– METIS –
A Flexible Database Foundation for the Unified Management of Multimedia Contents

Ross King*, Niko Popitsch*, Utz Westermann†
*Research Studio Digital Memory Engineering
Thurngasse 8/20, 1090 Vienna, Austria
{ross.king, niko.popitsch}@researchstudio.at
†Dept. of Computer Science and Business Informatics
University of Vienna, Liebiggasse 4/3-4, 1010 Vienna, Austria
gerd-utz.westermann@univie.ac.at

***Abstract*— Multimedia database systems largely focus on the management of media of one particular type and suffer from inflexible architectures, which makes it difficult to adapt them to individual application needs. This paper gives an overview of METIS, a flexible multimedia database foundation for the unified management of media of arbitrary types, characterized by profound customizability. METIS can be customized to feature any media types, metadata attributes, and associations desired for media description and classification. The system offers frameworks supporting the integration of any query operators, similarity measures, and/or feature extraction algorithms required for media retrieval. METIS makes use of a persistence abstraction layer that allows one to change storage back-ends and includes a highly customizable web front-end for administration and media management tasks. Along with the concept of semantic packs, which allow the bundling of domain-specific customizations, METIS constitutes a ready-to-use but nevertheless highly adaptable database foundation for a wide range of multimedia applications.**

I. Introduction

Among the plenitude of multimedia database systems that have emerged throughout the recent years, most of the systems focus on the management of media of a single type, such as videos (e.g., [1], [2], [3]) or images (e.g., [4], [5], [6]). But many applications – consider, for example, an e-learning environment with various multimedia educational material like video recordings of lectures with associated presentation slides or an online philatelic database giving access to additional multimedia information such as documentary films on postage stamp images – require a unified and integrated management of media of multiple types. Due to their isolated view of a single media type and their heterogeneous models applied for media description and indexing, however, the mere combination of several media type-specific database systems will not result in a unified, integrated media management, but rather in a set of isolated silos of media of a particular type [7]. Truly *multi*media database systems that offer a unified and integrated management of media of different type are still found rarely in the field [8].

A further problem we find with current multimedia database systems is that they often focus on a single application domain such as news (e.g., [9], [10]) or medicine (e.g., [11], [12]) and are based on more or less hard-wired architectures that are tailored to the requirements of that domain. But in order to provide a foundation for a wider range of applications, a multimedia database system must show profound flexibility and customizability in order to be adaptable to individual application needs, which may differ substantially: the above mentioned e-learning environment certainly must apply schemes for media classification and description that differ from the philatelic database as well as different features, similarity measures, and index structures for media retrieval. This argument for flexibility similarly applies to the back-end and front-end of a multimedia database system: a company running the e-learning environment for the education of its employees might demand the reliability of a commercial relational database server for storage and the ability to address media already stored in some media server, whereas the provider of the philatelic database, suffering from a tight budget, might be

perfectly satisfied with keeping all data in the file system. It is also difficult to provide generic user interfaces for media annotation and retrieval that are acceptable for both applications.

In the present paper, we give an overview of METIS, a general-purpose multimedia database system for the unified and integrated management of media of different kinds that follows the ideal of flexibility and offers customizability at all architectural levels:

- The *system core* supports the flexible definition of arbitrary media types, metadata attributes, and associations, which are necessary for media categorization, description, and interrelation. It also supports the integration of new data types for the metadata attributes, as well as new functions and operators for media retrieval and automatic feature extraction.
- The *storage back-end* is built on top of a persistence abstraction layer which allows one to operate the system on a large-scale database server or with small-scale file storage. It further provides an extensible locator mechanism for transparent access to media in different storage systems and locations.
- The *visualization layer* provides generic user interfaces for system administration and media management tasks. It is based on a framework that supports the rapid customization of these interfaces for many different levels of application needs.

Thus, METIS establishes a highly generic database foundation usable within a wide range of multimedia applications. In order to manage the complexity associated with this flexibility and allow users to set up working systems quickly, METIS additionally features the central concept of semantic packs, roughly comparable to extenders of object-relational database systems. Semantic packs are transportable and easily-deployed bundles that contain customizations from *all levels* of the system for an application domain. This includes:

- domain-specific definitions of media types, metadata attributes and associations,
- domain-specific implementations of data types, query operators and feature extraction algorithms,
- implementations of locators for additional storage locations and customizations of the web front-end desirable for the domain.

In the remainder of this paper, we discuss the architectural components of METIS in more detail: Section II treats the core of the system, Section III details the persistence abstraction layer, Section IV covers the visualization layer of METIS, Section V explains the concept of semantic packs, while Section VI gives an overview of our current and future work on the METIS architecture. Section VII relates METIS to other approaches in the field. Section VIII concludes the paper with a summary.

II. METIS Core

In this section, we provide an overview of the core of the METIS system, with an emphasis on the flexibility and customizability of the framework.

We have implemented the METIS core as a Java servlet, easily deployable as a web archive (WAR). In this manner METIS is not only platform independent, but can also be deployed in a variety of environments, ranging from Apache's Tomcat (currently used by our prototype) to commercial application servers, thus contributing to the flexibility of the system.

In the following, we describe the METIS data model available for the management of media objects (II-A). The data model is extended by the concept of complex media objects (II-B), followed by a short summary of functions and operators (II-C). We are concluding this section with a brief discussion of the system's query processing component (II-D).

