Diplomarbeit
an der
Technische Universität Wien
und Universität Wien

Creation of an MPEG-7 Feature Extraction Plugin for the platform METIS

- on basis of Web Services -

vorgelegt von
Andrew Lindley

eingereicht bei
Univ.Prof.Dr. Wolfgang Klas

Institute for Distributed and Multimedia Systems
Faculty of Computer Science
Liebiggasse 4/3-4, 1010 Wien

zur Erlangung des akademischen Grades
Magister rerum socialium oeconomicarumque

Wien, Mai 2006

Die Arbeit ist in englischer Sprache verfasst.

Contact Information

Andrew Lindley
Matrikelnummer: 0025314
E175

Leitermayergasse 31/10
A-1180 Vienna
Austria

andrew.lindley@qse.ifs.tuwien.ac.at

Eidesstattliche Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benützt und die aus fremden Quellen wörtlich oder inhaltlich entnommenen Gedanken als solche kenntlich gemacht habe. Die Arbeit wurde bisher weder in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt noch veröffentlicht.

Wien, am 24. Mai 2006 _____ Unterschrift

Danksagung

Zu aller Erst möchte ich mich ganz herzlich bei meinen Eltern bedanken, dass sie mir diese Ausbildung ermöglicht haben und mir auch sonst ins allen Belangen immer zur Seite gestanden sind. Vielen herzlichen Dank dafür und auch für eure Geduld die ich mit Fortlauf der langwierigen Arbeit zur Genüge auf die Probe gestellt habe. Ganz besonderer Dank gilt meinem Vater dafür, dass er ein Auge auf das Englisch meiner schriftlichen Arbeit geworfen und unzählige Stunden für das Korrekturlesen geopfert hat.

Weiters möchte ich mich ganz speziell bei meinem Betreuer Herrn Dr. Ross King für seine kompetente Unterstützung und seine flexible Betreuung während der gesamten Arbeit und noch darüber hinaus, bedanken. Meinem Prüfer Prof. Klas, dessen kompetente und ansprechende Art mich sehr begeistert und letztendlich auch den Ausschlag dazu gegeben hat diese Diplomarbeit in Kooperation mit den Researchstudios zu verwirklichen, möchte ich ebenfalls besonders hervorheben. Die enge Zusammenarbeit zwischen DME und der Arbeitsgruppe von Prof. Klas hat zu jeder Zeit wunderbar funktioniert.
An dieser Stelle sei es mir gestattet Prof. Horst Eidenberger an der TU Wien dankend zu erwähnen, da wir ohne sein Fachwissen das eXperimentation Model wohl nicht zum Laufen gebracht hätten.

Last but not least möchte ich diese Arbeit meiner Partnerin Marianne widmen, die es immer wieder schafft ein Lächeln auf meine Lippen zu zaubern und mich unzählige Male motiviert hat.

Abstract

Content Based Image Retrieval (CBIR), i.e. querying multimedia material by making use of the data's feature characteristics, is an emerging field. A variety of standards for modeling different aspects of audio-visual information exist and are mainly distinguished by the level of complexity they are able to capture. One of them, the ISO/IEC standard MPEG-7, also known as Multimedia Content Description Interface, offers a large set of various application-independent tools for adding a degree of information which can be passed to and accessed by devices.

The technology is to be supported by the multimedia middleware framework METIS due to its needs for sorting out duplicate images in its repository and offering basic CBIR functionality. The developed solution for extracting MPEG-7 conform descriptions is based on the standard's reference implementation, the eXperimentation Model (XM).

In the first part of this paper an overview of the underlying technologies is presented focusing on the metadata standard and on the quality of its descriptors. The second part deals with the implemented software solutions including, on the one hand a Java wrapper software for offering the XM feature extraction functionality as a multi-user web service and on the other hand a plugin for consuming and integrating the remote service into a METIS data model.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

The approach of integrating structures for Content Based Image Retrieval (CBIR) in the multimedia middleware framework METIS is prominent due to the fact that providing this feature is essential for certain audio-visual application scenarios for which the system serves as basis. An introduction to this topic and brief knowledge is already available through former plugin development projects, enabling the system to track duplicate images in its repository. However as the implemented mechanism can neither be easily extended to provide a broader range of feature characterization, nor is it based on a common data model, there is a need to bring the CBIR approach to the next level. That means to offer support for emerging standards like MPEG-7, the Multimedia Content Description Interface and its rich set of standardized description tools as well as provide mechanisms for automatically or semi-automatically extracting them from the system's given AV material. This data can then be used for building description schemes for multimedia classification or for creating feature based queries.

In the stage of doing research on the field of CBIR tools and extraction algorithms, the eXperimentation Model, a non-normative reference software implementation for MPEG-7 was identified to mostly match our requirements as it offers applications for triggering MPEG-7 compliant feature extractions and contains a good set of implemented algorithms. Although the software was used for proof of concept in the standardization process it lacks of detailed documentation on its configuration - even getting it to work is a challenge. However, properly configuring the settings is mandatory for receiving suitable results. Therefore all available information on the framework including hints from expert correspondence are aggregated in this paper to give a complete overview on the XM for setting up its infrastructure and for understanding its design.

The architecture of the XM framework does not allow directly adding an interface for automatically using its applications, so a solution for wrapping the service is required. Due to various limitations the eXperimentation Model has from its underlying hardware and software, as well as the fact that there is actually no need to run the METIS instance on the same machine the XM is available on, a solution for separating the components is presented. Therefore the main part of the software implementation process concerns the creation of a multi-client server architecture for offering the MPEG-7 feature extraction functionality with a well-defined interface and a platform-independent technology.

Finally in addition to the METIS aspect, the "feature-extraction as web-service" also very well fits the BRICKS approach, a distributed, service-oriented architecture for European cultural institutions, where the work is expected to be continued. The resulting web service solution, however, is not restricted to BRICKS but has the main advantage of being accessible by any application which wants to make use of the XM for MPEG-7 feature extraction functionality.

1.2 Thesis Outline

The requirements for this master thesis are to offer support for a basic set of MPEG-7 Visual descriptors by extending METIS through its plugin mechanism. For extracting feature characteristics from its attached media the MPEG-7 reference software is used, which is made available as a stand-alone component by using the web service technology, including functionality for remote data handling and a wrapper to set up its external infrastructure. The created semantic pack must be capable of capturing the description results. On basis of this provided data further CBIR functionality can be built.

First of all, it is important to have an insight into the main technologies upon which the practical work is built. METIS is the core technology the developed solutions extend and are dependent on - it is described first. The paper provides a brief introduction to its main concepts as e.g. semantic packs and plugins for extending the core functionality and tries to depict the system's design approach of being flexible at every level.

Afterwards we point out why there is an actual need for products like MPEG-7 and subsequently we present the Multimedia Content Description Interface. Its normative elements as well all Visual Descriptors for capturing multimedia characteristics at various abstraction levels are described in detail. Finally a short comparison to other textual description standards is given.

The chapter about the MPEG-7 reference software XM concentrates on assembling all rarely available information about obtaining, compiling and configuring the framework, containing valuable hints we retrieved through communication with several expert groups. Finally a qualitative analysis is presented on how efficiently the MPEG-7 Descriptors cover their intended application domains in the task of distinguishing different media content. The last part of this section presents a selection on current projects and interesting tools built on either XM or the Multimedia Content Description Interface. As currently no overal picture on working groups developing with the standard is available the presented resources may be helpful for carrying out further research activity.

Finally a brief introduction to the web service technology completes the overview of underlying technologies. It describes the SOAP Engine Apache AXIS for integrating web service support into Java and presents reasonable extensions for transferring binary data via SOAP.

The second part of this paper then deals with the designed software solutions, i.e. the XM Server - a web service implementation of the eXperimentation Model including support for multi-client and remote file handling, the XMFE METIS plugin - a client software for integrating feature extraction support for METIS, and the XMBasic semantic pack - a data model which is capable of capturing the extracted data. The software is described by outlining its main design paradigms, its implemented functionality as well as the problems it is currently suffering from. For a better understanding, UML diagrams are used.

Part I
Overview of Underlying Technologies

2 METIS - A Highly Flexible Multimedia Middleware Framework

The following sections will give a brief overview on the core functionality of the multimedia middleware framework METIS and its architecture. We will primarily concentrate on its expressive and flexible data model for media description and classification as well as on the concepts of semantic packs and kernel plugins to demonstrate that METIS is able to serve as the basis for most multimedia applications.

2.1 Introduction and System Overview

The problem of common multimedia database systems available these days is that they only focus on the management of a single type of media (e.g. videos or images). Their inflexible architecture that often concentrates on a single application domain makes it difficult to adapt them to individual needs.

METIS - a joint project by the University of Vienna[1] and Digital Memory Engineering (DME) of ARC Seibersdorf Research[2] - is a general-purpose multimedia database which offers unified and integrated management of media and at the same time provides highly customizability at all architectural levels. Following the design motivation of "being highly flexible at every level" the middleware application METIS is located between a persistence abstraction layer - allowing the system to operate on a large-scale relational/object database, as well as on a small-scale file storage (e.g. a XML Serialization) - and a highly customizable visualization layer. The expressive METIS data model that builds the system's core, allows any possible definition of arbitrary media types (e.g. for classification), the associations between them, their description (e.g. by metadata attributes) as well as features for media retrieval (e.g. by automatic feature extraction mechanisms). [1] Furthermore to cope with the complexity of these tasks and to offer portability to every possible application specific field, the concepts of *semantic packs* and kernel *plugins* are introduced. These features are discussed in detail in the section 2.4**.**

Building data models is as comfortable for developers as creating them at the level of ontologies (for example OWL or RDF [5]) which can then easily be mapped to METIS by existing tools.

The flexibility and interoperability is also reflected by the techniques and

[1] see http://www.informatik.univie.ac.at
[2] see http://dme.researchstudio.at/en/

technologies METIS is built on. The METIS core is implemented as a Java servlet - packed as a war archive - and so cannot only easily be deployed on different platforms but even on different application servers. By using a highly configurable visualization pipeline built on the Apache Cocoon framework, it is possible to adapt the user interface to the target application's need and at the same time have a strict separation of content, style and logic.

By managing all these requirements and offering flexibility from the back- to the frontend, METIS is able to be adapted to be the basis of most multimedia applications - other systems following the mentioned paradigms do exist, but are found rather rarely in the field. [1]

2.2 METIS Basic Data Model

Figure 1 illustrates the main elements on which the core data model is built on as well as their interrelationships. On the one side we are working on the level of instances - all basic media like images, text, etc. are represented by *single media objects* (short SMOs) that can be considered as an abstract, logical representation of a certain media. The actual media files (e.g. a picture of a stamp) are attached to a SMO via *media instances.* A Media Instance carries metadata (like e.g. sampling rate, encoding format, etc.) of the file it represents as well as a *media locator,* a kind of pointer, which allows METIS to address the resource on different storing mechanisms like file systems, web servers or databases. [1] Note that the architecture of a SMO which is able to arrange and gather multiple media instances - even of the same media instance - under one logical object is a very flexible and useful structure. In that way it is possible to let the system decide which media instance to use in certain situations, application scenarios, etc. - based upon the provided meta information. Just imagine a video stream of a soccer match which is present in two different quality levels where the system may differentiate by e.g. the bitrate or the storage location which file to stream to a local PC or to a portable cell phone - further possible operations could be server load balancing or adaptions to bandwidth limitations.

