John R. Smith
IBM

The MPEG Query Format: Unifying Access to Multimedia Retrieval Systems

Mario Döller
University of Passau

Ruben Tous
Universitat Polit'ecnica de Catalunya

Matthias Gruhne
Fraunhofer IDMT

Kyoungro Yoon
Konkuk University

Masanori Sano
Science & Technical Research Laboratories, NHK

Ian S. Burnett
RMIT University

Recent studies on multimedia devices report that in 2009 shipments of multimedia-enabled mobile phones will outnumber shipments of TV sets (see http://www.multimediaintelligence.com/index.php?option=com_content&view=article&id=78:-multimedia-mobile-phone-shipments-surpass-tv-shipments-in-2008-according-to-multimedia-intelligence&catid=37:frontpagetitleonly&Itemid=142). Trends toward portability, personalization, and interactivity can be observed across a broad range of market segments, representing a significant increase in the production of multimedia content. This increased availability and use of multimedia content highlights the need for efficient multimedia data storage and retrieval solutions.

Currently, there are numerous storage and retrieval approaches ranging from commercial products, such as Oracle Multimedia (see http://www.oracle.com/technology/products/intermedia/index.html), to the purely academic.[1,2] The diversity of these individual projects prevents users from experiencing broad and unified access to different multimedia collections. In fact, almost every solution provides a different retrieval interface and is based on different multimedia metadata description formats, such as MPEG-7 (see http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm), Metadata Encoding and Transmission Standard (see http://www.loc.gov/standards/mets/), and so on. An example of the technology diversity in multimedia storage and retrieval is the broad range of multimedia query languages used across these projects, the more well known approaches being XQuery and its numerous extensions for multimedia,[3] SQL/MM, MMDOC-QL,[4] and so on. Combining these varied query approaches with alternate metadata description formats and retrieval interfaces prevents effective interoperability among current multimedia retrieval systems.

The goal of the MPEG Query Format (MPQF) is to facilitate and unify access to distributed multimedia repositories. To achieve this goal, the MPQF standard specifies precise input and output parameters to express multimedia requests and to allow clients easy interpretation and processing of result sets. Moreover, the management component of the MPQF covers searching and the choice of the desired multimedia services for retrieval. For this purpose, the standard provides a means to describe service capabilities and to undertake service discovery.

Throughout this article, we use the term *multimedia service* as a substitute for any multimedia database, multimedia repository, or multimedia retrieval system that supports the MPQF. In turn, a multimedia service can denote a single multimedia repository or an aggregated service providing access to a multimedia repository set. It is also important to note that while the MPQF will become part 12 of the MPEG-7 standard, the specification explicitly allows for deployments using any XML-based multimedia metadata description format.

Editor's Note

The growth of multimedia is increasing the need for standards for accessing and searching distributed repositories. The Moving Picture Experts Group (MPEG) is developing the MPEG Query Format (MPQF) to standardize this interface as part of MPEG-7. The objective is to make multimedia access and search easier and interoperable across search engines and repositories. This article describes the MPQF and highlights some of the ways it goes beyond today's query languages by providing capabilities for multimedia query-by-example and spatiotemporal queries.

—John R. Smith

1070-986X/08/$25.00 © 2008 IEEE

Published by the IEEE Computer Society

Why a new query language?

To satisfactorily explain the need for a new query language it's necessary to take a detailed look at currently available interfaces for multimedia retrieval. The current landscape of query languages especially designed for multimedia applications includes many approaches. In the literature, there are, for example, extensions of SQL (for example, SQL/MM) and Object Query Language (for example, Pcte-Object-Query-Language, or POQL^{MM5}), languages bound to a specific metadata model (for example, MMDOC-QL), languages concentrating on temporal or timeline retrieval (for example, MQuery[6]), or languages that integrate weighting capabilities for expressing user preferences (for example, Weighted-Similarity-Query-by-Example, or WS-QBE[7]).

Most of these multimedia query languages use proprietary metadata models to express descriptive information. Generally, this information is represented by XML instance documents based on a specific XML schema—such as TV-Anytime (see http://www.tv-anytime.org/) or MPEG-7. In these cases, we need to consider query languages designed for XML data; these include solutions such as Information Retrieval Query Language (XIRQL),[8] XML Query Language (XQL),[9] or Tree Query Language (TQL),[10] but the most well-known approach is XQuery.[11] The main drawback of XML is its limitations in expressing the content's semantic meaning. This drawback led to the development of the Resource Description Framework (see http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/). In contrast to XML, which provides a syntax to encode data, RDF supports a mechanism to express the semantics and relationships between and about data. Recent work uses RDF for representing, reconciling, and semantically tagging multimedia resources to improve multimedia retrieval by semantic means.[12] This development created the need for domain query languages; representative solutions include XsRQL (see http://www.fatdog.com/xsrql.html) and the newest recommendation from the W3C, Simple Protocol And RDF Query Language (SPARQL).[13]

We can see then that a range of multimedia query languages is available, leading to questions about the reason for developing a new multimedia query language. Can we simply categorize the MPQF as yet another query language?

First, let's consider more closely the pure multimedia query languages: one common drawback of most query languages is a lack of support for weighting of query terms to reflect user preferences. Furthermore, existing languages frequently are restricted to only one media type, don't have a formal semantic, and rely on proprietary metadata descriptions that prevent interoperability or only target a subset of possible multimedia query paradigms. Also, combinations of information retrieval (for example, query-by-example) and exact retrieval on metadata (for example, finding a file size greater than 100 Mbytes and with the video's producer being ''John Doe'') are not supported.

Secondly, let's look at XML-based query languages, which are strong in expressing data-centric, exact requests (such as give me all images whose file size is greater than 100 Mbytes) but lack the ability to express fuzzy requests common in multimedia retrieval (such as query-by-example). There already are some approaches aimed at extending XQuery in this direction.[3] However, these approaches are restricted to the XQuery data model, which doesn't contain semantics for processing multimedia objects. Therefore, an XML-based query language alone is inadequate for multimedia retrieval.

Finally, let's consider SPARQL as representative of RDF query languages; it's strong in semantic search and suited for use in the Semantic Web environment. Comparing SPARQL to the MPQF reveals several similarities. SPARQL is composed of three main parts: query language, protocol, and XML results format. The SPARQL protocol provides a means for conveying SPARQL requests from clients to query processors via HTTP or SOAP communication and specifies how fault messages can be expressed. Similarly, the MPQF introduces management tools that aid in searching for and choosing desired multimedia services. These management tools include service discovery, querying for service capability, and service capability description. In contrast to SPARQL, the MPQF doesn't specify and extend a specific binding (for example, HTTP) but is, of course, aware of being transported within a SOAP request. Furthermore, the MPQF supports a more comprehensive set of error messages defined through classification schemes.

