QoS parameters are thus specialized for image, text, audio and video and give, for instance, the format of the coding, the size of the file, the color of a video.

Querying distributed multimedia databases require several steps: initialization of the querying environment, searching to isolate potential documents and last, display of the pertinent documents. The initialization step must then be seen as the set up of the user environment concerning preferred information providers as well as quality of service requirements. The search phase of the request consists of the pre-selection of a set of multimedia documents of potential interest for the user. It is processed in selecting the documents according to criteria expressed on content description metadata such as keywords, author or date and aims at minimizing the volume of transferred data. In our system, this step builds as result, all the metadata on the content of the document as well as the metadata required for QoS negotiation. This set of metadata consists of the structure of the multimedia objects and the set of QoS parameters for each of their components. After having reduced the search space, and as soon as the user asks for accessing a document, the QoS manager initiates negotiation in order to set up a transmission contract satisfying the required QoS level. This phase implies (1) to determine the monomedia objects that will be transferred and (2) to set up the required system configuration, that are the components involved in the transfer. The QoS manager evaluates dynamic QoS parameters such as resource availability or system load and determines the components and corresponding resources that are requested for the transfer. Once the negotiation is completed successfully, the content of the multimedia object is transferred to the client for display.

Figure 4 shows the different parts of the metadata set that has been defined and implemented for QoS management. We can see that QoS information are included in the database itself as characteristics of

data elements, in the database schema, where the structure of the database is described and in the management information base (MIB) describing the system components and the user's profiles. We can then consider the QoS metadata set as a view on the different levels of the distributed multimedia database. It clearly appears that integrating new QoS functions such as adaptation requires to enhance the metadata set in the different levels: in the meta-database part in including alternatives to monomedia documents, in the data set in setting quality to these alternatives and in user's profile in giving preferences on the adaptation decisions.

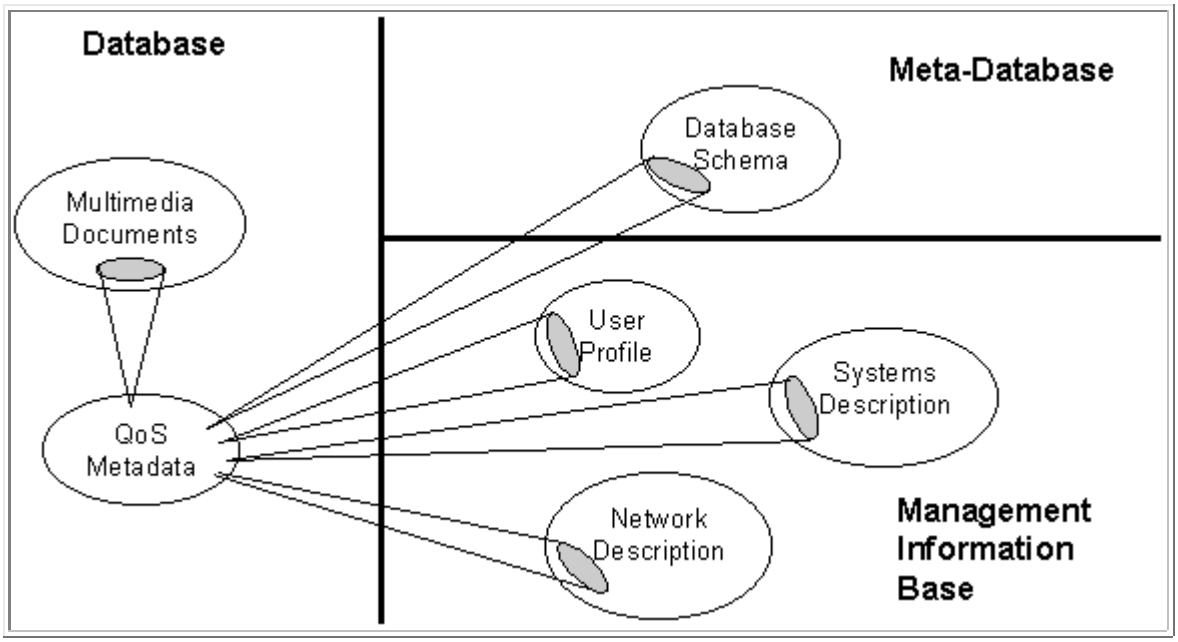

Figure 4. Metadata for QoS Management.