A. Data Model

It is the major goal of METIS to offer a multimedia database solution for the unified management of any type of media that can be flexibly adapted to individual application requirements. It is important that this objective is already reflected by the data model METIS applies for the internal representation and management of media.

Figure 1 illustrates the major constituents of the METIS data model and their interrelationships. In this model, all basic media (text, audio, images, etc.) are uniformly represented as so-called *single media objects*. A single media object constitutes an abstract, logical representation of actual media; concrete media files are attached to a single media object through an arbitrary number of *media instances*. Media instances carry elementary metadata about the media (such as size, encoding format, bit rate, etc.) and are connected to the actual media data via so-called *media locators*,

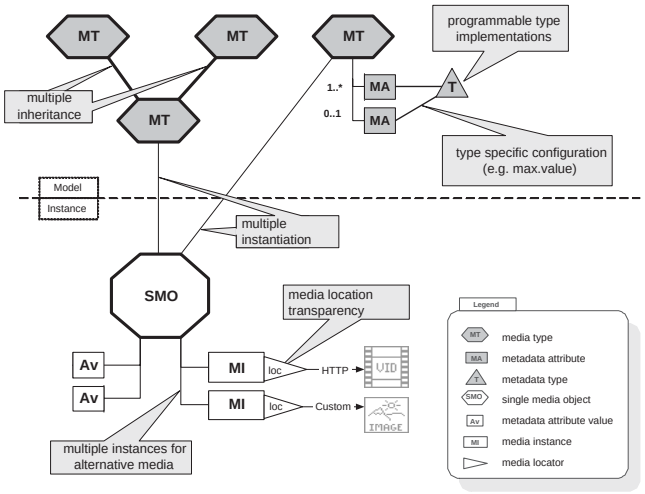

Fig. 1. The METIS core data model

allowing the system to uniformly address file systems, web servers, and databases as media sources.

This ability to gather different physical manifestations of media under a common logical media object is a very useful and flexible feature: it can in one case be exploited for load balancing, by addressing multiple instances of a given medium on alternative servers, or it can be exploited by applications that support the automatic adaptation to available network bandwidth to choose between media instances of varying quality.

In order to provide a basis for the semantic classification of media objects in METIS, these may be organized in categories, known as *media types*. As we cannot know in advance what application-specific categories will be required, the METIS data model permits the definition of any custom media types needed by an application. In the case of the abovementioned philatelic database application, for example, possible types could be “Austrian stamp”, “overprinted stamp”, or “perforated stamp”. Media types may be arranged in specialization relationships in order to be able to model taxonomies of media types. Multiple inheritance between media types is supported in order to enhance the expressiveness of the model. Multiple taxonomies may exist in parallel for the classification of media along different, possibly orthogonal, dimensions. For example, the philatelic database could provide parallel taxonomies for the country of origin, perforation, and damages of stamps.

Single media objects can be classified by assigning them an arbitrary number of media types. We are aware that this concept of multiple instantiation of media types overlaps with the concept of multiple inheritance; however, this allows the classification of media objects according to parallel taxonomies of media types, without being forced to pollute these otherwise independent taxonomies with artificial, hybrid media types.

Media types allow the classification of media objects. For the more detailed description of high-level characteristics as well as low-level features of media objects, the METIS data model includes the concept of *metadata attributes*. Metadata attributes are connected to media types in order to specify the metadata associated with the media objects of that type. Metadata attributes are freely-definable first-class objects, which means they can be shared among multiple media types, and are also propagated through the inheritance hierarchy. In this way attributes are independent of media types, and we may therefore, for example, use a single Dublin Core attribute `dc:title` (with its inherent semantics) [13] for two different domain-specific taxonomies of media types: the name of stamp in the philatelic database, and the title of a lecture in the e-learning environment. The connection of a metadata attribute with a media type may be configured with a specific cardinality, which declares how often values of the attribute (called *metadata attribute values*) must/may be defined for media objects of that type. Metadata attributes may also be restricted to a range of values and can be given default values.

Metadata attributes are typed; their values are instances of a specific *metadata type*, and METIS offers a framework that enables the integration of arbitrary type implementations by means of dynamically loaded Java classes. These implementations can provide comparison functionality, integrity checking routines, hashing functions for indexing, and the like, which are appropriate for the instances of the type. In this manner, not just standard data types like string, integer, float, etc. may be made available; it is possible to supply types of arbitrary complexity (e.g., an e-mail type, a type for color distributions of images, or even a type capturing an MPEG-7 media descriptor [14], [15]) depending on the application area, the feature extraction methods available, or the search techniques that should be employed.

Besides expressing media characteristics and features via attribute values, it is a common approach to describe media by their relationships to other media. For this purpose, the METIS data model finally offers the concept of *associations*. Associations establish

binary, directed relationships between media objects; their semantics are given by *association types*, which are freely definable for an application domain. For the sources and the targets of associations of a given type, a media type has to be provided that has to be implemented by all participating media objects in the respective role. Additionally, it is possible to provide cardinalities for these media types to specify how often objects of that type must/may participate in the association.

In summary, the combination of customizable media types, metadata attributes, and associations results in a highly expressive and flexible model for media description and classification. In addition, the separation of logical single media objects from their physical media instances, transparently addressed by media locators, gives METIS the ability to uniformly handle heterogeneous media in different storage locations. The expressiveness of its data model should allow METIS to be customized to conform to most popular media description and metadata standards in the field. The model should be capable of capturing all but the most complex RDF Schemas [16] and RDF descriptions [17] compliant to these schemas.