On the other side it is necessary to provide a basis for semantic classification of media objects - in METIS these logical hirachical categories are known as *media types* (short MT). Due to the fact that this classification is highly application specific, METIS permits the definition of any arbitrary media type needed. Media types may join in multiple instantiation and multiple inheritance associations to classify single media objects. For more detailed descriptions of high-level characteristics or low-level features of associated media objects *metadata attributes* may be connected to the media types. These attributes are typed (e.g. standard attributes of type: string, integer but also types of arbitrary complexity like e.g. an MPEG-7 media descriptor may be easily made available to the system), they may be shared between different MTs, having different cardinalities, default values and may also be restricted to a range of values. The last part that completes the basic data model are *associations.* Associations implement a common approach to describe media objects by their relationships to other media - semantics are provided by domain specific association types. [1]

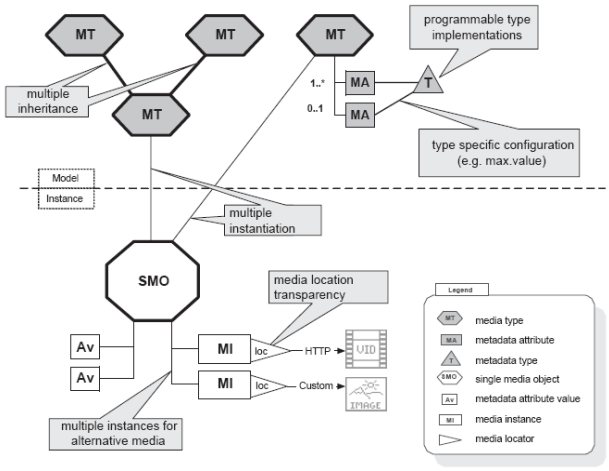

Figure 1: The METIS core data model [1]

> "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." [1, page 3]

2.3 Complex Media Objects and Templates

Although the management of multimedia documents, in other words media documents which consist of several different media items, is not the central goal of METIS, the data model includes an object type - the *complex media objects* (short CMO) - which allows a representation of such documents. CMOs are similar to SMOs meaning that they instantiate media types, they may take part in associations and may get metadata attribute values assigned. The main difference however is that they serve as containers for other (single media) objects. [1]

The problem of how to deal with temporal, spatial and interaction relationships between the different single media objects in a complex media object (for example how to synchronize a video and an audio source) was solved by outsourcing this problem to a simple but flexible template mechanism which has the big advantage that METIS is open to any existing multimedia document format standard (like MHEG, SMIL [3] or SVG). A *template* is an XML document in a certain multimedia document format (e.g. SMIL) enriched by

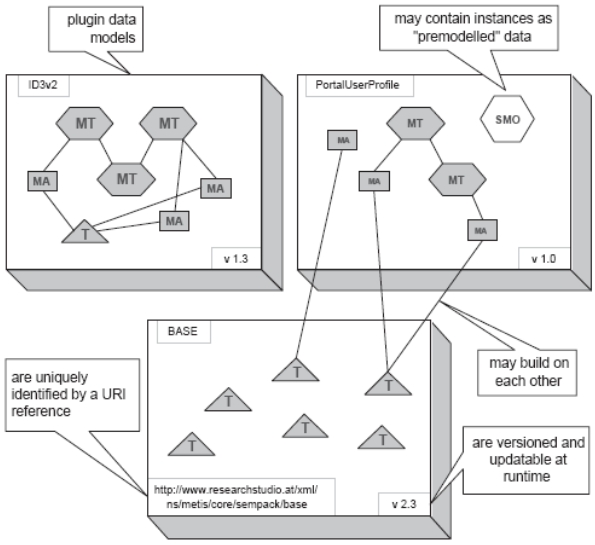

Figure 2: METIS Semantic Packs [1]

specific placeholders for media content. Whenever a presentation of a CMO is requested, these placeholders are substituted with format-compliant media references to the CMO's media - by using a format specific XSLT stylesheet at runtime. [2] As CMOs may be bound to numerous templates, it is possible to define their content independent of presentation style and format and then choose the best suiting output format for a presentation channel at runtime. Currently METIS contains default templates replacement stylesheets for XHTML, SVG and SMIL.

2.4 Semantic Packs and Plugins

On the one side the generic data model of METIS provides many advantages as presented above - but at the same time introduces the difficulty that declared types, attributes or objects have no defined semantics. An application that uses the stored data model cannot interpret attributes or media types nor distinguish between model elements with the same name from different domains. In XML this kind of problem was solved by using namespaces [4] - in METIS the concept of *semantic packs* is introduced. Semantic packs, which come as Java JAR files, are uniquely identified by an URI reference that is carried by all object names in it. The main difference between XML namespaces and semantic packs however is that latter not only groups elements belonging to the same application domain together but also physically serves as a container which includes the definitions of its objects in serialized XML form. Elements contained in an semantic pack may be metadata attributes, media type hierarchies, associations and their associated

data type implementations, media locators as well as templates for the web front-end. [2] Thereby it is possible to reduce administration complexity by prepacking application specific semantic packs and delivering a domain specific preconfigured version of METIS very easily. Modelling semantic packs is quick by using existing tools, like a plugin for the ontology editor Protégé which maps ontologies to METIS data types. In that way it is possible to supply conformity to other metadata modelling standards by just exporting an ontology to a semantic pack and importing it to METIS. [2]

A good example of a semantic pack is the METIS basic, which contains the implementations of the primitive data types like string, integer, etc. - and as it is possible to organize semantic packs hierarchically - it is the basis that most other packs make use of.

The second very important feature in the sense of modular functionality extension, which goes hand in hand with the introduction to semantic packs, is the idea of *kernel plugins.* While semantic packs provide customization for the data model, kernel plugins extend the systems functionality. They are automatically loaded Java classes, which have access to the METIS core, the total framework as well as to every semantic pack and their's functionality may extend every point of the framework (e.g. metadata extractors, extensions to the web front-end, etc.). Two kinds of kernel plugins are distinguished: light-weight plugins and heavy-weight plugins. On the one hand there are light-weight plugins - or so called *functions* - which are implemented by the metadata type and function frameworks and therefore are closely connected to their corresponding semantic pack to offer basic functionality. On the other hand there heavy-weight plugins which offer larger functional implementations and customizations to the data stored (e.g. classification plugins, etc.). They may contain collections of Java classes that implement these extensions, as well as any third party class libraries on which these implementations are based. [2] It is also important to notice that these classes not only have access to all customization frameworks in METIS but also to the event system. A plugin may subscribe to different events (e.g. MediaInstanceUpdate event) or even offer own specific events. This mechanism of interaction allows is very useful to provide a loose coupling between the different plugin extensions in an application. [2]

On details about the MPEG-7 feature extraction plugin (XMFE) and its used semantic pack developed for this master thesis see section 8.2 which will give a deeper view in the presented mechanisms.

3 MPEG-7 - the Metadata Standard

> "Audio-visual information must allow some degree of interpretation, which can be passed onto, or accessed by a device or a computer code. MPEG-7 aims to create a standard for describing these operational requirements." [6, page 65]

MPEG-7, also called the Multimedia Content Description Interface, is the ISO/IEC 15938 standard for describing multimedia content by a rich set of tools for content management, organization, navigation and automated processing. It defines a large library of elementary metadata descriptors, which can be grouped to hierarchical description schemes and also offers a language (DDL) for those who want to extend the standard. Its tools for capturing multimedia characteristics reach from basic features (timbre, colors, etc.) which can automatically be extracted from the content to highly abstract ones (e.g. objects, time and interaction) which require human annotation. Because of the fact that it is a generic standard i.e. it does not specify how its descriptors get extracted, it is able to address different applications in several environments. [13] by at the same time leaving space for improved technical industry solutions . MPEG-7 was the first standard which made an effort to formalize multimedia content description.

3.1 History of MPEG and Overview on Standards

The abbreviation MPEG - which is nowadays a very common word due to the widespread usage of digital set-top boxes and stand-alone DVD players - stands for *Moving Picture Coding Experts Group*. This working group of the ISO/IEC[3] was founded in the year 1988 with the goal to develop standards for coded representation of moving pictures, audio data as well as their combination and to define methods for their distribution and compression. While the first MPEG meeting in Ottawa (Canada) in the year 1998 involved 25 experts, meetings these days in average have the magnitude of about 300 participants from over 200 companies. Their most important standards are shortly characterized below.

MPEG-1 1993, storage and retrieval.

One of the main goals of MPEG-1 - the first product of the MPEG working group - was to define a proper storage and compression format for audio and video data. The standard consists of four parts and a reference simulation whereas the first one ("systems") handles synchronization and multiplexing and therefore provides very elementary audio and video streams. The second part ("video") distinguishes four types of image frames for processing and compression.

Disadvantages of MPEG-1 are the maximum data transfer rate which is 1.5 Mbit/s and corresponds to the data rate of a compact disc (CD), the external

[3] International Organisation for Standardisation/International Electronics Commission

addition of metadata and the low resolution of images with 352x288 pixels. This is the standard which the Video CD and MPEG Layer III (MP3) are based on. [9]

MPEG-2 1996, digital television.

The concept of MPEG-2 is in most parts very similar to MPEG-1 and thereby in wide parts a backward compatible extension. An essential advantage is the ability to scale the compressed video which allows an encoding at different quality levels as well as a variable data transfer rate between 4 and 9 Mbit/s achieved by a better compression ratio. The higher quality of videos is based on a maximum resolution of 720x576 pixels with the disadvantage of a - compared to other mediatypes - large file size. [10, page 9] MPEG-2 is a generic standard - it was defined in terms of extensible profiles, which support features required by an importing application - e.g. the main profile supports digital video transmission at a range of 2 to 80 Mbps over cable, satellite or other broadcasting channels. The multichannel audio provides among others five full bandwidth channels, two surround and seven multilingual channels. [6]

Parts 1,2 and 3 of the standard are used in products like the DVD (Digital Versatile Disc) or set top boxes for digital TV - widespread consumer products which were sold millions of times.

MPEG-4 v2 1999, coding of audio-visual contents.

While the two previous standards concentrated on the transfer, the storage and the compression of audio-visual data, MPEG-4 structurizes this data and at the same offers technologies to satisfy requirements of authors, service providers and consumers. The main aspects of MPEG-4 are [10, page 10]:

- MPEG-4 enables the representation of multimedia content by defining individual - not necessarily rectangular - objects.
- It contains standard technology to represent time-varying 3D information.
- The "system" part also provides a framework to handle the management and protection of rights arising from individual objects.
- MPEG-4 describes the alignments of objects in an audio-visual scene but also manages and synchronizes the data which is associated with these objects.
- The user may interact with a audio-visual scene.

An important MPEG-2 amendment to enable rich multimedia applications in the television domain has been developed which supports the carriage of MPEG-4 objects on MPEG-2 Transport Streams**. [9]**

MPEG-7 2001, multimedia content description interface.

While the preceding standards of the working group mainly focused on the coded representation of audio-visual information, MPEG-7 concentrates on a completely different aspect - the standardization of an interface to describe multimedial material. It offers a large set of tools and different abstraction levels - reaching from low level signal information to high level semantic characterization - for the description and interpretation of content. The main difference to other metadata standards is that it is generic and not tied to a certain application. A short testimony defined by the MPEG-alliance summarizes the quintessence of MPEG-7 best :

> While MPEG-1, 2 and 4 represent the content itself ("the bits"); MPEG-7 tries to represent the information about the content ("the bits about the bits"). [11]

A detailed description of the standard is given in the following sections.

MPEG-21 2004, multimedia framework.

The vision behind MPEG-21 is that there are many elements to build an infrastructure for the delivery, the consumption, the manipulation or organization of multimedia content - but the "big picture" to describe how these elements relate to each other has been missing. MPEG-21 assumes that there are users on the one side and digital items (e.g. a music collection) on the other side in which the users execute actions. This action then generates other digital items which may become object of transactions.

The vision for MPEG-21 is to define an open multimedia framework that will enable transparent and augmented use of multimedia resources across a wide range of networks and devices used by different communities. The intent is that the framework will cover the entire multimedia content delivery chain encompassing content creation, production, delivery, personalization, consumption, presentation, and trade. Thereby MPEG-21 does not specify the behavior of a distributed multimedia system but rather defines where standards are required in the multimedia framework. [12]

Besides these five standards MPEG-1 to MPEG-21, which were developed throughout the 17 years of activity - the group is at the moment establishing new standards, named consecutively from MPEG-A to MPEG-E. While MPEG-A is called "Multimedia Application Format" (MAF) and provides a framework for the integration of different elements from various MPEG-standards into a single specification that is usable for specific but widely usable applications - MPEG-E called "Multimedia Middleware", which is still being developed, has the goal to provide the download and execution of multimedia applications. For more details on these standards see. [8]

3.2 Why is there a Need For Products like MPEG-7?

Nowadays it has become very easy to create and publish digital content for everybody even without having a lot of technical background information. While in the beginning stages of the internet due to bandwidth limitations of 56 KB modems this material was mainly restricted to text and images, which was viewed on personal computers with standard monitors, we are now confronted with lots of audio-visual material of high quality, available for example, as the wide success of VoIP (Voice over Internet Protocol) or IP-TV (Internet Protocol Television) shows, saved in private or professional databases and spread over broadband connections and wireless networks to a multitude of different technical devices.

To have an idea about the volume we are talking about it is a good idea to look at the search engine Yahoo[4] which has 19.2 billion web documents, 1.6 billion images and more than 50 million audio and video files in its search index - with the tendancy to increase significantly.

On the other hand the largest data pool is worthless if it is not possible to search, interact or display complex and inhomogeneous material. Search engines on the internet like Google and Yahoo managed to build up index structures with increased performance - that are necessary to examine and deal with the huge amount of web-documents on the one hand, and to create a user friendly syntax for HTML retrieval on the other hand. Although search engines play an essential role for every day work and managed to become one of the most important tools on the internet, they cannot hide important deficits in the handling of multimedia information - giving us an idea why products like MPEG-7 are needed: First of all, HTML offers almost no way to add specific meta information to its content or to tag elements with it. This leads to the problem that a machine is neither able to interpret this information nor to distinguish between elements of different domains.

> "Today we need new forms of representation, allowing some degree of interpretation of the information's meaning, that a device or a computer code can access." [13, page 78]

Many technologies reacting to this problem were developed, which reach from the simple metadata set Dublin Core up to the idea of an Semantic Web, including the W3C[5] standards RDF (Resource Description Framework) and the web ontology language OWL. But how does one define an object (e.g. a soccer player) in a video scene, maybe characterize it and describe its movement over a certain amount of time?

A search mechanism as offered at the moment by leading internet search engines: image retrieval based on untyped keywords, is probably is not satisfactory for most multimedial applications as we will probably have problems finding the name of the motorcycle Arnold Schwarzenegger rode in Terminator II, where

[4] Heise Online, 9th of august 2005, http://www.heise.de/newsticker/meldung/62612
[5] The World Wide Web Consortium (W3C), www.w3c.org

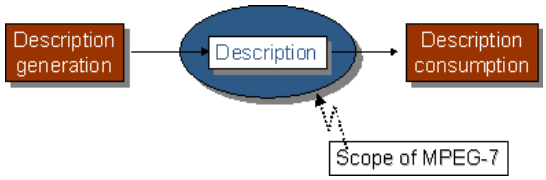

Figure 3: Abstract block diagram of a possible MPEG-7 processing chain - showing the scope of MPEG-7 [14]

the input of our query is provided as an image. TV and film archives, broadcasting channels or digital libraries represent a typical application domain using MPEG-7 which makes audio-visual information as searchable as text. These examples only outline a very brief idea about what MPEG-7 really is able to do and how it can be used - for further examples on typical applications see [6, page 70 ff] - we will try to give a better picture in the following sections by describing the standard more in detail.