3.3 CORPORATE MEMORY MANAGEMENT SYSTEMS

DMR Consulting Group Inc. has initiated the IT Macroscope project[13], a joint research project that aims to develop methods allowing organizations: i) to use IT to increase competitiveness and innovation in both the service and product sectors; ii) to organize and manage IT investments; iii) to implement information system solutions both practically and effectively; and iv) to ensure IT investments are profitable. Associated to this set of methods, was identified the necessity to develop tools to fulfill the needs for designing and maintaining methods, designing training courses, and managing and promoting IT Macroscope products. These tools are based on the concept of corporate knowledge. Corporate knowledge is made up of strategies, visions, rules, procedures, policies, traditions and people and its management requires the acquisition, storage, the evolution and dissemination of knowledge acquired by the organization[14]. Corporate memories integrate these functions.

In the framework of this project, we adopted an approach based on the concept of repository, defined by [15] as "a class of applications that can be characterized as storing and managing both data and metadata", and thus by extension managing knowledge and meta-knowledge. The DMR corporate knowledge repository, called the Method Repository, captures, stores, retrieves and disseminates throughout the organization all the consulting and software engineering processes and the corresponding knowledge produced by the experts in the IT domain[16,17]. It also allows adaptation to the particular needs of an organization, and evolution at acceptable levels of cost. The ultimate goal of the repository is to facilitate the management of methods and projects. To facilitate the comprehension, we focus on system development projects and corresponding methods. Methods give rules and procedures to conduct projects. Projects generates objects such as documentation, or code.

During the early stage of the development, the main issue we faced while developing the DMR corporate knowledge repository was to support the different levels of knowledge representation and by the way to choose a knowledge representation formalism. For the management of methods and projects, have

identified four modeling levels: project data, project model, method model and last the model of knowledge. Figure 5 depicts these four levels.

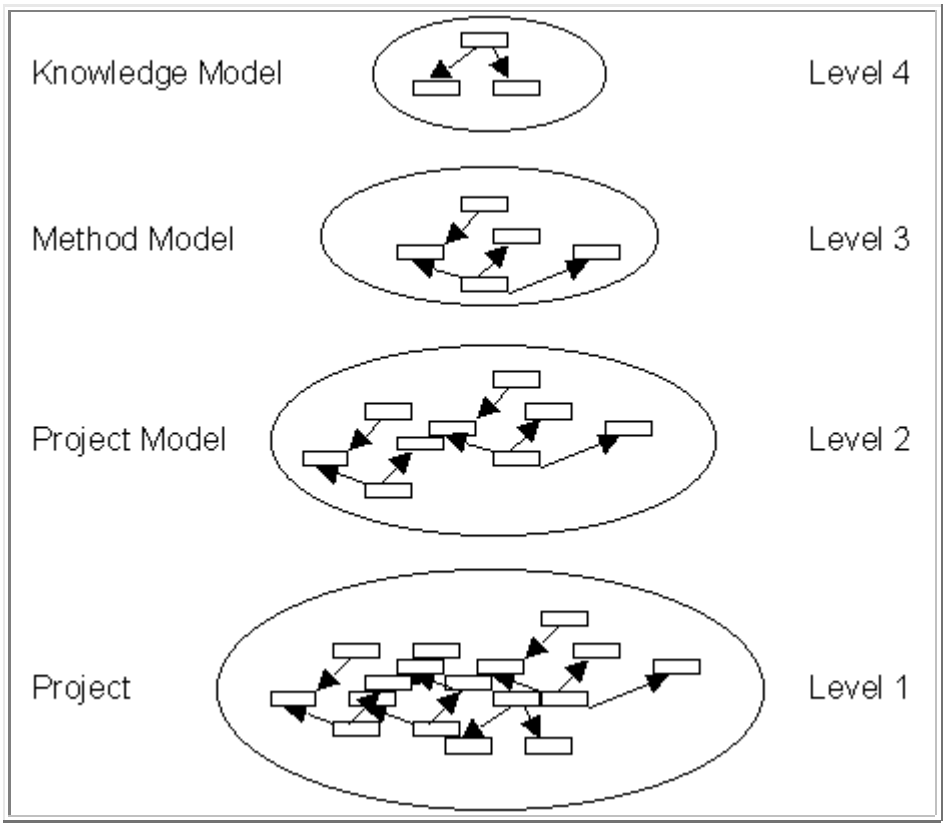

Figure 5. Modeling Levels in Corporate Memory.