B. Complex Media Objects

The management of multimedia documents is not the central consideration of METIS; however, the data model includes an object type that allows a very general representation of such documents, the *complex media object*. Complex media objects are similar to single media objects in the sense that they instantiate media types, may be assigned attribute values, and may participate in associations. However, complex media objects extend single media objects in that they serve as containers of other single media objects.

For the representation of the various temporal, spatial, and interaction relationships between the single media objects in a complex media object, we have chosen not to develop yet another multimedia document model. As we aim at a unified media management, we want to be open to as many existing multimedia document format standards as possible such as MHEG [18], SMIL [19], SVG [20], or even standards yet to be defined. For this reason, we have outsourced the problem of multimedia document modeling to a simple but flexible template mechanism, schematically represented by Figure 2.

Templates are first-class objects in the system and may be shared between multiple complex media objects. A template consists of an XML document in a chosen multimedia document format, enriched by *placeholders* for the media content that is supplied by the complex media objects using the template. For example, as indicated in the figure, an author of a SMIL-based template has introduced placeholders for a logo and a large picture to be assigned by the creators of any complex media objects using that template. When the presentation of a complex media object is requested, a format-specific XSLT stylesheet [21] replaces the placeholders in the template at runtime with format-compliant references to the respective media objects connected to these placeholders.

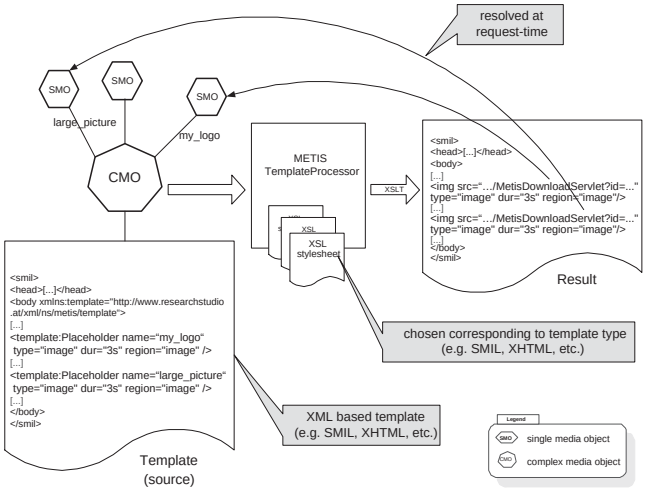

Fig. 2. Complex Media Objects

We are well aware that this simple approach to the representation of multimedia documents has its limitations. For instance, the METIS data model merely makes explicit the basic media objects of which a complex media object consists. All other details, such as the temporal synchronization and the spatial layout of these objects, are implicitly hidden in the template. As a result, METIS has difficulties in supporting the querying of complex media objects along such presentation aspects. However, we believe that a semantics-oriented, content-based querying of complex media objects should be more relevant in practice; in this regard, the METIS data model with its media types, metadata attributes, and associations offers rich facilities for the description of complex media objects.

Despite its limitations, the template-oriented representation of multimedia documents has noteworthy benefits. First, it reflects real world media production workflows, in which design and content responsibilities are separated between media designers and

editors. Second, changes to a template are immediately visible in the presentation of all complex media objects using that template. Finally, as a complex media object can be connected to more than one template, it is possible to define the content of a complex media object independent of presentation style and format, and then choose the template best suiting the needs and restrictions of the presentation channel at runtime.

C. Functions and Operators

In the previous subsections we presented the rich means by which the METIS data model offers for media representation and description. In the following, we describe how the functionality for processing and working with these data is provided in METIS, through operators for metadata attribute value comparison, functions for computing similarity between single media objects, and operators for sorting and ordering.

As this kind of functionality is usually application-specific (e.g., the philatelic database application will certainly apply different similarity measures for retrieving similar images than a passport photo database application that wants to provide face-recognition support) and as we provide a generic and highly customizable data model, it is not sufficient to simply provide a static library of predefined functions and operators. Instead, METIS again follows the ideal of flexibility and customizability and provides a framework supporting the inclusion of arbitrary new functions and operators.

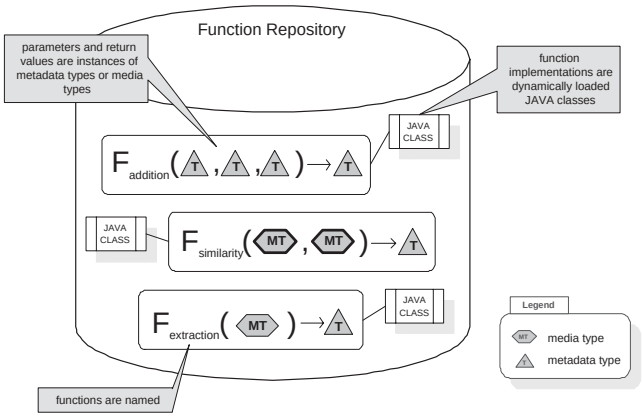

Fig. 3. The METIS function repository

An overview of this *function framework* is given by Figure 3. The *functions* in the framework have a *name*, may accept an arbitrary number of *arguments*, and deliver a *return value*. Function arguments and return values are typed, either by media types or metadata types. This means that functions can be passed and process media objects, both single and complex, as well as instances of metadata types, such as values of metadata attributes. Similarly, the result of a function call may be a media object or metadata type instance.