3.3 Scope of the Standard

The scope of MPEG-7 is clearly defined. The standard offers a rich set of tools to describe and structure different aspects of multimedia content on several abstraction levels reaching from low-level features (e.g. signal characteristics, color, etc.) to high-level semantic information (while it is normally possible to extract low-level descriptors automatically - the higher the abstraction level the more human interaction and annotation is required). The standard further aims to manage data flexibly and to globalize data resources. Figure 3 illustrates a possible MPEG-7 processing chain including feature extraction (analysis), the description itself, and a search engine (application).

However tools and applications like search engines, filter agents, or any other program making use of the descriptions as well as mechanisms and algorithms for feature extraction are not in the scope of the standard. This has the advantage that interoperability is provided and at the same time the industry is given room for competition and technical improvements. [14]

3.4 MPEG-7 Main Elements

Before explaining the standard's structure in detail we will present the four main normative elements: *Descriptors* (Ds), *Description Schemes* (DSs), *Description Definition Language* (DDL) and System Tools, and outline their functionality. The word "*description*" used in the context of MPEG-7 is defined as a set of instanciated description tools, on the one hand the metadata (Descriptors) and on the other hand its structure and relationship (Description Schemes) To complete this basic picture an interaction of these elements is outlined in Figure 4.

3.4.1 Descriptors

A Descriptor represents a Feature of audio-visual material wheareas a Feature here is understood as "a distinctive characteristic of the data [that] signifies something to somebody" [13, page 80] (e.g. color of an image, genre of a piece of music, etc.). A Descriptor defines the syntax (associated data types, legal values) and the semantics of the Feature representation. An example of a rectangle Ds may contain the x,y values of edges in a coordinate system and the values representing the RGB dominant color of the rectangle - but usually the information contained in a Ds is more complex. (e.g. see [15]) An instantiation of a Descriptor for a certain data set is called Descriptor Value.

3.4.2 Description Scheme

A Description Scheme may consist of Descriptors or other Description Schemes and is responsible for organizing the structure and the semantics of the relationship between its components. The distinction between the concepts of Ds and DSs is that Discriptors are concerned with the representation of Features whereas the Description Scheme handles the structure of Descriptions[6]. [15]

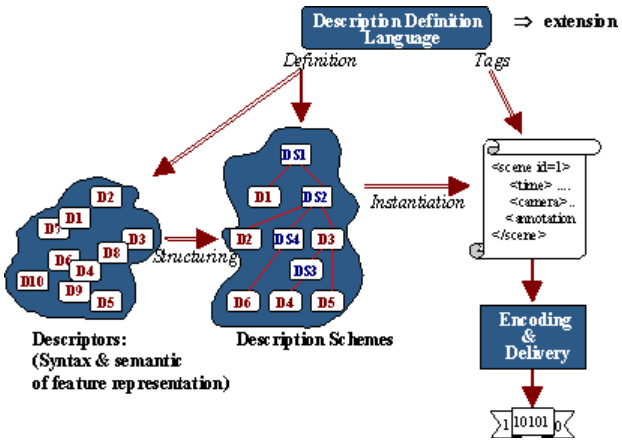

Figure 4: The main MPEG-7 elements [14]

3.4.3 Description Definition Language (DDL)

The Description Definition Language (DDL) is the glue on which MPEG-7 is built on. It allows the creation of new Description Schemes and the manipulation and extension of existing ones and possibly Descriptors. The DDL defines

[6] Definition of Description: A Description consists of a Description Scheme (structure) and the set of Descriptor Values (instantiations) that describe the data. In other words, a Description is an instance of a description schema.

syntactical rules how to combine Ds and DSs as well as temporal, structural and spatial relationships of elements in a Ds and between different DSs. Technically, it was planned to develop a markup language fully defined within MPEG - but with the rising of the eXtensible Markup Language the group decided to change course - and now it is based on the XML Schema with specific extensions.(e.g. support for vectors, matrices and typed references) and therefore has the advantageof being platform and application independent as well as human readable.

The extension of existing Descriptors or Schemes through the DDL has been considered to be very important because - although MPEG-7 defines several hundred Descriptors and Descriptor Schemes in the normative part of the standard, developed to suite most application domains - the praxis has shown that they are not sufficient and that the extension with domain specific characterization is mandatory. [10]

3.4.4 System Tools

MPEG-7 offers a set of system tools to prepare MPEG-7 Descriptions for efficient transport and storage, to manage synchronization between content and descriptions and that offer two different ways of representation. On the one hand side there is the textual format (abbr. TeM), which is defined in terms of the DDL and is XML conform. On the other hand it is possible to create a binary representation, so called BiM which stands for Binary Format for MPEG-7. Just imagine an XML file containing the description for a video - for example decomposed into segments and having low level descriptors like a color histogram automatically extracted for each segment. This file would probably be very verbose and not suitable for consumption in a constrained and streamed environment. [12] The binary format has the advantage providing a more efficient compression ratio than the textual description, flexibility for streaming and efficient access to description elements without having to parse the entire bitstream. Another important purpose of the system tools lies in the management and protection of rights, access or tampering.

3.5 The Structure of the Standard

In this section an overview of the different MPEG-7 components which are currently standardized is presented. The complexity and the magnitude of the ISO/IEC 15398 standard requires to split it up into originally eight and after adding support for profiles lately even ten parts. However note, that some parts (as e.g. Visual - Audio) can be viewed independently while others are highly correlated as only their combination allows to create a structure being capable of capturing and characterizing audio-visual information.

Additional information which is more focused to the industry can be found at the MPEG Industry Forum site[7]. The standard's complete schema definition

[7] MPEG Industry Forum Site, http://www.mpegif.org

including description examples as well as an online syntactical validation service is provided by the National Institute of Standards and Technology[8].

3.5.1 Part1: MPEG-7 Systems

The MPEG-7 System section may logically be divided into four sets of requirements that are dealt with in the standard - while the first three concern topics in the transmission process, the fourth one handles different forms of description representation. We will first present these requirements [10] and finally have a short look at concepts of the "MPEG-7 Terminal Architecture" at the end of this section.

1. Transmission: Descriptions of multimedia content should be able to get broadcasted by a number of various protocols. Further it must be possible to split up descriptions in separate smaller parts to cope with the problem of bandwidth limitations or restricted environments.

2. Synchronization: A definition of temporal relationships between audio-visual components as well as mechanismsfor managing and synchronizing them are requested - and presented in the transport level which we will discuss in more detail.

3. Management: Requirements on the management of an audio-visual data stream cover the field of data processing by an application, containing exact localization of the data, identification of data types, handling of dependencies between different elements, association of media descriptions with their contents and the protection of individual rights.

4. Representation: Descriptions of an MPEG-7 document should be represented by two different formats - on the one side a textual, platform independent, human readable format - on the other side a lossless, highly compressed and efficient binary data transmission format.

To see how these requirements were realized in the MPEG-7 System standard specification view [16, part 1] for further reading.

3.5.1.1 Concepts and Terms of the Terminal Architecture MPEG-7 Systems provides the means to represent coded multimedia content descriptions. An entity that makes use of such coded representations of the multimedia content description is generically referred to as "MPEG-7 terminal," or just "terminal" in short and may correspond to a stand-alone application or be part of an application system. The three main architectural layers it consists of are: application, systems layer (normative), and the delivery layer. An MPEG-7 terminal consumes description streams and outputs a (dynamic) representation called description tree. While the standard is not concerned with any storage or

[8] National Institute of Standards and Technology (NIST), http://m7itb.nist.gov/M7Validation.html

transmission of media (it has no specific delivery protocols specified), or the way the application processes the current description - it does though make assumptions about the delivery layer (e.g. requires functionalities for synchronization, framing and multiplexing of description streams with other streams) [14] In that relation it is important to mention the concepts of "Schema Streams", "Description Streams" and "Access Units".

- "Schema Streams" define the structure of an MPEG-7 description and are only necessary when an application does not have information about a schema used for the description generation.
- "Description Streams" contain either a complete MPEG-7 description or fractions of it
- "Access Units" are carried by the delivery layer and stand for the smallest data entity to which timing information can be attributed

3.5.2 Part2: Description Definition Language

While an introduction as well as a presentation of the schema language DDL and its main tasks were already presented in section 3.4.3, we will primarily concentrate on the third point of the standard document - the XML Schema specific extensions - here.

3.5.2.1 Short Summary on XML Schema XML Schema[9], which was approved as a W3C recommendation in May 2001, is a complex language to define classes of XML documents and to restrict their content, regarding elements and attributes - to values as well as to data types which may be used in these classes. XML Schema Structure, the first part of the two W3C Schema specifications parts, defines the composition of an XML Schema as well as its components and vocabulary of markup constructs. The main components which may get defined by a Schema are simpleType or complexType, as well as Attribute (groups) and Element declarations.

complexType vs. simpleType: W3C XML Schema considers elements that have a simple content model and no attributes as "simpleTypes", while all the other elements (such as simple content with attributes and other content models) are "complexTypes" In other words, elements that can only have text nodes and do not accept any child elements or attributes, are considered to be simpleTypes - in all other cases complexTypes.

The second part of the XML Schema recommendation deals with data types in a Schema and defines limited facilities for applying data types to document content (similar to DTDs which assigns types to elements and attributes) [17]

[9] XML Schema, W3C, http://www.w3.org/XML/Schema

3.5.2.2 XML Schema specific Extensions To satisfy specific MPEG-7 requirements, like the field of low level audio-visual content description, a need for more effective data types like Matrix and Array, as well as time data types was given. The standard distinguishes between two different kinds of extensions, *MPEG-7 structural-* and *MPEG-7 data type extensions* that have been added to the XML Schema Language specification: [16, part 2]

1. structural extensions: Contain multi-dimensional matrices and one-dimensional arrays using the data type "list". By providing a new facet "mpeg7:dimension" the dimension of these elements can be handed over at the time of schema definition by taking a positive integer value, or marking as "unbounded" - that indicates variable size. In latter case either a derived simpleType redefines this unbounded dimension or the attribute "mpeg7:dim" is provided in conjunction to enable the size of an array or matrix.

2. data type extension: The basicTimePoint and basicDuration primitive time data types have been provided within the DDL. First one specifies a time point , appropriate to the Gregorian calendar specification (dates, day time and time zone) - basicDuration depicts a period according to days and time of day - both by using a subset of the ISO 8601 format.

3.5.3 Part3: Visual

Part three of the ISO/IEC 15938-3 specifies standardized methods for the description of visual content only - that are still images, videos and 3D models. The MPEG-7 Visual Descriptors are grouped in five categories by the *features of color, texture, shape, motion* and *localization* as well as a domain specific face recognition tool. Further this part also offers supporting tools - so called *Basic Structures* - that provide efficient representation of visual features on grids, time series and multiple view on 3D objects.

Regarding the structure of the diplomarbeit, this part of the MPEG-7 standard is dealt with in section: 3.6

3.5.4 Part4: Audio

MPEG-7 part four standardizes audio description tools (including both Descripors and Descriptor Schemes) which enable the characterization of audio content (containing sound, music and speech). Most tools in this part of the standard are based on features measuring similarities in sounds. It is able to distinguish two different levels of audio description tools - on the one hand the generic ones, including a group of *low-level descriptors* for signal spectral, parametric and temporal features, called *MPEG-7 Audio Framework* - on the other hand the *high level description tools* enabling sound recognition, indexing, spoken content and query-by-humming applications, for example.

3.5.4.1 Low-Level Audio Descriptors Generally speaking there are two possible ways of describing low-level audio characteristics. The first one is to

divide an audio sample e.g. a piece of music into regular time intervals and to extract a representative value from each part, while the second way is to use segments which describe regions of similarity/dissimilarity in sound. Both possibilities are represented by MPEG-7 Audio Descriptors. All in all the low-level MPEG-7 Audio Framework consists of seventeen Descriptors, representing spectral and temporal features. They play an important role in describing audio material and therefore provide a basis for the construction of higher-level audio applications. The framework is grouped as follows:

1. *Basic*: The basic audio Descriptors, AudioWaveform and AudioPower contain scalar values which describe a certain temporal period of a signal. Due to the general construction of these tools they are applicable for all kinds of signals. While the first one describes an audio waveform characteristic (minimum/maximum), the latter one is useful as a quick summary of a signal by describing temporally smoothed instantaneous power.

2. *Basic Spectral*: The four basic spectral Descriptors which are used for the characterization of frequency parameters (e.g. short-term power spectrum, shape of the power spectrum, etc.) may be derived from a single time-frequency analysis of an audio signal.