The SPARQL XML results format specifies how a possible result set can be encoded and returned as an XML document. In the case of the MPQF, the Output Query Format specifies the same functionality. However, the OQF also deals more precisely with the requirements of multimedia retrieval as it provides ranking and confidence information for every result item, including paging functionality and support for relevance feedback operations.

The most important section and differentiation of a query language is its expressiveness in formulating search requests. In the case of SPARQL, formulating search requests is specified in the query language document. SPARQL is a syntactically-SQL-like language for querying RDF graphs via pattern matching (by specifying a triple pattern). The queries themselves look and act like RDF. The language's features include basic conjunctive patterns, value filters, optional patterns, and pattern disjunction. SPARQL's main strengths include its simplicity in formulating requests, the implicit join handling within the syntax, and the semantic expressiveness related to RDF. Nevertheless, like XQuery, SPARQL is not designed for searching within multimedia databases and doesn't provide the unique functionalities desirable for multimedia queries. Such requirements include, for instance, content-based queries (such as query-by-example), fuzzy queries, or spatiotemporal queries for audiovisual data.

It's clear that there is the need for a standardized multimedia query language. If we consider the well-known quotation from Tim Berners-Lee, ''SPARQL is to the Semantic Web (and, really, the Web in general) what SQL is to relational databases,'' our ambitious aim is that the MPQF will be to multimedia services what SQL is to relational databases.

MPQF concepts and benefits

Essentially, the MPQF is an XML-based query language that defines the query and reply formats exchanged between clients and servers in a distributed multimedia search-and-retrieval system. The two main benefits of standardization of such a language are interoperability between parties in a distributed scenario (for example, content providers, aggregators, and clients) and platform independence (which also offers benefits for non-distributed scenarios). The result is that developers can construct applications exploiting multimedia queries independent of the multimedia service used; this capability fosters software reusability and maintainability. As a technical specification from the Moving Picture Experts Group (MPEG, see http://mpeg.chiariglione.org), this initiative is an international, open standard targeting all application domains.

Information and data retrieval

One key feature of the MPQF is that it allows the expression of multimedia queries combining the expressive style of information and XML data-retrieval systems. Thus, the MPQF combines, for example, keywords and query-by-example with, for example, XQuery, allowing the fulfillment of a broad range of users' multimedia information requirements. Both approaches to data retrieval are designed to facilitate clients' access to information, but from different points of view. Given a query expressed in a user-oriented manner (for example, an image of a bird with blue wingtips), an information-retrieval system is designed to retrieve information that might be relevant even though the query is not formalized. In contrast, a data-retrieval system (for example, an XQuery-based database) deals with a well-defined data model and is designed to determine which objects of the collection satisfy clearly defined conditions in a relational algebra expression (for example, a movie's title, a video's file size, or an audio signal's fundamental frequency). For a data-retrieval system, like a relational database or an XML repository, a single erroneous object among thousands retrieved means a total failure. In contrast, an information-retrieval system can supply results that appear relevant and expect a user to filter those results further.

With regard to information retrieval, the MPQF offers many possibilities that include but are not limited to query-by-example-media, query-by-example-description, query-by-keywords, query-by-feature-range, query-by-spatial-relationships, query-by-temporal-relationships, and query-by-relevance-feedback. For data retrieval, the MPQF offers its own XML query algebra for expressing conditions over the multimedia-related XML metadata (for example, MPEG-7, Dublin Core, or any other XML-based metadata format) but also offers the possibility of embedding XQuery expressions.[11]

Query format overview

MPQF instances are XML documents that can be validated against the MPQF XML Schema. An MPQF instance always includes the MpegQuery element as the root element and beneath that the Query or Management element (see Figure 1). MPQF instances with the Query element are the usual requests and responses of a multimedia search process. The Query element can include the Input (Input Query Format, or IQF) or Output (OQF) element, depending on whether the document is a request or a response. A special query input constitutes the FetchResult element, which is used, for example, in asynchronous mode for collecting search results. Alternatively, below the root element, an MPQF document can include the Management element. Management messages (which can be both requests and responses) provide a means for requesting service-level functionalities, such as multimedia-services discovery or other types of service provision, interrogating the capabilities of a service or configuring service parameters.

MPQF Data Model

An MPQF query is expressed in terms of a certain information representation space. The MPQF Data Model formally defines this information representation space and constitutes the basis of an MPQF query processor and optimizer. When writing an MPQF query, we must consider that it will be evaluated—for example, by an MPQF aware interpreter or a content provider responsible for distributing the query—against one or more multimedia services Also, we must consider that, from the MPQF point of view, a multimedia service is an unordered set of multimedia content containing, for example, a reference to audiovisual data or text documents. Multimedia content, formally described elsewhere,[14] refers to the combination of multimedia data and its associated metadata (see Figure 2). The MPQF allows search and retrieval of complete or partial multimedia content data and metadata by specification of a filter condition tree and desired processing granularity. Conditions within that tree operate on one evaluation item at a given time, or two evaluation items if a Join operation is used.

By default, an evaluation item is multimedia content in the multimedia service, but other evaluation item types are possible. For instance, an evaluation item can be a segment of a multimedia resource, or an XPath item related to the multimedia content's metadata XML tree, as defined in the XQuery 1.0 and XPath 2.0 data model.[15] As mentioned previously, the MPQF allows search and retrieval of both complete and partial multimedia content. The scope of query evaluation and the granularity of the result set can be determined by an EvaluationPath element specified within the query. If this EvaluationPath element is not specified, the output result is provided as a collection of multimedia content (as stored in the repository) all satisfying the query condition.

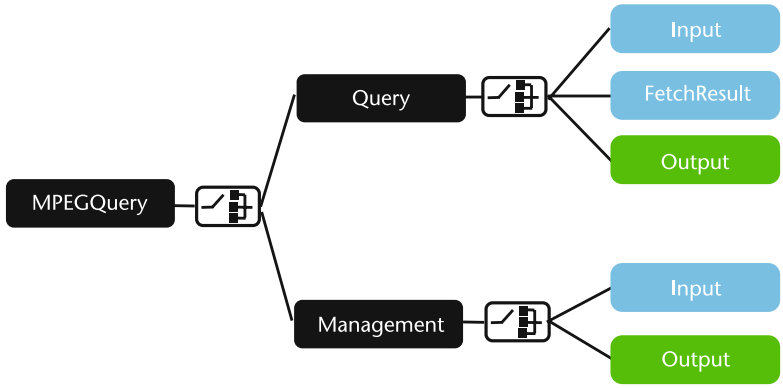

Figure 1. Schema overview of the uppermost elements of the MPEG Query Format.