Level 1 is the project data level. It contains all the objects generating during a project and constitutes the project database. For system development projects, they can be documents, activities, code or specifications. At this level we find metadata describing characteristics of the project objects themselves such as document or code size, document location or history of activities. Project objects are defined according to a model defined in the second level and called the project model level. The project model can be considered as the project database schema and is composed of several concepts. For system development projects, the concepts are written in italics in the following description. Projects produce *deliverables* (documentation, specification, code, Ö). A deliverable has sections or *information elements*. Each information element documents a part of the system or concerned *domain* and may be produced using *techniques*. *Activities* produce deliverables and are guided by techniques pertinent to the production of information elements. Concepts of the project model, of course depend on the type of project that is managed in the system. We can easily imagine that for manufacturing projects, these concepts will be different.

We have seen before that projects are conducted according to methods, that means that projects are derived from methods, and that concepts in a project model derive from methods. That leads us to define an upper modeling level, namely level 3 which is the method model level. The method model defines the concepts that are used in the method. In the system development context, methods are composed of *deliverable types*, *information elements types*, *activity types*, *technique* and *concept types*. This method level allow us to define instances of methods for specific needs encountered in the software industry. It allows also the evolution and refinement of existing methods.

Level 4, the knowledge model, is the ultimate layer and is the description of knowledge objects. The knowledge model allows the definition of concepts for the three previously defined levels and defines the formalism we chose to represent knowledge. It is composed of concepts, conceptual relations and graphs that structure concepts and conceptual relations.

These different modeling levels constitue a list which is not exhaustive. The project objects, i.e., deliverables may be seen as model of the system under development and a lower layer may be added, the system data layer. As the opposite side, when considering the knowledge model, it should be possible to define a higher level, nevertheless the human understanding limits this number of layers.

4.0 REQUIREMENTS FOR METADATAMODELING

In the previous examples, we can see that for each system, metadata is managed according to the needs but without a coherent and complete view of the different elements that compose the metadata set. That leads us to express several requirements concerning the modeling issues of extensible metadata managers. Extensible metadata managers should support (i) several modeling levels, (ii) homogeneous manipulation of these different levels and (iii) extensibility of the corresponding models and metamodels.

In the previous section, we have seen that different metadata types are used to efficiently implement management systems. These metadata types are produced at different modeling levels. In relational database systems, metadata exist at the three different modeling levels we have presented: the data level, the database level and the meta-database level. In QoS managers, metadata come from the data level as well as from the management information base. In corporate knowledge repositories managers, metadata come from the project or method levels. Generally the only distinction which clearly appears is the one done between the data level and the model level, that is between the two lower levels. Generally, the model level is used while working on the structures of the data from the lower level. Besides, the information considered as data in one level can be considered as metadata in the upper level. As example, the application model described in a database schema, can be considered as metadata to efficiently manage and access the application, while it can be considered as simple data for a tool in charge of integration and interoperation of databases. Thus it appears fundamental to introduce several modeling levels in metadata managers and to clearly identify the level on which tools and operators are working.

The different modeling levels require specific operations for metadata generation, access or management. In relational database systems, metadata coming from the database level are generated through the data definition language and constitutes the database structure description. SQL standardized the data definition language and consequently the structure of the meta-database. Nevertheless, since instances found in the data level are manipulated using the data manipulation language, we can see that the different modeling levels are manipulated differently and sometime it is difficult, nay impossible to manage or access data from both levels homogeneously, that is with only one query. And yet, recursivity in the perception of data and metadata requires to propose homogeneous ways to manipulate the different modeling levels.

As we have introduced in section 2, a lot work in the field of metadata management has conducted to standards propositions and implementation, and such work is still in progress. The new standards that will be proposed shortly will have to be integrated in existing frameworks as well as adapted to existing application domains. The fully distributed environment provided by the internet also require the interoperability between standards and existing implementations. That leads us to express an important requirement on the extensibility and adaptation of models and metamodels. The kernel of metadata managers should then use a representation formalism allowing this extension, adaptation and interoperability. A root modeling level, defined as the knowledge level should be introduced to support interoperability and extensibility.

5.0 MODELING LEVELS FOR EXTENSIBLE METADATA MANAGERS

The requirements previously defined for modeling issues in extensible metadata managers require to introduce a modeling architecture and a knowledge formalism allowing to describe the different levels of this architecture. In this section we present the four level modeling architecture and the metadata that can be produced in these different levels. Then we introduce conceptual graphs as a our knowledge representation formalism.

In section 3 we have seen that relational database systems support three modeling levels: data, application model and relational model. Figure 6 shows a four level modeling architecture that should be supported by extensible metadata managers. This architecture introduces modeling and metamodeling allowing to generate metadata essentially on the structure of data elements. The different modeling levels also enables

to incorporate models and metamodels for metadata.