3. Signal Parameters: AduioFundamentalFrequency and AudioHarmonicity are the two signal parameters which address the field of periodic or quasi-periodic signals. While the first one describes the fundamental frequency of an audio signal the second one allows the distinction of sounds by their harmonic spectrum.(e.g. harmonic: music, inharmonic: bell-like sounds and non-harmonic: noise)

4. *Timbral Temporal*: These Descriptors are able to depict temporal characteristics of sound segments and therefore are especially useful for describing musical timbre - in other words a characteristic tone quality which is independent of pitch and loudness. The first Descriptor in this group, the LogAttackTime, captures the period of time it takes for a signal to rise from silence to its maximum amplitude and by that is suitable to distinguish between a sudden and a smooth sound - the second one, the TemporalCentroid, indicates where in time the energy of a signal is concentrated (e.g. difference between a decaying piano note and a sustained organ note by identical lengths and attacks of the notes.)

5. *Timbral Spectral:* There is a total number of five low-level timbral spectral audio Descriptors which represent spectral features in a linear-frequency space. These tools especially fit domains of harmony perception.

6. *Spectral Basis*: The two spectral basis Descriptors represent low-dimensional projections of a high-dimensional spectral space to support compactness and recognition. Together they are able to create a compact representation on the independent subspaces of a spectogram and thereby have the advantage to provide more salience (e.g. individual instruments) and

structure. They are normally used in sound classification and indexing tools.

The *Silence Segment* completes the picture of the MPEG-7 Audio Framework. This Descriptor is a very simple but extremely important and effective tool which attaches the simple semantic of "silence" - in this context is understood as "no significant sound" - to an audio segment. It may be used for further segmentation of audio streams as well as in the function of an indicator for not processing certain segments. [14]

3.5.4.2 High-Level Audio Description Tools (Ds and DSs) While the low-level audio Descriptors are able to represent sound on a very low abstraction level without any domain specific knowledge, MPEG-7 Audio includes a set of high-level Descriptors and Description Schemes which cover a wide range of application areas - to provide description richness. The five types of standardized audio description tools are: *audio signature,* which summarizes a set of low-level features with the goal to provide unique content identification for robust automatic identification of audio signals, *musical instrument timbre*, which describes perceptual features of instrument sounds with a reduced set of Descriptors, *melody description*, that includes a rich representation for monophonic melodic information to facilitate efficient, robust and expressive melodic similarity matching, *general sound recognition and indexing* as well as *spoken content* - the latter two are good examples of how to use and integrate the low-level framework. For detailed reading see [16, part 4].

3.5.5 Part5: Multimedia description schemes

After having presented an overview on the tools - on the one side the MPEG-7 Description Definition Language - with which it is possible (amongst other things) to define additional application specific description tools - and on the other side the standardized MPEG-7 media specific audio (only) and visual (only) descriptors, we will now have a look at the core of the standard which combines descriptions and description schemes for generic as well as for multimedia features - the *Multimedia Description Schemes (MDS)*. While parts three and four of the standard are settled on a lower abstraction level (e.g. size, texture, color, etc. for video and mood, tempo (changes), or positions in sound spaces for audio), part five - the MDS - would allow a semantic representation of content e.g. similar to "A soccer video with the goalkeeper on the right side and the ball with the player on the left". Further it also offers tools for describing the creation and production process of content, usage and context. [12]

The Multimedia Description Schemes are grouped in different classes, according to their functionality (see Figure 5): *Basic Elements*, *Schema Tools*, *Content Description*, *Content Management*, *Content Organization*, *Navigation and Access*, *User Interaction*. A general survey on their functionality is given in the following sections.

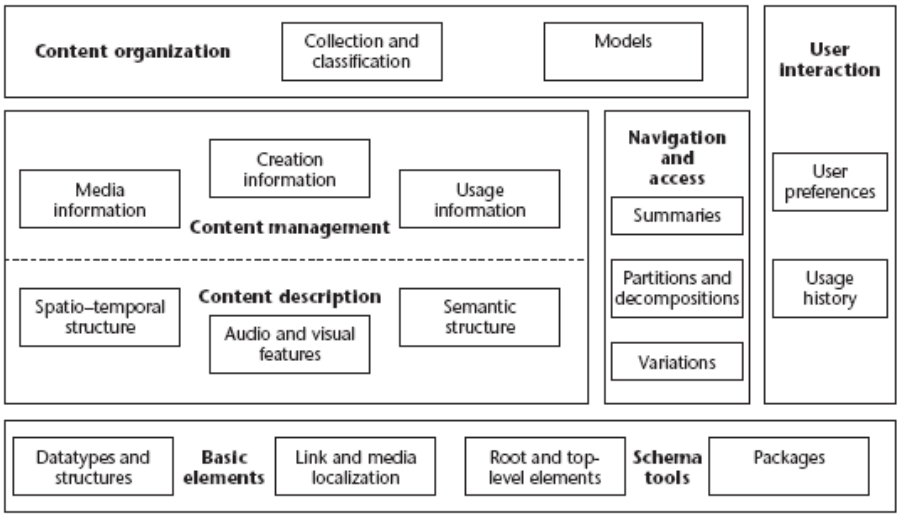

Figure 5: MPEG-7 Description Tools Overview [13]

3.5.5.1 Basic Elements and Schema Tools To create a MPEG-7 description *Schema Tools* which aid in formatting, packaging and annotating are required. First of all every MPEG-7 description must start with a root element (<Mpeg7>) including a description metadata header, which indicates either the presence of a semantically complete description (<Description>) or of a description unit (<DescriptionUnit>). While the latter one enables the transmission of description fragments, a complete description provides a standalone representation of an audio visual content for a certain application. In the case of a complete description, a *MPEG-7 top level element* must follow the root element - they are organized in three groups:

- Content Management: This top level-element describes management related aspects of content as e.g. creation, usage or media.
- Content Entity: indicates which audio-visual entities are described, such as image, video, audio, audio-visual, multimedia or even ink content.
- Content Abstraction: describe abstractions of the content, like models, semantics or summaries.

Otherwise - using the description unit - the root element may be followed by an arbitrary instance of a MPEG-7 Descriptor or Descriptor Scheme. Finally the *Package Tools* allow the organization and arrangement of MPEG-7 Ds and DSs into packages with the purpose of customizing description tools for applications (e.g. search engines) or users.

On the one side the *Basic Elements* offer a set of extended data types for audio and visual elements, string data types (e.g. country or currency codes following ISO standards) as well as mathematical structures like matrices and vectors, which are needed by DSs to capture audio-visual features. On the other side it also has mechanisms for linking and locating (e.g. time and media locators), it provides further basic description tools for, people, places, time (e.g. time points, duration) or controlled vocabulary, and so forth. Although MPEG-7 never aimed to standardize textual feature characterization tools, they proved to be necessary to provide a complete set of description tools - the Basic Elements therefore offer textual annotations structures for: Free Text, Structured (by including specific fields corresponding to the questions "Who? What object? What action? Where? When? Why? and How?), DependencyStructure (considering dependency between grammatical elements as e.g. the relation between a verb and a subject) and Keyword. [18], [14]

3.5.5.2 Content Management The description of content management (CM) deals with information related to content though it is independent of what actually is represented in the content itself. With the standardized CM description tools it is possible to characterize a complete life cycle of content reaching from creation and production to media formats and transcoding to finally consumption. The following five top-level elements are available in a description: [18]

1. *Media Description*: This top level element depicts the physically media information itself by using a Scheme which encapsulates different content coding aspects, like media identification (to identify multimedia content independently from its available instances), media format (e.g. file format and coding parameters) or quality rating. Because a certain content - described by MPEG-7 - may have multiple instances or in addition is even available in different modalities (e.g. audio or audio-visual), formats - encoded by different Coding Schemes - the idea of *media profiles* is introduced. These media profiles refer to the presented parameters (formats, transcoding hints, etc.) of each variation of the content and further also provide locators and identifiers to all available instances There is always one master profile, that corresponds to the originally created or recorded profile.

2. *Creation Description*: contains tools for describing creation and production information, such as author, creation time, date, title, characters or where related material is available, as well as typical user annotation classifications like target audience, genre or rating.

3. *Usage Description*: Information regarding usage rights, usage information (as availability and audience), or financial information are contained within this top-level element. MPEG-7 does not explicitly deal with the handling of rights, but rather gives access to the rights holders relevant information.

The last two content management top-level elements are *Classification Scheme* and *User Description.* While the first one offers a term-based description, the latter one describes user related information of the content such as the user himself, his preferences (e.g. for personalized access or filtering) and his history which may help to refine his preferences.

3.5.5.3 Content Description The standard offers Description Schemes to annotate perceptible information, including structural and semantical aspects of audio-visual content. While the structural tools describe a content in terms of spatio-termporal segments, the latter one uses "real-world" semantics and conceptual notations, as objects or events, to do this. Both description methods may be combined by using a set of cross-linking mechanisms.

Structural Aspects The core element of providing structural aspects in an description of audio visual content is the Segment DS, which represents spatiotemporal relationships between *segments* (i.e. a section of an audio-visual content item) in form of a hierarchical segment tree and therefore is able to describe the result of a spatial, temporal or spatiotemporal partitioning of the content. The Segment Descriptor Scheme itself is an abstract class (in the sense of object-oriented programming) which contains elements and attributes that are common to all of its nine - specialized for audio, visual, still or moving regions - subclasses. A segment may be subdivided into parts of different types (e.g. a video segment decomposed in moving regions, which again split up in still regions), and these decompositions may also leave gaps and overlaps between their sub-sections. Finally each segment may then be characterized separately by using Descriptors for color, texture, shape, motion, etc. - although it is important to note that every Descriptor or DS which is attached to a segment is global to the union of the connected components building the segment.

Figure 6 shows an example of describing an image using several StillRegion decompositions. While the initial region SR1 describes amongst other things data as creation (e.g. title, author), usage or media information as well as a textual annotation (e.g. summarizing the image content) and a color histogram, we can further dcompose the image into individual regions, building a segment tree. For every decomposition step it is indicated if gaps/overlaps are allowed and which type of feature is instantiated. It is not necessary to repeat the creation or usage information in the tree hierarchy, since the child segments are assumed to inherit their parents values. [14]

Although this hierarchical tree structure is very adequate for most applications, by offering efficient access, retrieval and scalable descriptions, there are circumstances where it is inappropriate. In such a case the SegmentationRelation DS, which defines a simple set of nodes (segments) and edges (relationship between two nodes) to describe composition and positional relationships from a segment to another one, may be used for example to annotate a soccer player

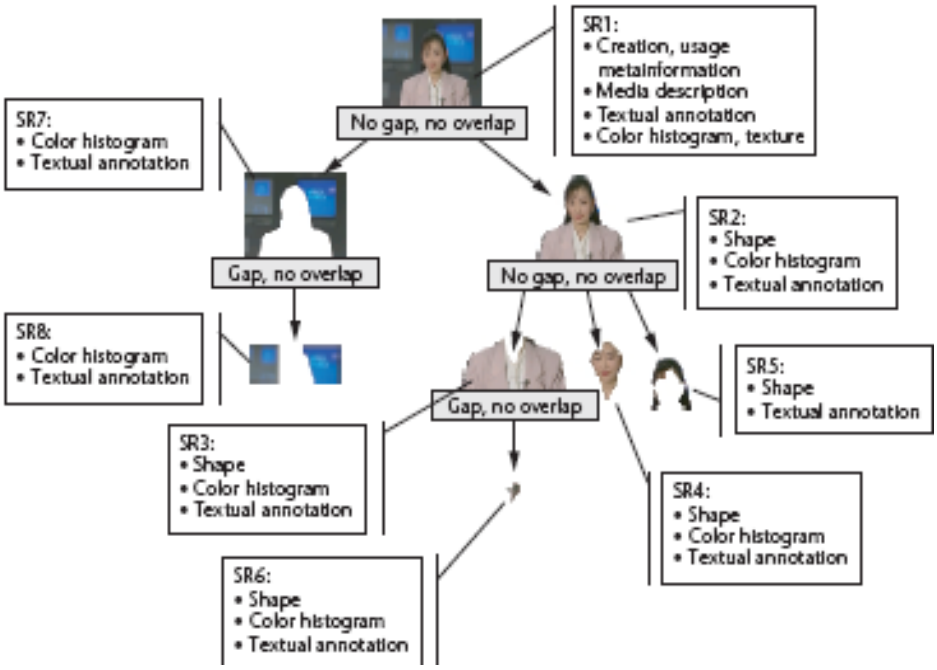

Figure 6: Examples of an Image Description using Still Regions [18]

(segment) on multiple video frames and his position "left of" (edge) the ball (segment). However it is important to note, that the SegmentRelation DS represents mainly structural information - the only explicit semantics provided here come from the textual annotation by using keywords such as ball or player.

Conceptual Apsects While the previous DSs dealt with structural aspects of content description this section provides structures for its semantical classification. The main Descriptor Scheme is the SemanticBase DS which is able to encapsulate the description of a narrative world and its elements by using - instead of decomposing segments - *objects, agents, events, concepts, places* and *time* to build an abstraction. A narrative world in this context is understood as a "reality" in which a description makes sense. There are several spezialized DSs derived from the generic SematnicBase - an example showing a conceptual description using SemanticTime, -Place as well as an Object and Event Description Scheme is given in Figure 7.