Distributed multimedia retrieval scenarios

The MPQF defines a request-reply, XML-based interface between a requester and a responder. In the simplest scenario, the requester might be a client and the responder might be a multimedia-retrieval system. However, the MPQF is designed for more complex scenarios in which clients interact, for instance, with a content aggregator. The content aggregator acts at the same time as both a responder (from the client point of view) and requester to several underlying content providers to which the client query is forwarded.

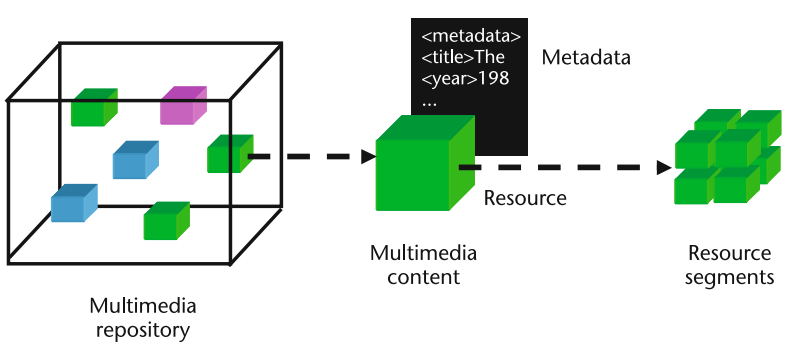

Figure 2. The MPEG Query Format Data Model: multimedia repository and multimedia content.

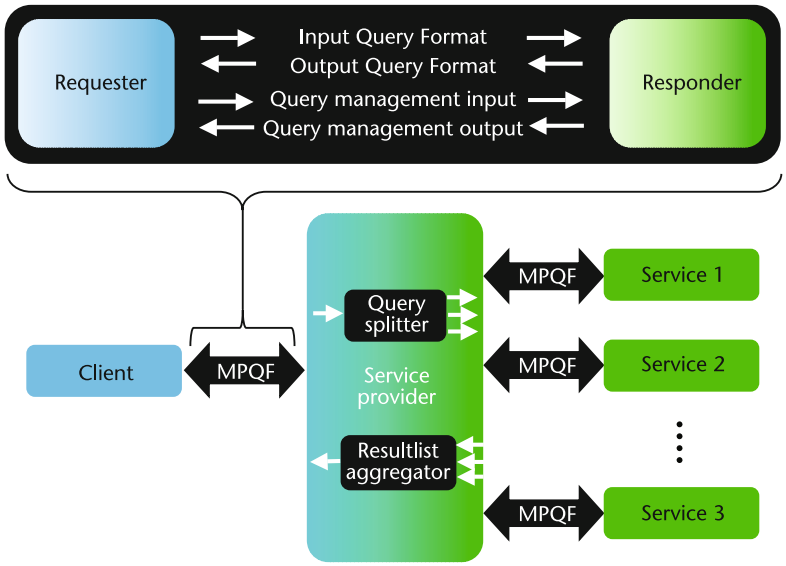

Figure 3. Possible MPEG Query Format use scenario.

Figure 3 depicts a possible setup for the use of the MPQF.

Besides the benefit that any MPQF aware client is able to interact in a standardized way with any MPQF service, the management part also supports the discovery of multimedia services in a highly distributed and heterogenous environment. For this purpose, the management part of the standard explicitly foresees service discovery functionality, which might result in registry applications, such as Corba's naming service[16] or the UDDI service[17] in Web services' service-oriented architecture. Such a registry service allows distributed multimedia services' registration (by sending their service capability description) and thus makes them available to a broader community; consequently, the service provider's retrieval system becomes much more valuable. Note that the standard only specifies how a client can interact with such a registry application; the actual process of registration and its implementation are out the MPQF standard's scope.

Another aspect of a distributed-retrieval scenario is the communication paradigm employed and whether it's synchronous or asynchronous. Whereas the synchronous mode satisfies most common multimedia retrieval scenarios, it's insufficient when the search operation requires a long processing time or the display capability of the client's device (for example, a mobile phone) is inadequate for presenting the results. For such environments, the MPQF standard supports a means for signaling the multimedia service the need for an asynchronous operation. In these cases, the service responds with a query ID and time-span information, which allows the client to fetch the result from a different place at a later time or on a different device.

Figure 3 shows the setup of a multimedia information retrieval system with different services and one service provider. The service provider contains a list of services and their functionality, and might also contain, for example, services specifically for audio identification or face recognition. The client submits a request to the service provider, which analyzes the query and then submits adapted queries to specific and relevant services. Each service, in turn, processes a specific part of the query and replies with a result list to the service provider. The service provider aggregates the result lists and creates a single result list in reply to the client. Therefore, service-provider processing is concentrated on the needs of query distribution and result-list aggregation. A description of a first service provider for cross-modal search for video identification using the MPQF can be found elsewhere.[18]

XQuery support

The MPQF allows the embedding of an XQuery 1.0 expression[11] within an MPQF filter condition tree. To be consistent with the MPQF Data Model, the embedded XQuery expression is restricted to the use of constructs that produce a Boolean true-or-false decision on a single evaluation item at the target database. Therefore, within an XQuery expression included in the filter condition tree, no output description is allowed.

Parts of the MPQF

Here we introduce the main parts of MPQF, where the input query format provides means for formulating requests, the output query format describes a standardized response, and the query management tools are about service discovery and service description.

Input Query Format

The IQF describes the contents of the Input element (see Figure 4) and defines the syntax of messages sent by the client to the multimedia service. These messages specify the client's search criteria. The IQF's two main compo-

nents allow specification of a filter-condition tree (by using the QueryCondition element) and definition of the structure and desired content of the service output (by the OutputDescription element). The IQF also allows declaration of reusable resource definitions and metadata paths (fields) within the QFDeclaration element, and the set of services where the query should be evaluated (the ServiceSelection element) in the case of communication with an aggregation service.