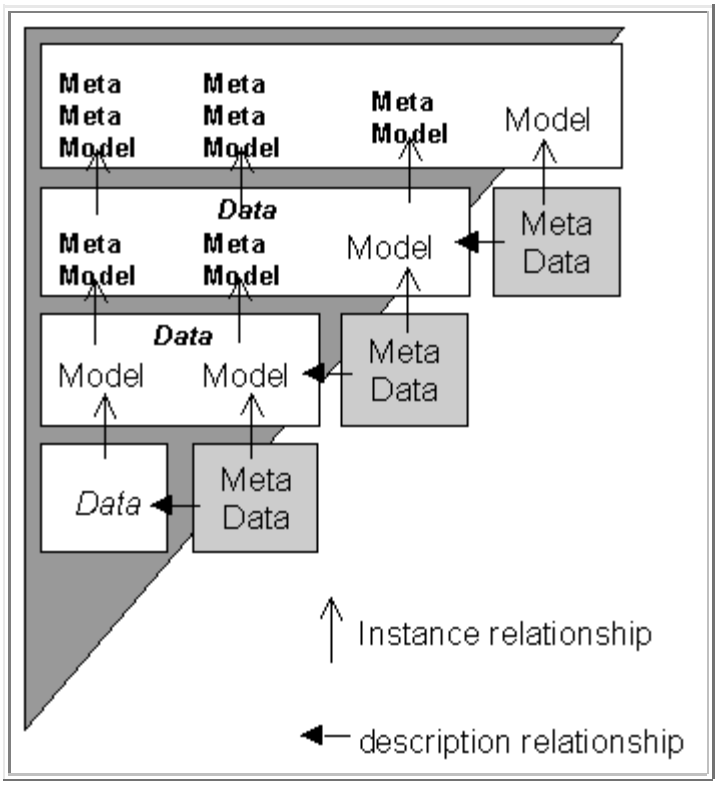

Figure 6. Four levels Modeling Architecture.

The data level corresponds to the instances that are manipulated in the system. These instances can be the tuples in relational database systems, the multimedia documents in distributed multimedia systems or the project in corporate memory managers. At this level, the type of metadata that can be extracted concerns the physical aspects of the data such as the volume, or the localization. Metadata concerning the data content can also also be generated at this level.

The next level concerns the model description, that is the concepts used to describe data and metadata. In relational database systems, the model level corresponds to the description of the concepts used in the application. At this level, metadata concerning the structure can be extracted. In relational systems this level corresponds to the instances of the metadatabase. The relational system extensively uses these instances for internal functions. In distributed multimedia systems, this level describes the composition of the multimedia documents an then can be used to optimize the transfer of components. At this level, we also find the model for metadata, that is the metadata schema or structure of metadata elements.

The meta-model level is the level defining the model formalism that is used in the system. In most systems, this level is the upper and last level. It describes the concepts used to represent information in the lower levels. In relational database systems, we here find the concepts used to implement the relational model. In corporate memory management system, at this level we find the concepts used for method representation. At this level, the type of metadata that can be extracted concerns the formalisms and its specificity. Such metadata are used for interoperability of tools, methods or systems.

The upper level, namely the meta-meta model level is the root modeling level allowing an homogeneous representation of the other levels. This level allows an easier interoperation between models of the lower level.

To support the architecture we have introduced, we need to choose a representation formalism that would allow an homogeneous representation of the different modeling levels as well as the manipulation of the different levels with the same language. We have chosen conceptual graphs for their expression power

and the possibility they offer to manipulate both types and instances. Conceptual graphs are a formalism whereby the universe of discourse can be modeled by concepts and conceptual relations. A concept represents an object of interest or knowledge. A conceptual relation makes it possible to associate these concepts. Conceptual graphs were developed by John Sowa in the early 80s[18]. They are a system of logic based on existential graphs and semantic networks.

A *concept* is a representation in mind of an object of the universe of discourse. A concept is the association of two referents: one refers to a type and the other refers to the object itself. For example, the mug on my desk can be seen in one context as a manufactured object and in another context as a piece of artwork. In that case there are two different concepts: `[#MANUFACTURED_OBJECT:#MY_MUG]` and `[#ART_WORK:#MY_MUG]` depending on the context.

A *conceptual relation* represents an association among an ordered set of one or two concepts and refers to the role played by each concept in the association.

A *conceptual graph* is a finite, connected, bipartite graph where the two kinds of nodes are concepts and conceptual relations, and where every conceptual relation has one or more arcs, each of which is linked to some concept.