Functions are implemented by dynamically loaded Java classes which conform to a certain interface imposed by METIS. These *function implementations* have full access to all METIS resources including the data model which makes them very flexible and powerful constructs. For example, a function could process a single media object passed as an argument and automatically classify this single media object by assigning it to a media type.

Functions are managed by the framework's *function repository*. The repository offers several methods for the querying of the functions available with the system. For instance, a list of all functions with a given name, arguments of certain types, or a given return type may be requested. The function repository further allows the registration and the unregistration of functions and their implementations.

The functions registered with the repository can be used for several purposes. On the one hand, they can be invoked within queries of the query processor (see below) in order to compare metadata attribute values attached to media objects, compute similarities between media objects, etc. On the other hand, functions can be used to automatically extract and compute values of metadata attributes of media objects. In this case, metadata attributes may be configured to obtain their values from the call of a certain function which takes a media object as its single argument. Whenever a media object with such a metadata attribute is modified, METIS automatically invokes the function, passes the media object, and sets the value of the attribute to the function's result.

D. Query Processor

The METIS core finally provides a processor for the querying and retrieval of media objects. The query processor aims at permitting a hybrid search [8] for single and complex media objects, taking into account the semantic classification of media, their high-level characteristics and low-level features, and their relationships to other media objects.

For this purpose, the query processor offers a simple, although currently rudimentary query language, which permits the selection of all media objects in the system that fulfill a given *condition*. A condi-

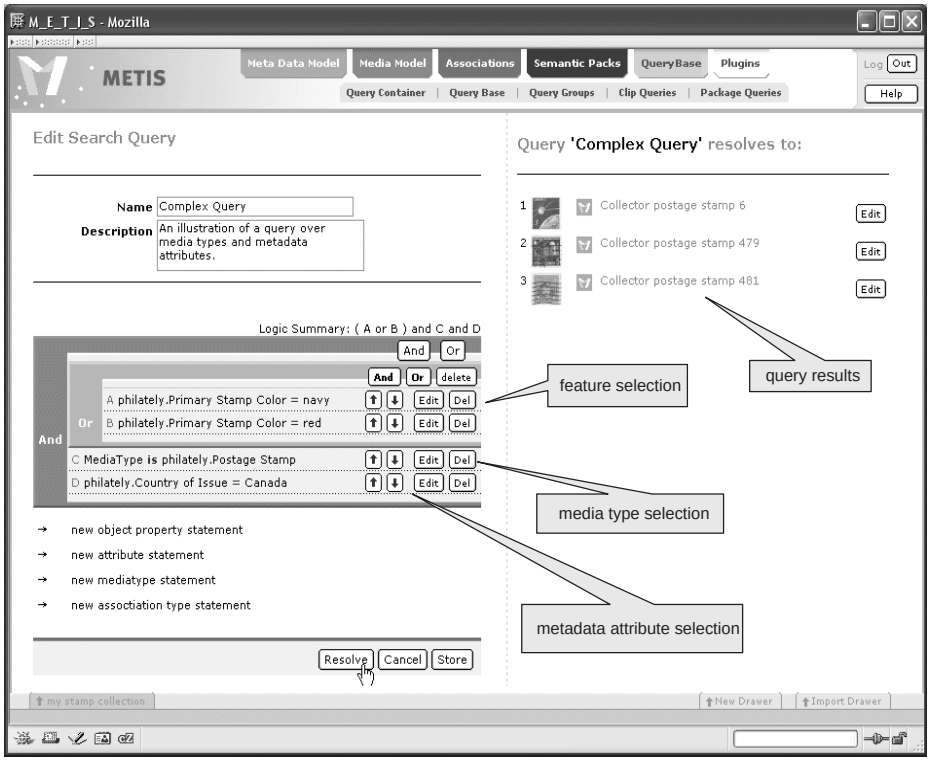

Fig. 4. The METIS query processor

tion consists of arbitrarily nested conjunctions and disjunctions of *selection predicates*. These predicates allow the selection of media objects according to their kind (i.e., single or complex), the media types they instantiate (either directly or indirectly via media type inheritance), the values of their metadata attributes (using appropriate comparison functions registered with the function repository), and their participation in associations.

Figure 4 is intended to communicate a better impression of the query language and its selection predicates by means of a screenshot of the query user interface of the METIS web front-end. The left of the query interface consists of a graphical tool for query formulation, which supports the creation of conditions with arbitrarily nested selection predicates; the right part of the query interface displays the query results. The depicted query selects media objects from the philatelic image database according to their classification by media types and the values of their metadata attributes, which comprise manually annotated high-level characteristics as well as automatically extracted low-level features. In particular, the query selects all media objects of type “Postage Stamp” with their country of issue being “Canada”, whose primary stamp color is either “navy” or “red”. It is important to note that all comparison operators for metadata attributes used in the query – i.e., the string comparison used to check the country of issue just as well as the feature comparison operator used for checking primary stamp colors – are implemented as functions registered with the function framework.

Admittedly, the expressiveness of the query language in its current state is limited. For instance, the language does not yet support joins between media objects. In many cases, however, the high degree of flexibility and extensibility of METIS provides methods to the circumvent these limitations: missing querying functionality can in many cases be outsourced and implemented as functions utilizing the function framework.

Finally, we note that queries are persistent objects and can be stored in METIS for repetitive use.

III. Persistence Abstraction Layer

The METIS system design is fundamentally object-oriented, and in the initial prototypes, used an object database system as the persistence mechanism. However, we observe a strong bias against object database systems among potential commercial project partners.