The optional QFDeclaration element allows declaration of reusable definitions of data paths and resources that can be referenced multiple times within a query. A data path is declared by a DeclaredField element, which contains a relative or absolute XPath expression pointing to an item of the service's metadata. A Resource element operates as container for one of the following components:

- MediaResource describes a resource by containing or pointing to the raw multimedia material.

- DescriptionResource specifies the container for any description based on a specific schema specified by the namespace declaration within the description.

The introduction of the QFDeclaration element allows for significant reduction of the query size if the same elements appear multiple times in one query.

The OutputDescription element describes the structure and content of the individual result items within the result set. Furthermore, it allows (by two optional attributes) the definition of an overall item count and the maximum number of items per output page. The occurrence of the second attribute indicates that paging of the result set is desired.

The OutputDescription's main features follow:

- ReqField describes the data path to the element, which a client asks to be returned. Paths are specified either using absolute XPath expressions, which refer to the description's root, or using relative XPath expressions, which refer to a given schema's complex type.

- ReqAggregateID describes the unique identifier of the aggregate operation the client asks to be returned. When one or more ReqAggregateIDs are used, the aggregate ID should be grouped.

- GroupBy describes the grouping operation the user wants to apply on the result set. For this purpose, the GroupBy element allows the specification of several GroupByField elements that describe the key for the grouping process and aggregation operations.

- SortBy describes the sort operation the client wants to apply on a result set. The result set can be sorted in increasing or decreasing order according to a given Field element or an aggregate expression. The Field element contains an absolute or relative XPath expression.

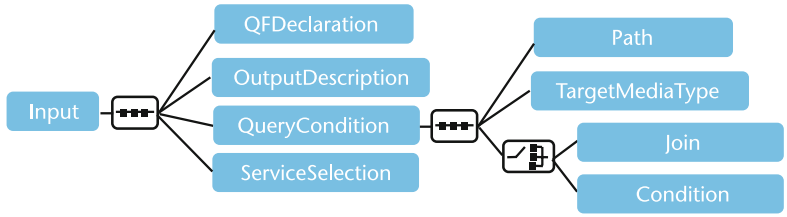

Figure 4. The Input Query Format.

The QueryCondition supports a client expressing filter criteria for a retrieval. The main entry point for formulating filter criteria is the QueryCondition element (see Figure 4), which features an optional EvaluationPath element, an unbounded number of TargetMediaType elements, and a choice between a Join and a Condition element. The EvaluationPath (an XPath expression) element specifies the granularity of the retrieval and therefore the metadata fragment node related to the evaluation item. For instance, in the case of MPEG-7, the EvaluationPath //Video would focus the retrieval to whole video objects whereas the EvaluationPath //VideoSegment would lead to the more specific video segment objects.

The TargetMediaType element contains MIME-type descriptions of media formats that are the targets for retrieval. For instance, the MIME-type audio/mp3 would filter all results for audio files depending on the MP3 format. Further, diversity in filter criteria is provided by the Condition element. The Condition element is a placeholder for a Boolean expression type (see Figure 5 on the next page, for its type hierarchy) and might result in an *n*-ary

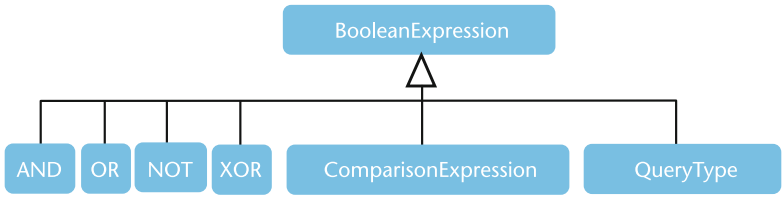

Figure 5. Boolean expression hierarchy.

filter tree. As outlined in Figure 5, the filter tree can be established by three main constructs, namely query types, comparison expressions, and Boolean operators. In general, instances of query types always represent leaf nodes within the filter tree. Nodes of type comparison expression can occur as inner nodes and leaf nodes. Nodes that represent Boolean operators are always inner nodes. To indicate the importance of individual parts during the retrieval process, we can assign a preference value to every Boolean expression element, and hence to each node within the filter tree.

In the MPQF, a comparison expression is defined as common by the following term: A op B, where $A, B \in$ OperandClass and op $\in \{<, >, =, \leq, \geq, !=, \oplus\}$. The symbol $\oplus$ defines a contain operation for strings. OperandClass denotes a representation of a specific data type, such as Boolean, string, arithmetic, date and time, or duration. Note that both operands (A, B) must belong to the same OperandClass within a comparison expression. Every element of an OperandClass can be described by the data type value, an XPath expression pointing to a specific data type value, or a corresponding expression (for example, string or arithmetic) resulting in specific data type value. String, arithmetic, and Boolean expressions are similarly defined as comparison expressions but restrict their operands (A, B) to the corresponding data type (for example, string operands for string expression, arithmetic operands for arithmetic expressions, and so on).

The following short example (Code 1) demonstrates the use of a comparison and arithmetic expression for MPEG-7 documents. By reverting to the previous term definition A op B, the example instantiates for op an equal comparison operation, for the operand A an arithmetic expression, and for B an arithmetic value (value 0). In series, the arithmetic expression is composed of the operation (op) modulus: the operand A is a relative XPath expression pointing to an arithmetic value (totalNumOfSamples attribute of the AudioLLDVectorType type) and operand B again is an arithmetic value (value 2). During an evaluation process, this filter condition would result in documents that have an even number in the totalNumOfSamples attribute of the AudioLLDVectorType type.

```
<Condition xsi:type="Equal" >
 <ArithmeticExpression xsi:type=
          "Modulus">
  <ArithmeticField typeName=
          "AudioLLDVectorType">
   /SeriesOfVector[@totalNumOfSamples]
  </ArithmeticField>
  <LongValue>2</LongValue>
 </ArithmeticExpression>
 <LongValue>0</LongValue>
</Condition>
```

The current MPQF standard provides the following set of query types:

- QueryByMedia specifies a similarity or exact-match query-by-example retrieval where the example media can be an image, video, audio, or text.
- QueryByDescription specifies a similarity or exact-match query by example retrieval where the example media is presented by any XML-based multimedia description format (for example, MPEG-7).
- QueryByFreeText specifies a free text retrieval where the focused or to-be-ignored fields can be declared.
- QueryByFeatureRange specifies a range retrieval for, for example, low-level features like color.
- SpatialQuery specifies the retrieval of spatial elements within media objects (for example, a tree in an image), which might be connected by a specific spatial relation.
- TemporalQuery specifies the retrieval of temporal elements within media objects (for example, a scene in a video), which might be connected by a specific temporal relation.
- QueryByXQuery specifies a container for limited XQuery expressions.
- QueryByRelevanceFeedback specifies a retrieval that takes into consideration result items of a previous search as bad and/or good examples.