As in the case of Segment DSs a graphical representation of a tree - nodes containing semantic information and edges specifying the relationships (making use of the Semantic Relation DSs) between them - may be used to organize the conceptual aspects of a description. The shown narrative world involves the objects "piano", "Tom Daniels", "Tom's tutor" which belong to the class of

Agents, and the abstract notion of "musicians". The event "to play" is characterized by a semantic time ("7-8 p.m., 14 Oct. 1998") and place description ("Carnegie Hall"). [14]

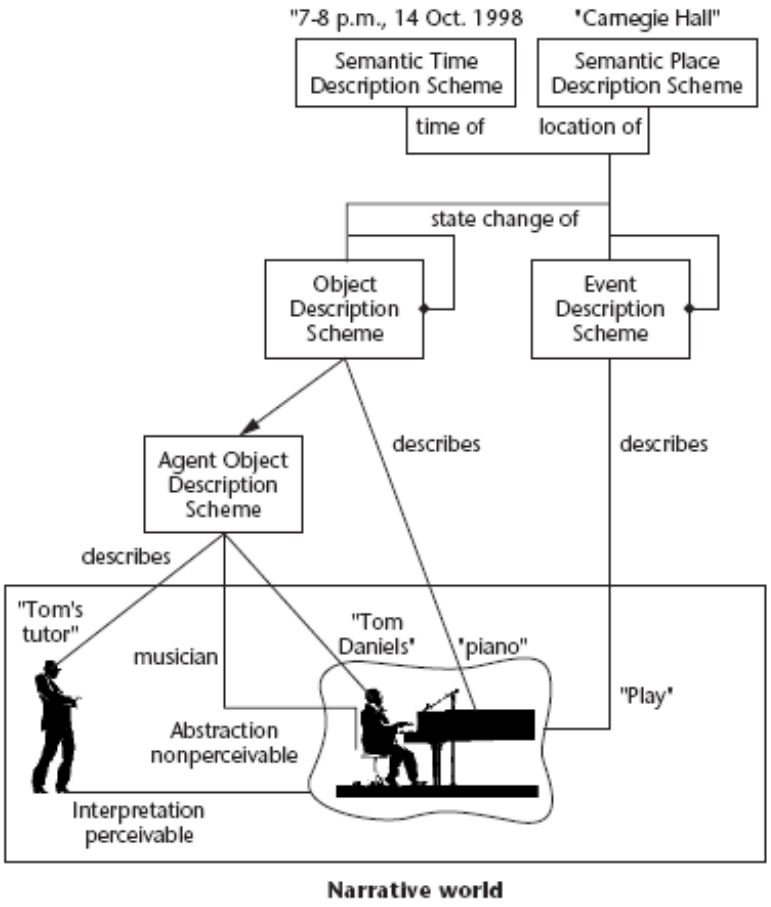

Figure 7: Example showing a Conceptual Description [18]

3.5.5.4 Content Organization The group of MPEG-7 content organization tools contains DSs for arranging and organizing *collections* as well as describing their common properties. The Collection Structure DS clusters groups of elements having similar content, segments, events (e.g. similar events in a soccer game) and objects. It further describes the relationships of its contained components such as the degree of similarity (e.g. of images in the cluster) as well as cross collection relationships.

A collection may additionally be described by attaching *models* or statistical

information to characterize the attributes of collection members. The Probability Model DS provides fundamental DSs for specifying statistical functions and probability structures and for example may be used to represent audio-visual data and classes of Descriptors using statistical approximation.

3.5.5.5 Navigation and Access The standard provides Description Schemes enabling efficient browsing, navigation and access of audio-visual content by describing *summaries*, *variations* as well as *views* and *partitions*. [14]

1. *Summaries*: The Summary DSs provide compact and expressive summaries of AV material which facilitate discovery and enable fast and effective browsing. The data may either be navigated hierarchical, which describes the content in different levels of temporal detail (i.e. coarse at the root (e.g. by using a single frame representation) and detailed towards the child elements), or sequential for example composing a slide-show. HighlightSegment DSs contain locators to certain parts of an audio-visual content providing access to associated key-frames, and textual annotations of characterizing these frames. It further is possible to create various summaries corresponding to different themes of a single AV content to present alternative views (e.g. of different granularity) on the data.

2. *Partitions and Decompositions*: description tools describe decompositions of audio and video signals in time, space and frequency to provide different views on their content. Generally these views correspond to frequency subbands or low-resolution views. These DSs are important for multiresolution access and progressive retrieval.

3. *Variations of Content*: Variations of AV content relate to compressed, low resolution and scaled versions, summaries, different languages and modalities (as video, audio, image, etc.). The Variation DS describes these variations of pre existing views, it gives information about the quality compared to the original and captures the type of change (such as color reduction, rate reduction, summary, and so on). Therefore one of their most important functionalities is to provide an application with the most suitable version of an audio-visual content, dependent on its resources, network and bandwidth limitations as well as user preferences.

3.5.5.6 User Interaction Finally the last set of MPEG-7 MDS offers three DSs for defining topics concerning user interaction. The *UserInteraction DS* characterizes user preferences related to the consumption of audio-visual material - allowing for example the matching of user preferences and MPEG-7 content descriptions, to personalize multimedia content access. With the *UserPreference DS* one is able to describe user preferences in terms of favored types of content and modes of browsing (including dependencies in time and place). Further it allows the weighting of preferences by their relative importance and deals with privacy aspects. The *UsageHistory* DS describes a history of actions taken by a

user in the past. This information can be exchanged between customers, their agents or devices to determine tendencies used for adaptational purposes. [14]

3.5.6 Part 6: Reference Software

The eXperimentation Model is the reference software implementation which was used for proof of conept in the standardization process. It contains a simulation platform for the MPEG-7 Descriptors, Description Schemes, Coding Schemes as well as for the Description Definitoin Language. Besides these normative elements the software makes use of several non-normative components, essentially for executing procedural code and reading binary data. The data structures together with the procedural code are called XM applications. They are divided into server applications (for extraction) and client ones (for searching, filtering, etc.). A detailed description of the XM framework including the usage of its applications for MPEG-7 feature extraction as well as its main limitations is given in chapter 4.

3.5.7 Parts 7 and 8: Conformance Testing; Extraction and use of MPEG-7 Descriptors

Offering a mighty framework for multimedia asset description alone is not sufficient - to complete the creation/consumption circle, possibilities for validating syntactical conformity of MPEG-7 implementations against the standard specifications are needed. Part seven of the Multimedia Content Description Interface includes guidelines as well as procedures for testing conformance of MPEG-7 implementations and mainly addresses two important areas. The first one deals with *conformance testing of descriptions*, which is split up in two stages system testing (check decoding results of binary descriptions against its XML representation) and DDL testing (check if an XML document is well-formed and valid against the schema definitions specified in parts 1-5 of the standard). The second one is *testing of terminals* which compares results of descriptions that are processed using a reference terminal against results which arise from using a test terminal. Note that no semantical examination of descriptions (e.g. that the field "author" really contains a true name of an individual) is provided. For information regarding the MPEG-7 Interoperability Test Bed (M7ITB), which implements MPEG-7 conformance validation according to the schema specifications, see [10].

Part eight, "Extraction and use of MPEG-7 Descriptions" contains informative material in form of technical reports and illustrates the extraction and use of description tools by examples. This part is designed to be technically compliant to part five (Multimedia Description Schemes) and part three (Visual) of the standard and therefore requires knowledge of their corresponding technical specifications. [14]

[10] MPEG-7 validation service, http://m7itb.nist.gov/M7Validation.html

3.5.8 Parts 9 and 10: Profiles and Schema Definition

Part nine "Profiles and Levels" and part ten "Schema Definition" of MPEG-7 were not included in the original standard specification published in 2002 and subsequently got approved as international standard 15938-9 and 15938-10 in 2005. The significant keyword characterizing *MPEG-7 Profiles and Levels* is "complexity". History has already shown for previous MPEG standards that profiling of tools - defined by the standard - plays an important role to allow deployment of the technology with reasonable costs and complexity. But unlike previous products, profiles and levels may be defined in MPEG-7 across all parts of the standard (current profiles especially concentrate on Systems (part 1), DDL (part 2), Visual (part 3), Audio (part 4) and MDS (part5)) and therefore had the need to be defined in an own part. While a profile in this context is understood as a subset of MPEG-7 tools which provides certain functionalities for one or more classes of applications, a level defines a set of constraints on a profile to limit its complexity. At the moment 15938-9 only specifies so called "description profiles" defining subsets of description tools - further work may define profiles of other types as e.g. System Profiles, limiting description size or depth.

Schema Definition, part 10 of the standard, assembles complete MPEG-7 schemes, collected from different standards, corrigenda and amendments, and thereby contains a single schema across different versions. Namespace versioning is also dealt with in this part. [14]

3.6 MPEG-7 Visual Descriptors

> "MPEG-7 provides the world's richest set of audio-visual descriptions. These descriptions are based on catalogue (e.g., title, creator, rights), semantic (e.g., the who, what, when, where information about objects and events) and structural [...] features of the AV content and leverages on AV data representation defined by MPEG-1, 2 and 4." [19, page 2]

Part three of the ISO/IEC 15938-3 specifies standardized methods for the description of visual content - that are still images, videos and 3D models. The MPEG-7 Visual Descriptors are grouped in five categories by the *features of color, texture, shape, motion* and *localization* as well as a domain specific face recognition tool. Further this part also offers supporting tools - so called *Basic Structures* - that provide efficient representation of visual features on grids, time series and multiple view on 3D objects. All of these tools are defined in the standard paper in syntax of the DDL as well as in an binary representation format. Note, that not all of them actually are Descriptors in the sense that they extract properties of media content - but some of them just provide structures for descriptor aggregation and localization. With respect of the importance of these tools to the practical work of the diplomarbeit we will present the idea and information behind these features in detail and give an overview on the most important facets if required. For a definition of the Descriptors DDL

representation syntax as well as for the interpretation and usage of certain bin values see [16, part 3] for further reading. The following information is provided by: [16, part 3] and [14]

3.6.1 Color Feature Descriptors

The color description tools include four descriptors that represent different aspects of color: representative colors (*DominantColor*), color distribution (*ScalableColor*) and spatial distribution of colors (*ColorLayout* and *ColorStructure*) and may be extracted from images of arbitrary shape. To complete this picture of Color Descriptors we finally want to mention the MPEG-7 Visual Feature supporting tools: *ColorSpace* and *ColorQuantization* which are used in DominantColor, and *GoFGoPC* an extension of ScalableColor for a group of frames.

1. *ColorSpace*: expresses in which of the supported color spaces - RGB, YCbCr, HSV, HMMD, linear transformation matrix with reference to the colors red, green, blue as well as the monochrome color space - the Descriptor lies. Most of the other color features make use of this Descriptor.

2. *ColorQuantization*: specifies the uniform quantization of a color space.

3. *DominantColor*: fits the targets for content based retrieval of either the whole image or arbitrary defined shapes by defining a set of characterizing dominant colors in that area. It contains fields (among others) representing the percentage of pixels having that associated color in the specified region, as well as a spatial coherency per dominant color attribute, a kind of quality testimony which is presented in Figure 8. The maximum number of dominant colors in a region is limited to eight.

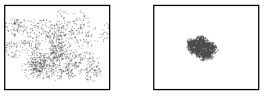

Figure 8: Examples of low (a) and high (b) spatial coherency of color [16, part 3]

   This descriptor is most suitable for representing local features where a small amount of colors is enough to characterize the provided information in the region of interest but it also is applicable for whole images for example, flag or color trademark images.

4. *ScalableColor*: This descriptor depicts a color histogram in the HSV color space, encoded by a Haar transformation[11] (which allows the sorting of data by frequency) and is used to find and compare images by their color characteristics.

[11] Haar Wavelet Transformation, Wikipedia, http://de.wikipedia.org/wiki/Haar-Wavelet

5. *ColorLayout*: Is used for high-speed image retrieval and browsing using image-to-image and sequence-to-sequence matching - and even sketch-to-image matching, by specifying a spatial distribution of colors. This Descriptor makes use of the DCT[12] the discrete cosine transformation to compute its YDC, CbDC, CrDC,YAC,CbAC and CrAC coefficients. This compact representation of color layout information allows visual signal matching functionality combined with high retrieval efficiency at very small computational costs and therefore even is utalizable on mobile terminal applications with strictly limited hardware and software restrictions.

6. *ColorStructure*: This descriptor is similar to the one of a color histogram in the meaning of describing color content - but further also pays respect to the structure of its content. Unlike the color histogram it uses a structuring element, which is composed of 8x8 connected pixels and instead of characterizing the relative frequency of individual image samples it characterizes the frequency of these structuring elements. In that way it is possible (as presented in Figure 9) to distinguish between two images having an identical amount of the same color, though the structure of the groups of pixels having that amount is different. The main areas of usage are image-to-image comparison in domains of still-image retrieval - even in arbitrary shaped and possibly non-connected regions.