This bias is rarely technical, but is rather based on a confidence in the well-established relational database technology, which is in most cases already in place.

In order to meet the requirements of such partners, and to maintain the spirit of our flexibility paradigm, METIS was re-implemented to include a *persistence abstraction layer*. The storage interface was sufficiently abstracted so that all database specific features reside in a single class, which implements a specific *database manager interface*.

All persistent METIS objects extend the common *persistent object base class*. This class demands the implementation of store, update, and delete function for each of its subclasses. These methods do not directly access the storage layer, but rather an abstract *database manager interface*, which in addition manages transactions, commits and rollbacks. In this manner, persistence is abstracted from the concrete storage back-end. For each supported storage back-end, one must implement a single, corresponding database manager class.

Currently, we have implemented a simple file system-based variant that serializes the METIS objects to XML files. A second implementation exists for the PostgreSQL database system; this reference implementation is essentially SQL92 compliant and thus it is easily adapted to other open source or commercial relational database systems. By altering a single configuration property, we can thus switch between a low-cost system to a large-scale back-end suitable for commercial use.

For performance reason, the persistent object class implements a simple write-through cache in main memory. All persistent objects (excluding the raw media data, of course) are cached in this way; more advanced caching mechanisms are under consideration.

As a final feature, the persistence abstraction layer provides a framework for the integration of new media locators into METIS, allowing new storage locations for media to be addressed from media instances. This *media locator framework* defines interfaces for different kinds of storage locations (such as interfaces for storage locations supporting read-only, read-write, or random access), which must be implemented accordingly by Java classes for the storage locations to be supported. This framework enables METIS to seamlessly reference media stored, for example, in legacy database systems. When media data is accessed via a media instance at runtime, METIS transparently loads and instantiates the appropriate locator implementation.

IV. Visualization Layer

METIS is packaged with its own web-based administration front-end. As our goal is not only flexibility in the back-end (as described above) but also in the visualization layer, we decided to realize the METIS front-end using a highly configurable *rendering pipeline*, based on a series of XSLT transformations. This enables us to easily adapt the user interface at many levels to the application requirements, from purely graphical requirements, such as corporate identity, to functional requirements, such as user-dependent interface behavior. Additionally, we want to provide the possibility of a interface skinning mechanism that enables a fast implementation of multi-lingual user interfaces.

We chose to make use of the Apache Cocoon [22] framework for the implementation of our rendering pipeline. With its concept of eXtensible Server Pages (XSP), Cocoon allows us to enforce a clean separation of content, style and logic; with its concept of sitemaps, Cocoon easily allows us to integrate METIS with legacy web-applications. However, within Cocoon, we chose not to use XSLT in the traditional manner, which involves defining an abstract XML user interface language, with abstract layout components that are incrementally replaced by content and style tags in the desired presentation format, typically HTML. In our experience, elaborate graphical design concepts, compatible with different browsers, require highly complex HTML code, which in turn necessitates a complex XSLT pipeline.

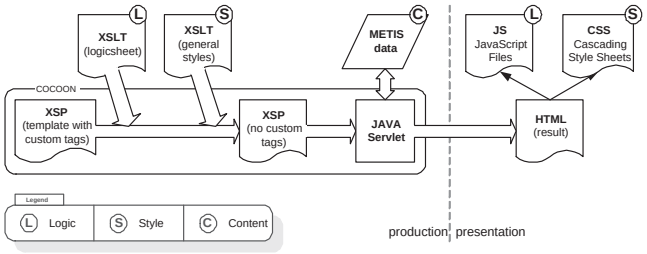

Fig. 5. The METIS rendering pipeline (simplified)

Instead, we derived our general visualization pipeline from the typical workflow for creating a web front-end. After a graphical user interface concept is defined, a graphic artist creates screen-designs, from which XHTML templates are created. These templates

are enriched by *logical tags*, representing application data (such as lists of METIS objects) and *visual tags*, representing graphical widgets (such as buttons). As indicated in Figure 5, in our general rendering pipeline, XSLT stylesheets (called logicsheets by Cocoon) incrementally replace these tags by Java logic and HTML code for rendering the graphical widgets, resulting in an eXtensible Server Page. This XSP page (similar to Java Server Pages) is then processed by Cocoon and the resulting Java servlet serves the final HTML code.

Thus, the METIS visualization pipeline enables highly flexible modification of the user interface - through modified templates, allowing a complete restructuring of the interface, through modified logic sheets, introducing changes to the interface logic, and through modified XSLT stylesheets, enabling a new look-and-feel for graphical widgets in the interface.