- QueryByROI specifies a retrieval on (spatio-temporal) region of interest in media resources.

Output Query Format

The OQF deals with the specification of a standardized format for multimedia query responses (see the Output element in Figure 1). The two main components cover paging functionality and the definition of individual result items. Additionally, the OQF provides a means for communicating global comments (by the GlobalComment element) and status or error information (by the SystemMessage element). Using a global comment, the responder can send general messages, such as the service subscription expiration notice or a message from a sponsor that is valid for the whole result set. When a proper result set can't be composed, or when a special message regarding the system behavior should be communicated with the client, the multimedia service can use the SystemMessage element. This element provides three different levels for signaling problems, namely status, warning, and exception. The codes and descriptions for the individual elements are defined in annex A of the standard specification. Finally, the validity period of a result set is indicated by the expirationDate attribute.

The use of the maxPageEntries attribute at the input query expresses a client's desire to retrieve a result set divided into individual pages. If this flag is set, the multimedia service is responsible for dividing the complete result set into a series of individual MPQF instance documents. For this purpose, the OQF provides two attributes, namely currPage and totalPages, to identify the individual pages.

The ResultItem element of the OQF holds a single record of a query result. In the MPQF schema, the element is based on an abstract type targeted at future extensibility and allows more concrete instantiations. Figure 6 illustrates the standardized version of such an extension.

The ResultItem has four attributes and six elements. The four attributes are recordNumber, rank, confidence, and originID. The recordNumber is a positive integer and the only required attribute. The recordNumber ensures the distinct identification of each record among the record set returned for the given query, and can be used in relevance feedback retrieval to refer to the relevance records. The rank is an optional attribute to indicate the record's relative similarity to the submitted query. The confidence is an optional attribute to demonstrate the subjective correctness of the query result. The originID is an optional attribute to indicate from which URI the specific record came. For example, when there are multiple service providers involved with answering a given query, the originID can help identify the service provider from which the result item is received. The available elements follow:

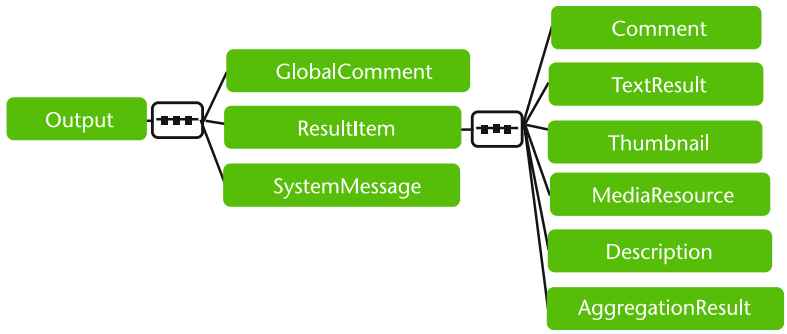

Figure 6. Schema diagram of the Output Query Format.

- Similar to the GlobalComment element, the comment is a placeholder for a text message to be transmitted to the client. The contained information should be focused on one specific result item.
- The TextResult element holds the retrieved result item as text type.
- The Thumbnail element carries the URL of a specific result item's thumbnail image.
- The MediaResource element contains the URL pointing to the location of the media resource of the retrieved result item. For example, a URL to the video or audio file.
- The Description element is a container for any kind of metadata response based on an XML schema. For example, if the multimedia service is composing the result set based on the MPEG-7 standard, the Description element holds an MPEG-7 instance document; and if the service is composing the result set based on the TV-Anytime standard, the Description element can hold an instance document using TV-Anytime metadata.

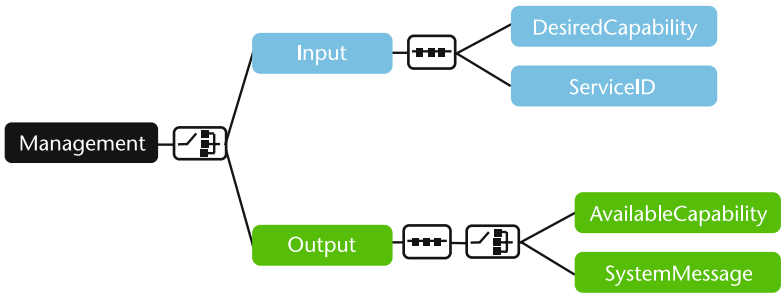

Figure 7. Overview of the element hierarchy of the MPEG Query Format's management tools.

- The AggregationResult element allows for schema-valid instantiation of results of an aggregation operation (for example, SUM). The main difficulty in expressing an aggregation operation is its missing Description element within the service's XML schema. Therefore, an aggregated element is identified through the aggregateID attribute in the OutputDescription element of the corresponding input query.

All these elements, except AggregationResult, can have an optional fromREF attribute and can occur a maximum of two times within one result item. This attribute indicates the origin result set in case of a Join operation.

Query management tools

The management part of the MPQF copes with the task of searching for and choosing desired multimedia services for retrieval. This part includes service discovery, querying for service capabilities, and service capability descriptions. Figure 7 depicts the element hierarchy of the management tools in the MPQF. As described previously, the management part of the query format consists of either the input or output element, depending on the direction of the communication (request or response).

A management request helps find suitable services (for example, by interacting with a registry service) that support the intended queries or scan individual services for their capabilities. The service's capability is described by its service capability description, which determines the supported query format, metadata, media formats, query types, expressions, and usage conditions. A management response contains the results of a service discovery request initiated by the requester. A service, aggregated service provider, or registry service returns either a list of available service capability descriptions or a system message in case of an error. If no service is available or there is no match to the requested capabilities, an empty Output element is returned.

A service capability description determines what type of retrieval functionality the respective service supports. The following elements are supported:

- SupportedQFProfile describes a service's supported query format profile. A QFProfile may define a subset of available query types and expressions that a service is capable of processing.

- SupportedMetadata describes the metadata that can be processed by a certain service using a list of addresses (URIs).

- SupportedExampleMediaTypes and SupportedResultMediaTypes indicate the media formats supported by a multimedia service for processing and responses. The formats are specified by MIME types.

- SupportedQueryTypes and SupportedExpressions describe the query types and expressions supported by a search condition. Annex A of the standard specifies the allowed information, types, and expressions based on classification scheme terms.