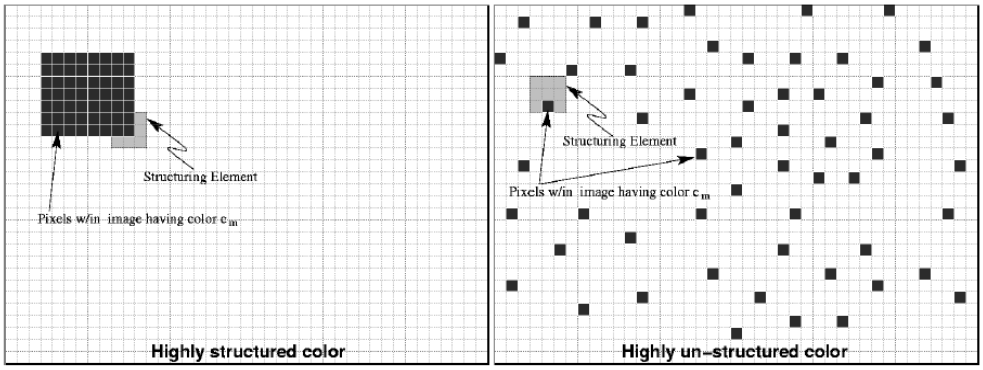

Figure 9: Examples of a highly structured (a) and unstructured (b) color [16, part 3]

7. *GoF/GoPColor*: offers a structure to apply the scalable color feature to a collection of (similar) images or video frames. It specifies attributes like "aggregation" indicating how the individual frame or pictures histograms are assembled to a combined representation in three different possibilities: average, intersection or median histogram each having advantages and

[12] DCT, Cardiff School of Computer Science, http://www.cs.cf.ac.uk/Dave/Multimedia/node231.html

disadvantages. The latter for example is able to eliminate effects like lighting changes, text overlays which the average histogram is vulnerable to.

3.6.2 Texture Feature Descriptors

Textures in MPEG-7 are understood in the meaning of patterns which are used to describe surface characteristics of objects. Similarity of patterns here is measured in terms of intensity of certain colors in an image or by the degree of uniformity of a structure compared to a certain environmental point. The three standardized Visual Texture Descriptors *TextureBrowsing*, *HomogeneousTexture* and *EdgeHistogram* are very important tools for similarity retrieval and browsing. Objects having contrary textures for example are on the one hand a highly regular brick wall with a smooth surface and a calm pattern structure - on the other hand the irregular bark of a tree.

1. *HomogeneousTexture*: characterizes the region texture of an image using the energy (and its deviation) in a set of frequency channels. An image in this relation can be seen as a mosaic of homogeneous textures - having its regions tagged by the frequency channels they lie in to build an index of the image data. The 30 frequency channels we distinguish are created by partitioning the frequency space with equal angles (30 degrees in the angular direction and with octave division in the radial direction) and numbering from left to right and from higher to lower frequency bands. The facet containing the energies computed from the feature channels is a 1-d array called "Energy" and provides - besides "Average" and "StandardDeviation", which specify the average image pixel intensity and its deviation, - the descriptors characteristic information. Good examples representing a homogeneous texture pattern are parking lots when viewed from a distance or agricultural areas.

2. *TextureBrowsing*: This descriptor has the function to represent a perceptual characterization of a texture, in terms of regularity (irregular, slightly regular, regular, highly regular), coarseness (fine, medium, coarse, very coarse) and directionality (the dominant direction(s) characterizing the texture alignment). Figure 10 shows examples of objects and their regularity gradations. This classification, which is similar to a human one is useful for browsing applications.

3. *EdgeHistogram*: Edges play an important role for image perception. The EdgeHistogram tool distinguishes five types of edges in local image regions, four directional (vertical, horizontal, 45 degree, 135 degree) and a non-directional one. This information is provided for every sub-image, which is defined by dividing the image space into 16 non-overlapping parts. The Descriptor is able to retrieve images with similar semantics.

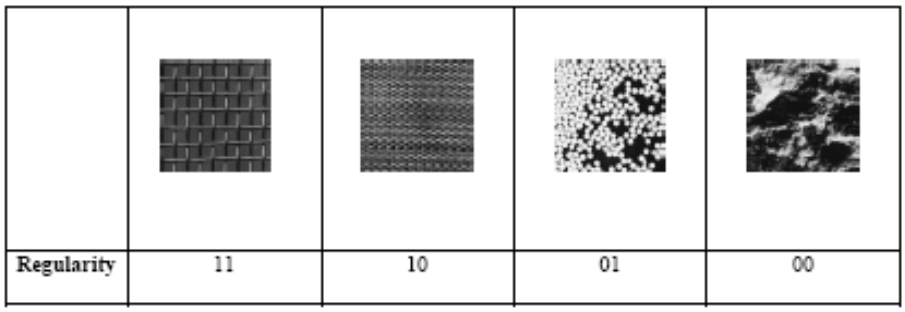

Figure 10: Examples of Regularity: highly regular (a), regular (b), slightly regular (c) and irregular (d) [16, part 3]

3.6.3 Shape Feature Descriptors

A shape feature in this meaning relates to the arrangement of pixels belonging to an object. This provided toolset either characterizes shape features of 2D objects or regions as the *RegionShape* and the *ContourShape* Descriptors do, or 3D objects like the *Shape3D*. While RegionShape captures the distribution of all pixels in a certain region, the ContourShape tool expresses contour properties of an object. Typical examples on the usage of these Descriptors are the localization of brand names or company logos in an image.

1. *RegionShape*: As the Descriptor makes use of all pixels forming the shape within a frame, it is possible to define shapes consisting of either a single region as shown in Figure 11 (a) and (b), a set of regions but also even more complex objects of disjoint regions or holes illustrated for example in Figure 11 (c), (d) and (e). The RegionShape Descriptor utilizes a set of ART (Angular Radial Transform) coefficients, which is a 2-d complex transform. It is denoted by fast extraction time, low computational complexity and even shows robustness to minor deformations along the boundary of an object.

Figure 11: Examples of different shapes [16, part 3]

2. *ContourShape*: The ContourShape feature captures a closed contour of a two-dimensional object in an image or video scene. It uses the so called CSS (Curvature Scale-Space) representation to extract perceptually meaningful features of the shape. To create such an CSS description, "Nsamples" equidistant points are selected on the contour which is then gradually smoothened by a low-pass filter. As a result the contour evolves and its concave parts gradually flatten out. This kind of representation is very suitable to capture characteristic features of the shape, enabling similarity-based retrieval. It has the advantage of reflecting properties of the perception of human visual systems, it is compact and offers good generalization. Further it is robust to partial occlusion of shapes and to non-rigid motion (like changes of camera perspective)

3. *Shape3D*: Generally it is able to describe a 3D object with carefully selected two-dimensional features - providing different views on it, but most of the time three-dimensional information is represented by polygonal mesh models (MPEG-4 deals with efficient 3D mesh model coding). A polygonal mesh[13] specifies a set of points which represent vertices of polygonal faces, to approximate the surface shape of an object. The Shape3D Descriptor provides content-based access to that kind of information (targeted e.g. for search, retrieval and browsing applications of 3D model databases) by providing an intrinsic shape description of 3D mesh models.

3.6.4 Motion Feature Descriptors

The MPEG-7 Visual Motion Descriptors include tools which characterize essential aspects of motion very effectively. There are four standardized Descriptors: *CameraMotion*, which offers a set of basic camera operations like panning and tilting, *MotionTrajectory*, characterizes the movement of key points, *ParametricMotion* tracks the evolution of an arbitrary shaped object or region over time and finally *MotionActivity* which is able to capture the pace of motion in a sequence.

1. *CameraMotion*: The Descriptor is based on the three-dimensional camera motion parameter information that supports the following well known basic camera operations:

    - fixed
    - panning (right, left), tilting (up, down) and rolling
    - tracking (left, right), booming (up, down), zooming (in, out) and dollying (forward, backward)

[13] Polygonal Meshes, Ken Power, Institute of Technology Carlow, http://glasnost.itcarlow.ie/~powerk/Graphics/Notes/node5.html

This metadata can automatically be extracted or produced by capturing devices and may get used to determine the building blocks of the CameraMotion Descriptor in two ways: "single" or "mixed". A building block is composed by a start time, the duration, the speed of the induced image motion, the fraction of time of its duration compared to a given temporal window and the FOE (focus-of-expansion). The Descriptor presents the union of these building blocks but while in the mixture mode global information about the camera motion parameters is captured and detailed temporal information is ignored by jointly describing multiple motion types - the non-mixture mode captures pure motion types and their union with a certain time interval. In the latter case it is therefore possible that a time window of an elementary segment overlaps with another one.

2. *MotionTrajectory*: The MotionTrajectory feature is a simple but high level feature. It defines the localization of one representative point of an object in time and space and essentially is defined as a list of keypoints (2D or 3D coordinates and time) along with interpolating functions describing the path of an object between keypoints. This Descriptor is able to suit a variety of tasks like the categorization of objects by speed, by behavior or other high level motion properties and therefore is useful e.g. for content-based retrieval in object-oriented visual databases or for trajectory identification in contexts with a priori knowledge (e.g. triggering alarms, when crossing forbidden areas) but beyond that it also allows enhanced data interaction and manipulation.

3. *ParametricMotion*: specifies the motion of objects in a video sequence. in the meaning of describing the evolution of arbitrary shaped regions over time. This two-dimensional geometric transform may be expressed in the ParametricMotion Descriptor by following 2D parametric models:

   - affine models, including translational, rotation/scaling and combinations of them
   - perspective models, which make it possible to regard deformations
   - quadratic models, allowing a more complex description of movements.

   These models are associated with objects, that are defined as group of pixels in an image for a certain amount of time. Due to the fact, that the Descriptor should be associated with a spatiotemporal region, temporal and spatial information is provided along with the motion parameters - and in that way leading to a very compact and efficient description.

4. *MotionActivity*: This Descriptor is able to capture the intensity of motion or the pace of an action in a video scene and thereby contains information about impressions a user has who is watching it (for example a user is able to distinguish between a fast paced sequence, like a tennis rally, or a slow action shot, like an interview scene). The MotionActivity tool therefore makes use of the following five attributes:

- intensity of activity: High intensity of activity, like e.g. found in a soccer match, indicates high activity and vice versa.
- direction of activity: Often it is possible to identify a dominant direction, amongst several objects with different directions, in a video sequence which is expressed by this facet.
- spatial distribution of activity: depicts if the activity of a scene is spread to several areas or concentrated in one region (e.g. bird view of a soccer game vs. news speaker)
- spatial localization of activity: to categorize spatial distribution of motion over the complete duration of a video segment or shot (e.g. having its motion on the left side of the frame)
- temporal distribution of activity: expresses whether an activity is sustained throughout the complete duration of a sequence or not.

3.6.5 Localization Feature Descriptors

The features contained in the MPEG-7 Visual Description tools are able to specify regions of interest in either a spatial domain, like the *RegionLocator* does, or in a spatio-temporal one, as the *SpatioTemporalLocator* is able to do. While the RegionLocator enables the localization of regions in an image or a frame by specifying them with a scalable box or a polygon, the SpatioTemporalLocator describes moving object regions (spatially connected as well as non-connected) in a predefined set of frames by identifying the region and characterizing the motion taking place there. For this the SpatioTemporalLocator makes use of the FigureTrajectory and ParameterTrajectory description tools. An example for using this Descriptor could be an observation platform or a museum.

3.7 A brief Comparison to other Textual Description Standards

Because MPEG-7 contains possibilities for annotating and characterizing content at different abstraction levels, it is important to distinguish between two groups of descriptions: On the one hand the textual ones, which correspond to the classical approach of human or semiautomatic metadata annotation - on the other hand descriptions characterizing the representation of audio-visual material, normally low -level features that can be extracted automatically from the data. We will first give a short comparison of MPEG-7 with other *textual description* standards, as Dublin Core, SMPTE/EBU as well as with standards that are part of the Semantic Web approach of the World Wide Web Consortium (W3C). In section 4.4 we will then provide a qualitative statistical-analysis of the MPEG-7 content based descriptors for image retrieval. The following information is given by [20].

MPEG-7 vs. SMPTE/EBU metadata The Society of Motion Picture and Television Engineers (SMPTE), an international organization of professional film and movie technicians which amon other things is responsible for the standardization of HDTV and EBU (European Broadcasting Union), defines a set of fifteen metadata classes (e.g. identification, administration, content description, technical descriptions, etc.) which all are represented by fields, clearly restricted in length and coding. Although the data set is very limited (and also not extensible), it also is not possible to divide the source into different segments for describing them separately, nor to create a dependency between its media elements. Both standards - MPEG-7 as well as SMPTE 395M - allow the description of multimedia content.

MPEG-7 vs. Dublin Core: Dublin Core (DC)[14] offers a small set of generic elements (controlled vocabulary) to describe arbitrary objects on a niveau of medial abstraction - they are organized in three groups: Content, Intellectual Property and Instantiation. As MPEG-7 it is based on XML and explicitly allows reusing and restructuring of its elements. Due to the fact that DC is not a media specific standard, it is only able to cover a small part of the MPEG-7 textual description functionality and especially has problems describing topics related to the usage or the navigation of multimedia content. In addition it is not possible to define subsets or aggregations which may result in ambivalent interpretations. Contrary to the SMPTE metadata there is no concept on how to integrate metadata into the media file. But its simple design makes DC easy to use (e.g. in xHTML or RSS 1.0).

MPEG-7 vs. Semantic Web Standards Semantic Web standards of the W3C, as RDF (Resource Description Framework), OWL (Web Ontology Language) or CC/PP (Composite Capability/Preference Profiles) offer rich meta-sets to define the structure of a description - therefore they are suitable to define descriptions of arbitrary complexity similar to MPEG-7 ones. However RDF only offers a meta-conept which does not specify the actual description form - it has to be defined autonomously. MPEG-7 as well as OWL allow the standardization of description vocabulary, but beyond that the latter enables the restriction to concrete object instances (e.g. for a description element like "cameraType" it would be possible to define a list of devices) which is absolutely mandatory to guarantee exchangeability and translation of descriptions. CC/PP, which is based on RDF is used to describe profiles (of structure and vocabularies) for different devices (e.g. for mobile phones, PDAs, etc.). It offers a greater degree of defining complex elements than MPEG-7 does and dominates the standard especially in the areas of user customization and personalization. Finally because of the higher niveau of abstraction which is offered by the Semantic Web Standards, the effort - compared to MPEG-7 - to implement them is higher. Topics of assigning and synchronizing metadata with their corresponding media are not dealt with and have to be addressed for example by using

[14] Dublin Core, Dublin Core Metadata Initiative, http://dublincore.org/

software as the Synchronized Multimedia Integration Language (SMIL [3]).