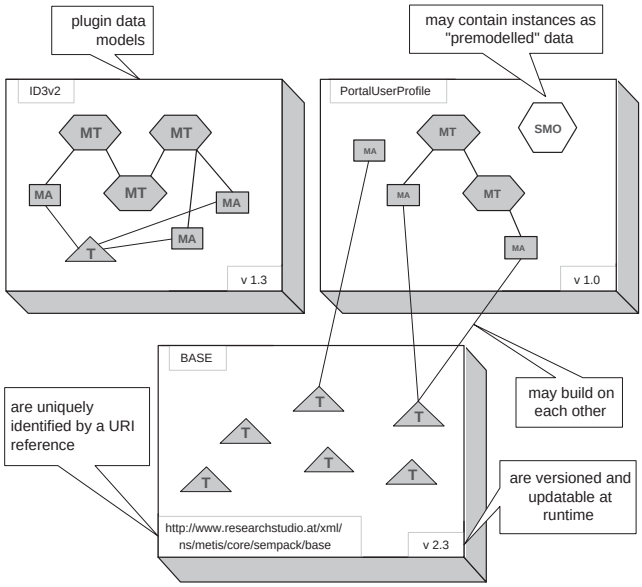

Fig. 6. METIS Semantic Packs

V. Semantic Packs

The generic nature and customizability of METIS results in many advantages, but also introduces two principle difficulties. First, as METIS supports a generic internal data model, the declared types, attributes or objects have no defined semantics. An application that wants to use the data model stored in METIS can neither interpret metadata attributes or media types nor distinguish between attributes and types from different application domains that happen to share the same name. A similar problem exists in XML, wherein tags with the same name yet completely different meanings may be found in a single document.

The second problem is that the administration of all the configurable elements offered at the various layers of METIS, from the data model to persistence to visualization, is extremely complex. A method of grouping these elements for specific application domains is required.

In METIS we approach these difficulties by introducing the concept of a *semantic pack*, which is illustrated by Figure 6. Semantic packs establish spaces of semantically related METIS objects, analogously to XML namespaces. They are identified by a unique URI reference. All object names in METIS are prefixed by the URI of the semantic pack to which they belong, thus solving the problem of unique identification.

Unlike XML namespaces, semantic packs do not only establish a logical space; they are also physical containers that include the definitions of the METIS objects in the space, possibly including metadata attributes, functions, media type hierarchies, and associations, as well as dynamically loaded Java class files such as basic data type implementations, media locators, feature extractors, and even templates for the web front-end. Using this mechanism, one can pre-package and deliver application- and/or domain-specific customizations of METIS, thereby reducing the aforementioned administration complexity.

Semantic packs can be hierarchically organized: a semantic pack can make use of objects from another semantic pack. For example, most semantic packs would make use of the METIS base semantic pack, which contains the implementations of the primitive metadata types, such as string, integer or date, required by almost any application. Versioning of semantic packs is also supported, that is, they can be extended and dynamically updated, as long as backwards compatibility is maintained.

Physically, semantic packs are Java archives (JAR), which makes them easy to transport, install, and administer. As an additional positive side-effect, these JAR archives can be signed, thus authenticating the origin of a semantic pack and preventing the integration of malicious code.

These characteristics establish semantic packs as an essential foundation for a modularized development. Possibly distributed parties can contribute their expertise and develop semantic packs with METIS

customizations for their domain. As a further benefit, it is possible to model existing metadata standards with METIS and bundle them as semantic packs; thereby, standard conformity can be established by merely installing a semantic pack.

VI. Open Issues

We have implemented the METIS architecture as described in the previous sections. As a result, we now possess a very flexible multimedia database system for the unified management of media of different types; however, there are still several areas requiring further development.

One essential aspect of the system that clearly requires more development is the query framework. We are now developing an indexing framework which permits the integration of arbitrary unordered, ordered, and multidimensional index structures into METIS for those applications that require more efficient querying of media. Comparable to index extension interfaces of object-relational database systems (e.g., [23]), such an indexing framework will even further increase the flexibility and customizability of METIS. We also intend to extend the limited query language of METIS towards relevant multimedia querying standards on the horizon, such as SQL/MM [24].

Furthermore, it is our intention to make METIS usable as a database foundation for the whole multimedia production cycle, in which people continuously and collaboratively work on media. As this requires the ability to manage and keep track of different versions of the same media and its metadata, we are working on integrating versioning support for media objects into METIS. In such a scenario with multiple users with different responsibilities accessing a common database, security becomes a pressing issue as well (not to mention the use of and access to media material that is frequently restricted by copyright regulations). Therefore, we additionally plan to provide METIS with a configurable user management and access control allowing the system to adapt to the security needs of an application.

Finally, we are striving to further optimize the persistence abstraction layer by evaluating different caching techniques. We are also investigating how different METIS instances running on distributed nodes can be integrated on the basis of a peer-to-peer network.

VII. Related Work

The essential idea underlying the architecture of METIS is to provide a multimedia database platform for the unified management of media of arbitrary types that can be flexibly tailored to the individual needs of an application. By comparing METIS to other approaches in the field, we show that a multimedia database system with such a flexibility and customizability is unprecedented.

Today, we find a broad spectrum of media type-specific database system prototypes, including video databases (like the well-known OVID [1], Stratification [25], and VideoQ [26] systems or more recent ones like SMOOTH [2] or IFINDER [3]), image databases (such as the classic Vimsys [27] and QBIC [4] systems or newer ones like MARS [5], KMeD [11], DISIMA [6], or IMKA [28]), and (though considerably fewer in number) audio databases [29], [30]; but, as we have already mentioned, far less effort has been dedicated to truly *multi*media database systems like METIS that address the unified, integrated management of media of more than one type.

Among these efforts, a large group consists of multimedia extensions for traditional object-oriented or object-relational database systems, such as Jasmine [31], or the multimedia extensions for the Oracle and IBM DB2 database servers [32], [33]. However, these extensions mostly constitute libraries of rather isolated data types for the handling of different types of media in a traditional database system, already making it difficult to speak of unified media management.

Another group of systems (such as AMOS [34], the ZYX-Datablade [35], ViSiOn [36], and MediaManager [37]) provides a more integrated treatment of media of different types in a database but, unlike METIS, emphasizes the modeling of multimedia documents based on these media and not classic media management tasks such as media description, search, and retrieval.