- UsageConditions describe a certain service's usage conditions (for example, payment is needed, authentication is required, and so on). Similar to query types, usage conditions are listed in a classification scheme. The assignment and validity of a usage condition can be specified fine granular (for example, for a specific query type) or general for the whole service.

The following code (Code 2) shows an example of a service capability description. The supported profile in this example is full. The supported metadata is based on the MPEG-7 schema and the supported media types include MP3 and Advanced Audio Coding, and result set, respectively. Moreover, the description supports the following functions: query types (100.3.6.1 stands for QueryByMedia), expressions (100.3.1 means that all Boolean expression types are supported), and usage conditions (200.1 notifies that the

service requires authentication). For simplicity and space concerns, we don't present the full classification scheme's Uniform Resource Name.

```
<AvailableCapability
    serviceID="http://Service-1">
  <SupportedQFProfile href="urn:...:
        Full"/>
  <SupportedMetadata>urn:mpeg:mpeg7:2004
  </SupportedMetadata>
  <SupportedExampleMediaTypes>audio/mp3
  </SupportedExampleMediaTypes>
  <SupportedResultMediaTypes>audio/aac
  </SupportedResultMediaTypes>
  <SupportedQueryTypes href="urn:...:
        100.3.6.1"
    usageRefList="ex1id1"/>
  <SupportedExpressions href="urn:...:
        100.3.1"/>
  <UsageConditions href="urn:...:full:
        200.1"
    usageID="ex1id1"/>
<AvailableCapability>
```

The MPQF allows the expression of several requests, such as query-by-image-example or query-by-free-text, which a natural-language-processing engine might need to analyze. For a multimedia service provider, it's likely that not all functions specified in the MPQF are supported. In such an environment, one of the important tasks for an MPQF client is to find the services that provide the desired query functions and contain the desired media format. The MPQF management tools support client identification of desired services. For this purpose, four conceivable service discovery scenarios are specified:

- Tell me everything you know.
- Tell me the services supporting the desired capability.
- Tell me the capability of the specified services.
- Tell me if the specified services support the specified capability.

The ''tell me everything you know'' scenario assumes that a user knows the location of an MPQF or registry service, and then wants to know all the service provider's functions or all registered services. In this case, the client issues an empty management input request to the service. If the service provides an aggregate service, all information on services is collected and returned.

In the ''tell me the services supporting the desired capability'' scenario, a client tries to find the service providing the client's desired functions. The DesiredCapability element expresses this scenario. This element states the requested capabilities (for example, metadata model, query types, and so on) the multimedia service must provide.

In the ''tell me the capability of the specified services'' scenario, the client is provided with a means to ask for information on all capabilities that a set of services (identified by the ServiceID element) provide.

In the ''tell me if the specified services support the specified capability'' scenario, the service discovery process is a combination of the DesiredCapability and ServiceID elements. In this case, a client is asking whether a specific set of services supports a given capability description.

There is a common return to a service-discovery request. For any type of management request, the same formatted instance is returned. The result is embedded in the management part's Output element and contains several AvailableCapability elements (as presented in Code 2) describing all matching services.

There are two different mechanisms for service selection: either the client connects directly to a service and performs the multimedia queries, or the client uses a service provider, which knows the other services and forwards the queries to them. Connecting directly to one service has the limitation that a client can only request queries corresponding to the service capabilities. A service provider, on the other hand, can distribute parts of the query to different services, which might be specialized in handling this query type.

Example scenarios

As mentioned previously, the MPQF provides many noteworthy concepts and filter criteria for the expression of multimedia requests. In this section, we highlight some important characteristic features of the MPQF and explain them in the context of usage scenarios.

Simple scenario

In this example, we show the use of a combination of free text and conditions over

Figure 8. Sample image for spatial query.

the XML metadata. Keywords are the most common way for searching, and they capture user information requirements satisfactorily in most situations. However, in multimedia content searches, the coexistence of many different media types (possibly in many different formats) often results in limited keyword-based searches. Typically, these contain explicit conditions regarding the features of the digital objects to be retrieved (for example file format, file size, resolution, and language). These searches, though simple, are common, and the MPQF allows them to be expressed in a simple way.

Consider, for example, a situation in which a professional user wishes to buy large images of Hong Kong to illustrate a publication. After capturing the user criteria through a form or some other kind of user-friendly interface, these criteria could be formalized using the MPQF and submitted to one or more service providers. The example query in the following code (Code 3) shows how QueryByFreeText and conditions over the XML metadata can be combined to express the user's image requirements. The query requests images (in any format) that are related to the keywords ''Hong Kong'' and which have a width greater than 1,000 pixels. In this case, the query is expressed in terms of MPEG-7 metadata, but other formats could be possible if the service provider required alternatives.

```
<MpegQuery>
 <Query>
  <Input>
   <OutputDescription thumbnailUse=
              "true">
    <ReqField typeName="MediaInformation
              Type">
      MediaProfile/MediaFormat/File
              Size</ReqField>
    <ReqField typeName="Creation
              InformationType">
      Creation</ReqField>
   </OutputDescription>
   <QueryCondition>
    <TargetMediaType>image/*</Target
          MediaType>
    <Condition xsi:type="AND" preference
          Value="0.1">
     <Condition xsi:type="QueryByFree
          Text">
      <FreeText>Hong Kong</FreeText>
     </Condition>
     <Condition xsi:type="GreaterThan
          Equal">
      <ArithmeticField
        typeName="MediaInformation
          Type">
        MediaProfile/MediaFormat/
          Frame@width
      </ArithmeticField>
      <LongValue>1000</LongValue>
     </Condition>
    </Condition>
   </QueryCondition>
  </Input>
 </Query>
</MpegQuery>
```

Query-by-example scenario

Here, we show use of QueryByMedia in the medical domain. Query-by-example offers an alternative approach whereby a search can be expressed using one or more example digital objects (for example a number of image files). Description instead of the example object is also considered query-by-example. In the MPQF, these two situations are differentiated: query-by-media applies to queries including digital media objects, while query-by-description uses the metadata feature descriptions.

Imagine an example scenario related to the medical domain. The Lister Hill National Center for Biomedical Communications at the US National Library of Medicine maintains a digital archive of 17,000 cervical and lumbar spine images. This collection of images is cataloged in a limited way due to the prohibitive costs of having radiologists analyze and annotate them with metadata. Let's imagine a doctor involved in an epidemiology or clinical study trying to find like reference material from prior stored images. The doctor would probably query on the basis of one or more example images and would retrieve a list of images ranked by similarity. This retrieval would be achieved in the MPQF through the use of QueryByMedia, which uses a media sample (such as an image or video) as the key for search.