A more detailed comparison especially stressing the interoperability of MPEG-7 with RDF and Dublin Core, also including examples of e.g. an MPEG-7 ontology in RDF, is given in [10, chapter 3].

4 XM - the MPEG-7 Reference Software

The eXperminentation Model (XM) is the simulation platform and reference software implementation of the MPEG-7 normative components. One of the main goals was to provide a draft of the standard itself by building the best possible model and to verify the standards normative components functionality. The XM includes code representing Descriptors, Description Schemes, Coding Schemes, the Description Definition Language as well as the BiM and TeM System components. Besides these elements it also makes use of several non-normative components (e.g. like the ImageMagic or Afsp libraries) to execute procedural code on data structures, mainly normative elements. The data structures together with the procedural code form a so-called application.

4.1 Applications and Modularity

The eXperimentation Model arose from combining different proposals during the collaborative phase of the standardization process and afterwards was updated and improved in an iterative way. After having reached the status committee draft, all components corresponding to the normative elements were tested against contributions and proposals in well defined test conditions, the so-called Core Experiments before they were integrated into MPEG-7 part six. Although most Descriptors and Description schemes have at least one representative application in the XM framework showing their functionality either in metadata extraction and annotation of audio visual content, or on how they can be used in simple software applications, they are limited through several restrictions. First of all the XM only implements very elementary application types (which we will have a look at in section 4.2) - now real word scenarios! The software itself is based on a command line interface, it does not offer a GUI - and is not designed to allow any interaction during run-time.

An application in the reference software is clearly related to only one particular Descriptor or Description Scheme. We distinguish two types of applications within the framework: the server side, which is responsible for data extraction and the client side, that handles searching, filtering and/or transcoding tasks. While the extraction of low-level elements of an audio visual content is normally an automatic process, the extraction of high-level features requires additional information, which is then read along with the media data during the extraction process to create a description. This high level input data is therefore also provided in the standard and extends the multimedia content set.

By default the reference software covers many individual Ds and DSs which results in a large executable. It therefore is extremely important that the XM framework is designed to support partial compilation - which allows the usage of only e.g. a single D or DS - and offers combination of subsets. This increases scalability and flexibility. Furthermore, in order to provide the reusability of code, all applications in the XM are built from modules which use descriptor independent interfaces. This has the advantage that - after having understood

the structure one time - it is possible to reuse and combine modules much easier in bigger applications, without having to know in depth what is done in the included tool. The following six application modules are known and connected to each other forming a processing chain: [14]

1. *Media Decoders:* The Media Decoder module is normally the starting point of a flat application chain. Its MediaIO class supports a multitude of different media input formats like audio WAV files, MPEG-1 video streams as well as their motion vectors, still images as e.g. JPEG, GIF or PNG, n-dimensional key point lists and other proprietary input formats for high level information. For this purpose external libraries which do not belong to the XM software source code, like the free ImageMagic libraries for reading and decoding still images or Afsp for handling audio media files in the XM software, are used. (A frozen snapshot of these tools may be downloaded from the Munich University of Technology[15]).

2. *Multimedia Data*: The MultiMedia class is responsible for loading audio-visual data into memory. Video sequences are treated differently because they are too resource intensive - for that only single frames of a sequence are loaded into memory.

3. *Extraction Tool*: The module of Extraction Tools performs the feature extraction for a single multimedia element and therefore receives references to the input media data and to the description file which stores the results of the extraction process. The extraction process itself is a non-normative tool of MPEG-7 part six. When processing video and audio files, where it is not possible to provide the total input data at one time, a frame/sample by frame/sample extraction is performed. It first initializes the process, then iterates over all frames extracting a part of the description each time and finally offers a post extraction function, which is required when certain descriptions can only be generated after all data was available. For this the XM software uses the external AddressLib, a generic video processing library that performs low level processing tasks.

4. *Descriptor Class*: The Descriptor Classes hold the description data for the normative parts (there are classes for each Descriptor and Descriptor Scheme) of the standard. While Visual Descriptor classes in the XM software use plain C++ all other classes are implemented using a generic module - the so called GenericDS - which is an interface from the C++ software to the DDL parser. Concretely It is implemented as XML parser providing the DOM-API. The Descriptor Classes also offer memory management functionality for description data, which is done by the DOM[16] parser library.

5. *Coding Scheme*: The module Coding Scheme contains the normative encoder and decoder for a Descriptor or Description Scheme. Coding here is

[15] MPEG-7 TU München, http://www.lis.ei.tum.de/research/bv/topics/mmdb/e_mpeg7.html
[16] DOM, Document Object Model of the W3C, http://www.w3.org/DOM

referred to as writing the description to a file, using the GenericDS, or as loading and parsing descriptor information into memory using the DOM parser library. Besides the textual XML representation of a description, MPEG-7 also standardizes the BiM, a binary format.

6. *Search Tool*: A Search Tool takes an MPEG-7 Description as input and a - not necessarily MPEG-7 compliant - description for the query. The search tool then navigates through the description and processes or manipulates the input data in a useful way: e.g. in a search & retrieval application the tool compares descriptions by computing similarity values for them.

4.2 Key Application Types in the Software

We will now provide an overview of the different applications, which are implemented in the XM software. These so-called *key applications* are basic and elementary application types which represent carefully selected scenarios (e.g. application types for searching, extracting features, etc.). They are able to form the basis of real-world scenarios by showing on the one hand how such an application is used by implementing key features and on the other hand by pointing out the architecture and the work flow of the process (see Figure 12 for an example of the XM key application type "Extraction from Media" - all other application types are displayed in: [14]). There are four application types implemented in the MPEG-7 reference software:

1. *Extraction from Media:* This key application, which is of the extraction application type, extracts Ds and DSs from a given input file under test (DUT). All low-level Descriptors therefore should have an application class of that type implemented. In this process, as shown in figure 12, the media is first loaded into the multimedia class (e.g. into memory) by using a decoder and then, in the next step, a selected extraction tool is used by the multimedia class to achieve the description, which is then encoded and finally written to a file. This process is repeated for all files in the media database.

2. *Search & Retrieval Application*: belongs to the client application type. The "Search & Retrieval Application" first loads all descriptions of the database into memory and decodes them. Afterwards a MPEG-7 conform query description is either directly extracted from the media or loaded from a local file before the query is run. The query processes over all elements of the database and computes distance values for sorting out a list of similar elements that are then written - as a new media database - to a file.

3. *Media Transcoding Application*: This basic XM application, as well as the above one are of the client application type. After loading the audio-visual material along with its descriptions, the descriptive metadata is consumed and the media files are modified (transcoded) according to their

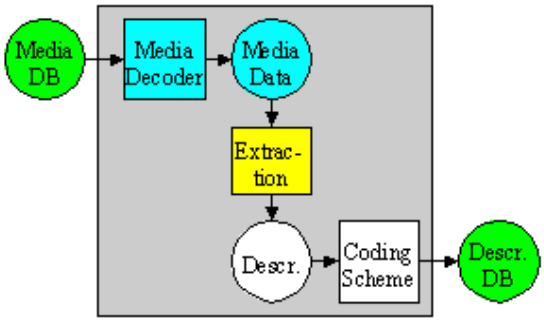

Figure 12: Example of an XM Key Applicaton Type: Extraction from Media - The description is extracted from the media input data. [14]

descriptions. Beyond that a query may be specified which is then processed on the description before the transcoding takes place. The resulting media database is then written to file.

4. Description Filtering Application: The last specified key application is either of the extraction or client application type. This depends on whether the description under test (DUT) is produced or consumed, whereby in both cases the input descriptions of the input database are filtered according to the query. The process is then completed by writing the resulting filtered descriptions to the output files.

Because each key application only needs to produces a single output - when it produces two or more different output types it can always be decomposed and therefore is not elementary - applications with more than one output are not key applications.

The next step, after having defined the key applications - which was done by using interfaces - all possible in- and outputs used in the application model must be determined. The resulting subset of possible inputs are media databases, description databases and queries - possible outputs are media- and description databases, anyway it is uncertain that a corresponding output type for the query input will be defined some day. However the abstract model does not consider the semantics of the media database (i.e. the list the best matching media files and the transcoded media database are not treated as individual output types).

Key Applications vs. Real World Applications The previously described applications in the eXperimentation Model are consequently called "key applications" because they represent basic and elementary application types.

Hence it is important to note, that they only implement key features (i.e. representative and common tasks) of application scenarios and do not necessar-

ily fit (detailed) real world requirements. However key applications may have arbitrary combinations of inputs - the model is generic for that - and as a result therefrom it is possible to decompose real world scenarios into a) processing networks of elementary key application blocks and b) user interfaces for interaction and presentation of results. This can be useful for designing products and helped for example defining the evaluation criteria (e.g. bit stream complexity, retrieval rate, computational complexity) of the MPEG-7 core experiments during the standardization process. [14]

4.3 Limitations and Problems using the XM

In this section we want to give an overview of the main problems and limitations we experienced using the eXperimentation Model during the practical work of the diplomarbeit. We especially want to thank Horst Eidenberger PhD, Associate Professor[17] of the Vienna University of Technology at this point for providing us with lots of valuable information regarding the compilation, the configuration and the execution of the XM - parts of this dialogue process and the resulting knowledge form the basis and the input for several practical hints presented here. Later on we will give an overview on his research results concerning the quality of the implemented MPEG-7 Visual Descriptors in section 4.4 and introduce their visual information retrieval project VizIR in section 4.5.4.

4.3.1 Obtaining the Software

In general the reference software implementation is available for UNIX as well as for all Windows 32 Bit operation systems with small OS specific differences (and difficulties). Because of that it is important to note that, although most MPEG-7 XM working group tests were performed on the UNIX distribution, the nature of our implemented software requires a machine running the Microsoft Windows XP Professional OS - therefore all our information given here is based on running the XM on Windows XP.

One is free to download the XM Software (a .zip archive with approx. 25 MB of size that fits both types of operating systems) from either the MPEG Working Group[18] or the Munich University of Technology, Institute for Integrated Systems[15] . While the working group directly hyperlinks to the ISO/IEC 15938-6(E) source which is probably up to date, the latter web page is not updated anymore but still provides the software as well as frozen snapshots of the required "third party libraries" Afsp and ImageMagic as well as a simple GUI for the XM software (which is available for Linux, but no longer supported).

[17] Horst Eidenberger PhD, Associate Professor, http://www.ims.tuwien.ac.at/~hme/

[18] MPEG Working Group, MPEG-7 Reference Software, http://www.chiariglione.org/mpeg/standards.htm

4.3.2 Compiling the Source

The attached guideline contained in the downloaded archive for compiling the XM source is unfortunately very short and beyond provides helpful details sparingly. Actually compiling the software was a major problem for us and is a tough barrier for using the software! We will now present valuable troubleshooting information, but for all who want to save lots of time and irritation we recommend using our precompiled version (for Windows XP including all external tools like parsers, libraries, XM GUI of the TU Munich, etc.), which is included on the CD with the software to this diplomarbeit (see annex 8.3.2). Note that although our included guide for installing the software (see Annex 8.3.2) describes the process and resources which are necessary for offering the XM functionality as web service, the XM, though, exists as a stand-alone component in the directory "mpeg7\newsrc" and may be used independently.

4.3.2.1 Troubleshooting for Compiling the XM

- When following the compilation guideline and having removed the modules that shall not be compiled from the XMLib module (make sure the XMWinExe module is to be compiled!) it is possible to get a compiler error for the XMLib[19]. This is because several modules don't compile correctly either under UNIX or Windows.

- However the suggestion to try the XMLib_OK.dsp instead of the XMLib.dsp is not up to date anymore. The XX_OK versions only had the modules integrated, which compiled without error for a certain operation system but with the main release of the software it was restructured and the XX_OK projects were replaced by an XX_ALL containing even those that do not compile.

- The compilation guideline tells the user to "remove the modules which you don't like to compile from the XMLib module". The problem here is that there are several similar XMLib modules, reaching from XMLibA, XMLibM, etc. having sub-folders with identical titles (e.g. BasicModules) which makes it difficult to find the right ones. This structure has been used since version 6 of the XM software and divides Audio, Visual and MDS into separate libraries holding their Descriptor modules. Additionally there is the "AP-Lib" (application modules) that is needed for the reuse in other systems - here the application modules of the individual Ds and DSs are found. When deleting a Descriptor from the project the following steps must be taken:

  1. *Delete the D/DS-modules from the project.*

     As described above the modules are organized according to their MPEG-7 parts as e.g. Audio or MDS

[19] According to the guidline the Microsoft Visual Studio .NET 2003 Professional was used for compiling the source

2. *Remove their corresponding entries from the AP-library.*
3. *Erase the "USEXX-define"* (see next step)

- A module that is removed from the project will leave a gap in the linking process and therefore its function calls must be removed from the main-routine. Actually the main-routine calls a function which then selects the proper application - the applications are specified by USEXX-defines and must be given at compilation time. The main problem here is, that a relationship between the individual modules and their corresponding USEXX-defines are at no stage given in the XM documentation. They are collected and listed in [10, Annex D]. By the way, deleting either BasicArchitecture or BasicModules leads per guarantee to a failure in the system's linking process.