Very few systems actually provide a unified media management that is comparable to METIS: the Media Integration Blade [38], which established a unified media management layer on top of the multimedia extensions for the Informix Dynamic Server, can be regarded as a precursor of METIS. But the Media Integration Blade suffers from a limited and to a large extent hardwired data model that makes it difficult to adapt to specific application needs.

The MediaLand system [8], very similar to METIS,

offers a flexible data model that permits the definition arbitrary, logical media type hierarchies, the definition of arbitrary attributes and associations for media of these types. METIS, however, with its concept of semantic packs, with its ability to integrate arbitrary data types for the values of metadata attributes, query operators, and feature extractors, with its persistence abstraction layer allowing one to change the storage back-end, as well as with its read-to-use web front-end that can be quickly customized to application needs, takes the ideal of flexibility to a further level.

VIII. Conclusion

In this paper, we have given an overview of METIS, a general-purpose database foundation for the unified management of multimedia content that is highly customizable to application- and domain-specific needs. We have described the data model METIS applies for the uniform management of media of different types (including multimedia documents), which allows the definition of any custom media types, metadata attributes, metadata types, and association types desired by an application for media classification and description. We have discussed the function framework – which not only permits the integration of arbitrary functions and operators for the querying of media objects and their descriptions, but also the integration of functionality for the automatic derivation and extraction of metadata attribute values – and sketched the query processor of METIS. We have illustrated the persistence abstraction layer, which allows us to react to individual storage requirements by changing the storage back-end of METIS with relatively little effort, and supports the integration of locators for transparent access to media in arbitrary storage locations. We have outlined the system's highly customizable visualization layer, and introduced the concept of semantic packs as transferable and easily deployable bundles of domain-specific customizations from all levels of METIS. We have given an overview of open issues upon which we are directing our current and future work, and emphasized the novelty of the METIS approach through an extensive comparison with prominent multimedia database systems existing in the field.

Currently, we are applying METIS in several projects. Apart from a philatelic image database serving demonstration purposes, which already appeared in the examples of this paper, we are using METIS to develop a news video database, which integrates a commercial audio logger with METIS for the automatic classification of news videos, and an MP3 music jukebox, which integrates music processing algorithms for automatic song classification and similarity search. The initial experience we have gained from these projects indicates that the METIS approach to providing a generic, highly customizable multimedia database solution as a common foundation for a wide variety of applications is very promising.

References

[1] E. Oomoto and K. Tanaka, "OVID: Design and Implementation of a Video-Object Database System," *IEEE Transactions on Knowledge and Data Engineering*, vol. 5, no. 4, 1993.

[2] H. Kosch, R. Tusch, L. Böszörmenyi, et al., "SMOOTH – a Distributed Multimedia Database System," in *Proc. of the 27th International Conference on Very Large Data Bases (VLDB 2001)*, Rome, Italy, Sept. 2001.

[3] J. Löffler, K. Biatov, C. Eckes, and J. Köhler, "IFINDER:an MPEG-7-based retrieval system for distributed multimedia content," in *Proc. of the 10th ACM International Conference on Multimedia*, Juan-les-Pins, France, Dec. 2002.

[4] M. Flickner, H. Sawhney, W. Niblack, et al., "Query by Image and Video Content: the QBIC System," *IEEE Computer*, vol. 28, no. 9, 1995.

[5] M. Ortega, K. Chakrabarti, and S. Mehrotra, "Supporting Ranked Boolean Similarity Queries in MARS," *IEEE Transactions on Knowledge and Data Engineering*, vol. 10, no. 6, 1998.

[6] V. Oria, M.T. Özsu, P. Iglinski, et al., "DISIMA: An Object-Oriented Approach to Developing an Image Database System," in *Proc. of the 16th International Conference on Data Engineering*, San Diego, California, Mar. 2000.

[7] R. Jain, P. Kim, and Z. Li, "Experiential Meeting System," in *Proc. of the ACM SIGMM Workshop on Experiential Telepresence (ETP 2003)*, Berkeley, California, Nov. 2003.

[8] J.-R. Wen, Q. Li, W.-Y. Ma, and H.-J. Zhang, "A Multi-Paradigm Querying Approach for a Generic Multimedia Database Management System," *ACM SIGMOD Record*, vol. 32, no. 1, 2003.

[9] H.D. Wactlar, A.G. Hauptmann, M.G. Christel, et al., "Complementary Video and Audio Analysis for Broadcast News Archives," *Communications of the ACM*, vol. 43, no. 2, 2000.

[10] H. Yang, L. Chaisorn, Y. Zhao, et al., "VideoQA:Question Answering on News Video," in *Proc. of the 11th ACM International Conference on Multimedia*, Berkeley, California, Nov. 2003.

[11] W.W. Chu, C.-C. Hsu, A.F. Cárdenas, and R.K. Taira, "Knowledge-Based Image Retrieval with Spatial and Temporal Constructs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 10, no. 6, 1998.

[12] D. Power, E. Politou, M. Slaymaker, et al., "A Relational Approach to the Capture of DICOM Files for Grid-Enabled Medical Imaging Databases," in *Proc. of the 2004 ACM Symposium on Applied Computing*, Nicosia, Cyprus, Mar. 2004.

[13] DCMI, "Dublin Core Metadata Element Set," DCMI Recommendation Version 1.1, Dublin Core Metadata Initiative (DCMI), July 1999.