As we have seen, a concept is the association of two markers: one for type and one for instance. Changing the position of an instance marker from left to right promotes it to a type marker. Figure 8 illustrates how an individual may be seen either as a type or as an instance.

```
[CONCEPT-TYPE:FEDERAL-ORGANIZATION]-
        (SUBTYPE)->[CONCEPT-TYPE:ORGANIZATION]
        (SYMB)<-[CATEGORY-LABEL:'federal organization']
        (DEFINED-BY)<-[GRAPH: [FEDERAL-ORGANIZATION:∀]-
                              (SYMB)<-[LABEL:*]
                              (ATTR)<-[GOVERNEMENT:*]
                              (MEMBER)<-[FEDERAL-ORGANIZATION:*]

[FEDERAL-ORGANIZATION:ENV-AGENCY]-
        (SYMB)<-[LABEL:'environment agency']
        (ATTR)<-[GOVERNMENT:'canada']
        (MEMBER)<-[FEDERAL-ORGANIZATION:AIR-AGENCY].
```

Figure 7. Type or Instance.

When the marker is on the left side `[CONCEPT-TYPE:FEDERAL-ORGANIZATION]`, it represents an instance of the right side type, and when the marker is on the right side, it represents a category as in `[FEDERAL-ORGANIZATION:ENV-AGENCY]`. That then allows to manipulate both instances and types.

6.0 CONCLUSION

In this paper, we have addressed modeling issues for extensible metadata managers. With the help of three examples, we have shown that metadata come from different modeling levels. We have then pointed out the necessity to introduce modeling levels, to make clear dictinctions between them and to propose homogeneous manipulations of these different levels.

We are presently working on the specification of a core model for metadata. This core model is built on the four level modeling architecture we have presented in this paper. This core metadata model supports a classification of metadata associated to the formalism, representation, the structure, the content, the

storage and the evolution of data. This classification allows to extend and adapt the core model for specific applications or systems such as distributed multimedia systems of digital libraries. We will propose tools based on database schema evolutions to implement extensibility and adaptability of the core model for metadata.

7.0 REFERENCES

[1] Klas, W. and A. Sheth, *Metadata for Digital Media: Introduction to the Special Issue.* ACM Sigmod Record, 1994. 23, no 4(december 1994): p. 19-20.

[2] FGDC, *Federal Geographic Data Committee* 1994, Washington DC, USA.

[3] Weibel, S., *Metadata: the foundations of resource description.* Digital Library Magazine, 1995. .

[4] CDIF, *CASE Data Interchange Format* URL: http://www.cdif.org

[5] MDIS, *Metadata Interchange Specification* URL: http://www.he.net/~metadata/standards.

[6] Bohms, K. and T. Rakow, *Metadata for Multimedia Documents.* ACM Sigmod Record, 1994. 23, no 4(december 1994): p. 21-26.

[7] Olken, F., *Joint Workshop on Metadata Registries* 1997, URL: http://www.lbl.gov/~olken/EPA/Workshop/call.html.

[8] Codd, E., *A Relational Model for Large Shared Data Banks.* CACM, 1970. 13(6).

[9] SQL, URL: http://www.jcc.com/sql_stnd.html

[10] Wong, J.W., et al., *Enabling Technology for Distributed Multimedia Applications.* IBM Systems Journal, 1997. 36(4), 1997 (in press).

[11] Hutchinson, D., et al., Quality of Service Management in Distributed Systems, in *Network and Distributed Systems Management,* M. Sloman, Editor. 1994, Addison-wesley: p. 273-303.

[12] Kerhervé, B., et al. Metadata Modeling for Quality of Service Management in *Distributed Multimedia Systems. in IEEE Metadata.* 1996. Silver Spring (MD), USA:

[13] DMR, *DMR Macroscope1996*, DMR Consulting Group.

[14] Stein, W.E., *Organizational Memory: Review of Concepts and Recommandations for Management*. International Journal of Information Management, 1995. 15(1): p. 17-32.

[15] Silberschatz, A., M. Stonebraker, and J. Ullman, *Database Research: achievments and opportunities for the 21st century.* ACM Sigmod Record, 1996. 25(1).

[16] Gerbé, O. and M. Perron. Presentation Definition Language using Conceptual Graphs. in *Peirce Proceedings Workshop.* 1995. Santa Cruz (CA) USA:

[17] Gerbé, O., B. Guay, and M. Perron. Using Conceptual Graphs for Methods Modelings. in *4th International Conference on Conceptual Structures. 1996.* Sidney, Australia:

[18] Sowa, J.F., *Conceptual Structures: Information Processing in Mind and Machine.* 1984, Addison-Wesley.