The following code (Code 4) shows how this query type would be used in combination with other conditions over the XML metadata to fulfill the described use case. The query requests JPEG images that are similar to the given sample image and that have an attached

Dublin Core metadata descriptor date greater than 2002-01-15.

```
<MpegQuery>
 <Query>
  <Input>
   <QueryCondition>
    <TargetMediaType>image/*</Target
          MediaType>
    <Condition xsi:type="AND">
     <Condition xsi:type="QueryByMedia">
      <MediaResource
       xsi:type="MediaResourceType">
       <MediaResource>
        <InlineMedia type="image/jpeg">
         <MediaData64>R0lGODlhDwAPAKECA
         AAAzMzM/////wAAACwAAAAADwAPAAA
         CIISPeQHsrZ5ModrLlN48CXF8m2iQ3
         YmmKqVlRtW4MLwWACH+H09wdGltaXp
         lZCBieSBVbGVhZCBTbWFydFNhdmVyI
         QAAOw==</MediaData64>
        </InlineMedia>
       </MediaResource>
      </MediaResource>
     </Condition>
     <Condition xsi:type="GreaterThan
           Equal">
      <DateTimeField>date</DateTimeField>
      <DateValue>2002-01-15</DateValue>
     </Condition>
    </Condition>
   </QueryCondition>
  </Input>
 </Query>
</MpegQuery>
```

Query-by-example description scenario

QueryByMedia and QueryByDescription are the MPQF's fundamental operations and represent the query-by-example paradigm. The individual difference lies in the sample data provided. QueryByMedia uses a media sample, such as image or video, as a key for search, whereas QueryByDescription allows querying on the basis of an XML-based description key. The example query in the following code (Code 5) shows how an example description can be included in a query to form a condition. The query requests JPEG images that possess descriptors exactly matching the attached MPEG-7 descriptors.

```
<MpegQuery>
 <Query>
  <Input>
   <QueryCondition>
    <TargetMediaType>image/jpeg</Target
              MediaType>
     <Condition xsi:type="QueryBy
              Description"matchType=
                 "exact">
      <DescriptionResource resourceID=
              "desc07">
       <AnyDescription
            xmlns:mp7="urn:mpeg:mpeg7:
                  schema:2004">
        <mp7:Mpeg7>
         <mp7:DescriptionUnit
           xsi:type="mp7:Creation
              InformationType">
         <mp7:Creation>
          <mp7:Title>World Cup 2006</
              mp7:Title>
         </mp7:Creation>
        </mp7:DescriptionUnit>
       </mp7:Mpeg7>
      </AnyDescription>
     </DescriptionResource>
    </Condition>
   </QeuryCondition>
  </Input>
 </Query>
</MpegQuery>
```

SpatialQuery

The sample in the following code (Code 6) illustrates the use of SpatialQuery, which allows a certain spatial relation to be used as a condition. Let's assume that a multimedia service provides a large set of images showing different animals.

```
<MpegQuery>
 <Query>
  <Input>
   <QFDeclaration>
    <Resource xsi:type="MediaResource
              Type"
          resourceID="animalA">
     <MediaResource>
      <MediaUri>http://db.mpqf.mpeg
              /cat.jpg</MediaUri>
     </MediaResource>
    </Resource>
    <Resource xsi:type="MediaResource
              Type"
          resouceID="animalB">
     <MediaResource>
      <MediaUri>http://db.mpqf.mpeg/
              dog.jpg</MediaUri>
     </MediaResource>
    </Resource>
   </QFDeclaration>
   <OutputDescription mediaResourceUse=
              "true"/>
    <TargetMediaType>image/jpeg</Target
              MediaType>
    <Condition xsi:type="SpatialQuery">
     <SpatialRelation
         sourceResource="animalA"
         targetResource="animalB"
              relationType=
         "urn:mpeg:mpqf:cs:Spatial
              RelationCS:2008:left"/>
    </Condition>
   </QueryCondition>
  </Input>
 </Query>
</MpegQuery>
```

As Figure 8 demonstrates, a user might be interested in finding all images that show a cat

and a dog and that have the cat on the dog's left side. In this case, the query presented in Code 6 formulates this request by using the SpatialQuery condition and a respective spatial relation. Note that this example assumes that the multimedia service is capable of extracting the necessary information (object recognition, low- or high-level features, and so on) to detect the animals. However, the query can be modified using further metadata descriptions as input parameters.

The query in Code 6 specifies that the output should use media resources by asserting as ''true'' the attribute mediaResourceUse in the OutputDescription element. Thus, a possible result set satisfying the request might be of the form shown in the following code (Code 7). Each result item is expressed by a MediaResource element and contains recordNumber and rank information.

```
<MpegQuery mpqfID="mm1221">
 <Query>
  <Output>
   <ResultItem recordNumber="201" rank=
              "1">
    <MediaResource>http://mpqf.org/
           animal1.jpg
    </MediaResource>
   </ResultItem>
   <ResultItem recordNumber="202" rank=
              "2">
    <MediaResource>http://mpqf.org/
           animal2.jpg
    </MediaResource>
   </ResultItem>
   <ResultItem recordNumber="203" rank=
              "2">
    <MediaResource>http://mpqf.org/
           animal3.jpg
    </MediaResource>
   </ResultItem>
  </Output>
 </Query>
</MpegQuery>
```

QueryByRelevanceFeedback

A relevance-feedback approach is a common and widely used multimedia-retrieval paradigm that employs client interaction to improve retrieval efficiency. It allows users to tune ongoing retrieval by indicating good and/or bad examples of previous result sets. If the result sample of Code 7 is considered a starting point, there are three result items. As an example, the first and third result items are considered to be good results, while the second is a poor match. The example shown in the following code (Code 8) expresses a relevance feedback condition by indicating the two positive examples (recordNumber is used) and one negative example by prefixing a NOT condition. The previous result set is identified by its mpqfID attribute value (mm1221).

```
<MpegQuery>
 <Query>
  <Input>
   <QueryCondition>
    <Condition xsi:type="AND">
     <Condition xsi:type="QueryBy
           RelevanceFeedback"answerID=
             "mm1221">
      <ResultItem>201</ResultItem>
      <ResultItem>203</ResultItem>
     </Condition>
     <Condition xsi:type="NOT">
      <Condition xsi:type="QueryBy
           RelevanceFeedback"answerID=
             "mm1221">
       <ResultItem>202</ResultItem>
      </Condition>
     </Condition>
    </Condition>
   </QueryCondition>
  </Input>
 </Query>
</MpegQuery>
```

Conclusions and outlook

The success of the MPQF depends primarily on the following issues:

- As shown previously, there are many possible multimedia retrieval interfaces. Therefore, the number of simple interpreter implementations that map MPQF requests to target retrieval systems is of vital importance.