[14] J.M. Martinez, R. Koenen, and F. Pereira, "MPEG-7 – The Generic Multimedia Content Description Standard, Part 1," *IEEE MultiMedia*, vol. 9, no. 2, 2002.

[15] J.M. Martinez, "MPEG-7 – Overview of MPEG-7 Description Tools, Part 2," *IEEE MultiMedia*, vol. 9, no. 3, 2002.

[16] D. Brickley and R.V. Guha, "RDF Vocabulary Description Language 1.0: RDF Schema," W3C Working Draft, World Wide Web Consortium (W3C), Jan. 2003.

[17] O. Lassila and R.R. Swick, "Resource Description Framework (RDF) Model and Syntax Specification," W3C Recommendation, World Wide Web Consortium (W3C), Feb. 1999.

[18] ISO/IEC IS 13522-5, "Information Technology – Coding of Hypermedia Information - Part 5: support for Base-Level Interactive Applicatons," ISO/IEC International Standard, International Organization for Standardization/International Electrotechnical Commission (ISO/IEC), 1996.

[19] J. Ayars, D. Bulterman, A. Cohen, et al., "Synchronized Multimedia Integration Language (SMIL 2.0)," W3C Recommendation, World Wide Web Consortium (W3C), Aug. 2001.

[20] J. Ferraiolo, J. Fujisawa, and D. Jackson, "Scalable Vector Graphics (SVG) 1.1 Specification," W3C Recommendation, World Wide Web Consortium (W3C), Jan. 2003.

[21] J. Clark, "XSL Transformations (XSLT)," W3C Recommendation, World Wide Web Consortium (W3C), Nov. 1999.

[22] Apache Software Foundation, "Cocoon," http://cocoon.apache.org, 2003.

[23] IBM Corp., "Virtual Index Interface Programmer's Manual," System Documentation Version 9.3, IBM Corp., Aug. 2001.

[24] J. Melton and A. Eisenberg, "SQL Multimedia and Application Packages (SQL/MM)," *ACM SIGMOD Record*, vol. 30, no. 4, 2001.

[25] T.G.A. Smith and G. Davenport, "The Stratification System - A Design Environment for Random Access," in *Proc. of the 3rd International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV 1992)*, La Jolla, California, Nov. 1992.

[26] S.-F. Chang, W. Chen, H. Meng, et al., "VideoQ: an Automated Content-Based Video Search System Using Visual Cues," in *Proc. of the 5th ACM International Conference on Multimedia 1997*, Seattle, Washington, Nov. 1997.

[27] A. Gupta, T. Weymouth, and R. Jain, "Semantic Queries with Pictures: The VIMSYS Model," in *Proc. of the 17th International Conference on Very Large Data Bases*, Barcelona, Spain, Sept. 1991.

[28] B. Benitez, S.-F. Chang, and J.R. Smith, "IMKA: a Multimedia Organization System Combining Perceptual and Semantic Knowledge," in *Proc. of the ACM International Conference on Multimedia 2001*, Ottawa, Canada, Sept. 2001.

[29] A. Ghias, J. Logan, D. Chamberlain, and B.C. Smith, "Query by Humming – Musical Information Retrieval in an Audio Database," in *Proc. of the 3rd ACM International Conference on Multimedia 1995*, San Francisco, California, Nov. 1995.

[30] H. Crysandt and J. Wellhausen, "Music Classification with MPEG-7," in *SPIE Proceedings Storage and Retrieval for Media Databases*, Santa Clara, California, Jan. 2003.

[31] S. Koshafian, S. Dasananda, and N. Minaasian, *The Jasmine Object Database: Multimedia Applications for the Web*, Morgan Kaufmann Publishers, San Francisco, California, 1998.

[32] R. Ward and others, "Oracle interMedia User's Guide," Oracle 10g System Documentation Release 1 (10.1), Oracle Corp., Dec. 2003.

[33] IBM Corp., "Image, Audio, and Video Extenders – Administration and Programming," DB2 Universal Database System Documentation Version 8, IBM Corp., June 2003.

[34] S. Boll, W. Klas, and M. Löhr, "Integrated Database Services for Multimedia Presenations," in *Multimedia Information Storage and Management*, S.M. Chung, Ed. Kluwer Academic Publishers, Boston, Masschussetts, 1996.

[35] S. Boll, W. Klas, and U. Westermann, "Exploiting ORDBMS Technology to Implement the ZYX Data Model for Multimedia Documents and Presentations," in *Proc. of the 8th GI-Fachtagung Datenbanksysteme in Büro, Technik und Wissenschaft (BTW 1999)*, Freiburg, Germany, Mar. 1999.

[36] T. Lee, L. Sheng, N.H. Balkir, A. Al-Hamdani, et al., "Query Processing Techniques for Multimedia Presentations," *Multimedia Tools and Applications*, vol. 11, no. 1, 2000.

[37] S.-C. Chen, M.-L. Shyu, and N. Zhao, "MediaManager: A Distributed Multimedia Management System for Content-Based Retrieval, Authoring and Presentation," in *Proc. of the 9th International Conference on Distributed Multimedia Systems (DMS 2003)*, Miami, Florida, Sept. 2003.

[38] U. Westermann and W. Klas, "Architecture of a DataBlade Module for the Integrated Management of Multimedia Assets," in *Proc. of the First International Workshop on Multimedia Intelligent Storage and Retrieval Management (MISRM '99)*, Orlando, Florida, Oct. 1999.