- To enhance interoperability among retrieval systems, which is one of the MPQF group's main aims, research and development needs to be carried out to support the distribution of one request among heterogeneous retrieval systems (for example, simultaneously address an MPQG-7 aware database and Dublin Core aware database) and to aggregate those individual results.

- The MPQF came out of MPEG-7 activities, but it's not tied to MPEG-7. Hence, it's important that the multimedia user community recognize the MPQF's versatility. This versatility allows groups of users to define their own metadata sets (for example as folksonomies and informal tags) while retaining interoperability in the vital search interface.

In addition to working toward finalizing the MPQF so that it can be published as an ISO/IEC standard, further work will include

creation of a set of reference software for the MPQF by April 2009. The reference software will concentrate on client and server tools and will be available as an amendment to the MPQF standard. Furthermore, investigations will consider its adaption and use within current MPEG technologies such as MPEG-21 or MPEG-A.

The standardization committee ISO/IEC JTC 1 SC29 WG1 (JPEG) initiated the JPSearch project in 2004 to develop relevant technologies to enable search-and-retrieval applications that focus on JPEG image archives. Recently, the JPEG community decided to adopt the MPQF for the specification of the JPSearch Query Format. JPSearch has decided to use an adapted version of MPQF as its JPSearch Query Format.

Currently, three funded European Union projects are planning to use the MPQF as an interface for their multimedia retrieval engines. The first, the Pharos project (see http://www.pharos-audiovisual-search.eu/) is developing a platform for the retrieval of audiovisual content. The Sapir project (see http://www.sapir.eu/) concentrates on the development of a large-scale, distributed, P2P architecture that will make it possible to search audiovisual content using the query-by-example paradigm. Finally, the Enthrone project (see http://www.ist-enthrone.org) targets an integrated management solution that covers the entire audiovisual service distribution chain, including protected content handling, distribution across networks, and reception at user terminals based on the MPEG-21 framework. **MM**

Acknowledgment

The authors thank all the MPQF members for their fruitful input and discussions during the standardization process.

References

1. M. Döller and H. Kosch, ''The MPEG-7 Multimedia Database System (MPEG-7 MMDB),'' *J. Systems and Software*, vol. 81, no. 9, 2008, pp. 1559-1580.
2. M. Cord, P.-H. Gosselin, and S. Philipp-Foliguet, ''Stochastic Exploration and Active Learning for Image Retrieval,'' *Image and Vision Computing*, vol. 25, no. 1, 2007, pp. 14-23.
3. L. Xue et al., ''VeXQuery: An XQuery Extension for MPEG-7 Vector-Based Feature Query,'' *Proc. Int'l Conf. Signal-Image Technology and Internet- Based Systems*, 2006, pp. 176-185, http://www.u-bourgogne.fr/SITIS/06/Proceedings/index.htm.
4. P. Lui, A. Charkraborty, and L.H. Hsu, ''A Logic Approach for MPEG-7 XML Document Queries,'' *Proc. Extreme Markup Languages*, 2001; http://www.idealliance.org/papers/extreme/proceedings/html/2001/Liu01/EML2001Liu01.html.
5. A. Henrich and G. Robbert, ''POQL^{MM}: A Query Language for Structured Multimedia Documents,'' *Proc. 1st Int'l Workshop Multimedia Data and Document Engineering* (MDDE), 2001, pp. 17-26.
6. J.D.N. Dionisio and A.F. Cardenas, ''MQuery: A Visual Query Language for Multimedia, Timeline and Simulation Data,'' *J. Visual Languages and Computing*, vol. 7, no. 4, 1996, pp. 377-401.
7. I. Schmitt, N. Schulz, and T. Herstel, ''WS-QBE: A QBE Like Query Language for Complex Multimedia Queries,'' *Proc. 11th Int'l Multimedia Modeling Conf.* (MMM), IEEE CS Press, 2005, pp. 222-229.
8. N. Furh and K. Grossjohann, ''XIRQL: A Query Language for Information Retrieval in XML Documents,'' *Proc. 24th ACM-SIGIR Conf. Research and Development in Information Retrieval*, ACM Press, 2001, pp. 172-180.
9. J. Robie. *XQL (XML Query Language)*, Aug. 1999; http://www.ibiblio.org/xql/xql-proposal.html.
10. G. Conforti et al., ''The Query Language TQL,'' *Proc. 5th Int'l Workshop Web and Data Bases* (WebDB), 2002; http://db.ucsd.edu/webdb2002/papers.html.
11. S. Boag et al., *XQuery 1.0: An XML Query Language*, W3C recommendation, Jan. 2007; http://www.w3.org/TR/2007/REC-xquery-20070123/.
12. D. Bonino, F. Corno, and P. Pellegrino, ''RQL: A Declarative Query Language for RDF,'' *Proc. 11th Int'l World Wide Web Conf.* (WWW), ACM Press, 2002, pp. 592-603.
13. E. Prud'hommeaux and A. Seaborne. *SPARQL Query Language for RDF*, World Wide Web Consortium (W3C) recommendation, Jan. 2008; http://www.w3.org/TR/rdf-sparql-query/.
14. J.M. Martinez. *MPEG-7 Overview*, ISO/IEC JTC1/SC29/W11 N5525, MPEG Requirements Group, March 2003.
15. M. Fernandez et al., *XQuery 1.0 and XPath 2.0 Data Model (XDM)*, W3C recommendation, Jan. 2007; http://www.w3.org/TR/xpath-datamodel/.
16. *Common Object Request Broker Architecture*, Object Management Group, 2008; http://www.omg.org/spec/CORBA/.
17. *Universal Description Discovery and Integration*, OASIS, 2004; http://www.oasis-open.org/specs/.
18. M. Gruhne et al., ''Distributed Cross-Modal Search with the MPEG Query Format,'' *Proc. 9th Int'l Workshop Image Analysis for Multimedia Interactive Services*, IEEE CS Press, 2008, pp. 211-214.

Contact author Mario Döller at mario.doeller@uni-passau.de.

Contact editor John R. Smith at jsmith@us.ibm.com.