4.3.3 Configuring the Runtime Parameters

Because of the limitation that the XM software is a command line tool, all inputs and outputs as well as their configuration parameters must be specified before starting an extraction process - interaction during run time is not supported by any key application. The extraction process may easily be invoked by using either a batch-file, activating the XMMain.exe and providing the used input settings like type of application (e.g. ColorLayoutServer) and descriptor configuration parameters, or by calling the executable XMMain directly with the given input values. Using the first method is the preferable way of launching an extraction process because it is not only possible to chain multiple tasks in a series but also allows the generation of different logs - including information regarding the success of the extraction process - as well as redirects STDOUT very comfortably by using the "s" tool.

The actual descriptor configuration is done by creating a so-called ".par" file - an input format for the XMMain. A list containing the mandatory values that must be set for all description applications[20] is provided in Table 1. This includes the name of the client application, a list of input files on which the extraction is run, a bit stream to that the results are written, as well as the used coding mode. The "NoOfMatches" which defines the size of the result-set when using a "search key application" has no functionality in our software implementation since we used the XM for feature extraction only. Due to the fact that the default settings may not be characterized very reasonably, the fine tuning of the descriptor settings (e.g. defining the used ColorSpace) is a very essential task. Unfortunately the information available in the reference software is minimal, only a small guide is found in the documentation[21] of the XM. Additional knowledge on some of the descriptors was collected by Eidenberger during his work with the software and is listed in Annex 8.3.2 - however it is far from giving a complete survey.

[20] Similar to Eidenberger: mpeg7/newsrc/wiTests/parameter_description.doc in the XM precompiled package

[21] see mpeg7/newsrc/Doc/html/index.html

Variable	Interpretation	Example
Application	name of used descriptor application	ColorLayoutClient
ListFile	reference to the media input file list	pluginMETIS/list_of_pictures.lst
BitStream	reference to extraction results	desc_extracted/ColorLayout.xml
NoOfMatches	size of the resultset	8
CodingMode	0...DDL, 1...BiM, 2...Binary	0

Table 1: Mandatory Descriptor Settings

For further explanations on the given (relative) path/file structure, the extraction files (.bat) or the input parameter settings (.par) of the reference software see section 7.2.2.5 describing the practical implementation of the diplomarbeit. Information regarding the efficiency of the descriptors using a given set of settings is provided in the next section.

4.4 Significance of the MPEG-7 Visual Descriptors - a Qualitative Analysis

How efficient are the MPEG-7 descriptors in the task of distinguishing different media content and how well do they suit their intended application domains? In this section we want to answer these questions from a statistical point of view by discussing the analysis of eight visual descriptors. These descriptors were applied to different sets of images whereas the extracted descriptions were then inspected, regarding their qualities in the fields of redundancy, sensitivity and completeness. The outcome of the study is of special interest due to the fact that the MPEG-7 design process focused on the optimization of the retrieval rate, rather than on the quality of the descriptor set. Goals of the evaluation process were to provide a guideline on the usage of the descriptors on the one hand (e.g. which of them make sense in combination for visual information retrieval) and to reply to the weaknesses with suggestions for their improvement on the other hand. The presented study came along with the development of the visual information retrieval project VizIR and had an influence on its implementation. The main concepts of the study as well as its interpretation is provided by [30].

4.4.1 Information regarding the Test Environment

The analysis included eight descriptors, namely all MPEG-7 color features (see section 3.6 for further explanations on MPEG-7 Visual): *ColorLayout*, *ColorStructure*, *DominantColor* and *ScalableColor*, the entire set of texture descriptors: *EdgeHistogram*, *HomogeneousTexture* and *TextureBrowsing*, as well as one shape descriptor: *RegionbasedShape*. Other basic descriptors were not investigated because either their resulting data cannot not be transformed to produce measurement on interval scale basis (as e.g. the second shape descriptor ContourbasedShape) or they belong to the temporal domain and therefore would only be comparable if the basic color, texture and shape descriptors are

aggregated over time. However it is wise to first evaluate the basic features before giving explanation on aggregated structures or high-level characteristics.

For the extraction of the description data, the MPEG-7 reference software XM was applied to images of three highly independent media collections: the Brodatz dataset, the Corel dataset and a collection of coats-of-arms images. While the *Brodatz* dataset consists of about a hundred monochrome images that especially should fit the texture descriptors and rather challenge the ones for color and shape, the *Corel* images, a widely spread set of color photos showing humans, landscapes, animals, etc. were assumed to deliver expressive results with color and texture descriptors. The *coats-of-arms*, a dataset of artificial color images, is located between the two others and is characterized by having clear structures and few color gradations. Because of restrictions concerning the applied statistical algorithms, only a randomly chosen subset of images was used in the study. Note that each feature was applied on the whole media object.

The resulting data vectors, containing the extracted feature information, were then normalized before computing the four statistical evaluation properties: mean and variance of description elements, distribution of elements, hierarchical and topological cluster analysis as well as a factor analysis. A description which is suitable to distinguish between different media content is characterized by a high variance and a well-balanced cluster structure. For details concerning the software configuration as well as on the analysis parameters see [30, section 3]. For an overview of the research questions as well as for the mapping to their corresponding statistical indicators see [31, table 1].

4.4.2 Presentation of the Results

4.4.2.1 Redundancy Analysis Redundancy information, i.e. the identification whether extracted description elements are unique or not, is a very valuable indicator for the organization of descriptors to a description scheme. Further this information may be used additionally to the binary format (BiM) for the compression of the description data.

The analysis however showed that the MPEG-7 descriptors generate *highly redundant* information - the ratio of description elements to redundancy-free ones varies from 4:1 to 7:1. These results are not very surprising due to the fact that the MPEG-7 descriptors were not evaluated with statistical methods in their design process. The descriptors are especially *vulnerable to monochrome media* - four out of eight descriptors represent color features - however their algorithms in the XM software were implemented to only use the luminance components. When the content contains enough color information ColorLayout, ColorStructure and ScalableColor are independent of each other as well as to other descriptors whereas it is interesting that all bins of ColorLayout are roughly unsimilar for any type of media content. The RegionbasedShape proves to be a good indicator for global shape information since the DC coefficient of its

underlying ColorLayout determines most elements and is unsimilar for different media content.

DominantColor is absolutely autonomous and neither intersects with any color descriptors, nor with the texture descriptors. Together with TextureBrowsing, ColorLayout and EdgeHistogram it belongs to the most independent descriptors - together they are *capable of covering most of the properties offered by the MPEG-7 Visual descriptors* and therefore form the most effective description scheme for visual content, exhibiting a minimum of redundancy. However in certain applications other descriptors may sill provide valuable information.

Figure 13 shows a self-organizing map of MPEG-7 descriptions for the Brodatz dataset. Every non-border cluster element is surrounded by six neighbors and therefore is displayed as a hexagon. A cluster which is populated by a certain descriptor is marked by its according texture - when it is shared, the hexagon is split up into triangular regions.[22]. This illustration contains a map of the Brodatz dataset and gives a first impression of the hierarchical cluster structure as well as on redundant descriptors. For example it shows that the HomogeneousTexture is capable of laying a fine-meshed net over the investigated media property. Although there are almost as many EdgeHistogram cluster elements as HomogeneousTexture ones on the map, the Edge Histogram elements are only connected loosely and spread over large regions. This wide-meshed structure allows the EdgeHistogram to cover a larger area of variance in the media data.

4.4.2.2 Sensitivity Analysis The sensitivity analysis is split up into three areas of interest: sensitivity of the color descriptors for monochrome content as well as for content with few color shades (like animations) and finally sensitivity of the texture features for coarse, medium and fine structures. Ideally, the extracted mappings of content to feature space should be robust against variations in the quality of the visual content (e.g. resolution, presence of color information, bleached input). For that, the analysis results on the one hand allow a testimony on how the input affects the extracted data quality of the descriptions - on the other hand they may be used for judging whether the descriptors are suitable for the proposed applications.

The analysis has shown, that *all color descriptors work excellently on photos*. However, especially ColorLayout, ColorStructure and ScalableColor have problems when only few color gradations in the media are present and deliver absolutely *poor results on monochrome content*. Although the DominantColor descriptor is very sensitive to brightness, it again works fine on all types of content. The statistical results of the color descriptors have clearly shown that the GroupOfFrames/GroupOfPictures (GoF/GoP), which was designed to generate

[22] The following abbreviations are used: CLD ColorLayout, CSD ColorStructure, DCD DominantColor, EHD EdgeHistogram, HTD HomogeneousTexture, RSD RegionShape, SCD ScalableColor, TBD TextureBrowsing.

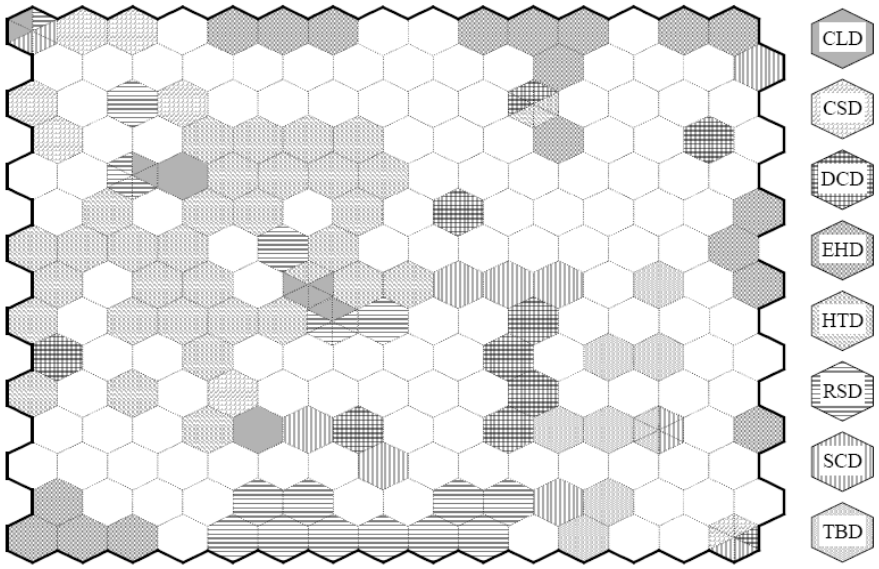

Figure 13: Self-Organizing Map of MPEG-7 description elements for the Brodatz Dataset. Neighbour clusters contain similar description elements. [30]

descriptions for short video segments and animations, does not provide valuable information in the absence of high sharpness, good lighting conditions as well as rich contours and shades in the media. Instead of being built on the ScalableColor feature it rather should have been based on DominantColor. For retrieval applications the study additionally recommends a redefinition of the MPEG-7 distance measures for the color descriptors.

From the point of statistical analysis the EdgeHistogram is the best texture descriptor because its data is characterized by high sensitivity and low redundancy. While RegionbasedShape is a good descriptor which can be applied to any type of media content, HomogeneousTexture is sensitive to the underlying media content and TextureBrowsing only produces partially ambiguous results, suitable for browsing but not for other MPEG-7 applications.

4.4.2.3 Completeness Analysis For a large dataset of dissimilar elements, the extracted values for each descriptor element should ideally be uniformly distributed over the available data range. On the one hand this would guarantee the optimal utilization of the data space and the best possible discrimination of the object by its feature characterization on the other hand. The completeness analysis is used to find "holes" in the distribution which are not covered by any descriptor and therefore cannot be captured in a description scheme either. Other measurements concern the utilization of the provided data range and the

occurrence of peaks.

The results have proven that most of the descriptor values could be transformed and quantized to smaller data types without loosing any precision. This could *save storage space* from 10% for ColorLayout, 50% for HomogeneousTexture and even up to 70% for the EdgeHistogram discription. While the clusters of color descriptors generate a fine-meshed structure, the texture descriptors suffer from large gaps, especially between EdgeHistogram and HomogeneousTexture, which TextureBrowsing because of its poor variance is unable to close. Suggestions for closing the gap without affecting the normative parts of the standard are offered in the paper.

To conclude, the best results are achieved when using ScalableColor in combination with the best working descriptors on a content, rich of colors. These descriptions then are even capable of registering small differences in the content by their fine-meshed cluster structure.

4.4.3 Summary

The goal of the study is to give a statistical analysis of the efficiency of eight MPEG-7 Visual descriptors on the one hand, since this was not taken into account in the MPEG-7 design process and on the other hand to define guidelines concerning the usage of descriptors as well as their combination. The analysis concentrated on fundamental properties as redundancy amongst the descriptors, their sensitivity for changes in the content and if the descriptors are capable of covering the proposed media property completely. The best descriptors for combination are ColorLayout, DominantColor, EdgeHistogram and TextureBrowsing. Others are highly dependent on them. The study showed that the color histograms are not suitable for monochrome media and that most of the basic MPEG-7 descriptions provide highly redundant information (e.g. in the presence of color shades). Therefore up to 80% of storage space could be saved when compression, quantization and transformation algorithms are used in addition. However some aspects of visual media content cannot be captured at all by the current set of MPEG-7 descriptors.

4.5 Current Projects and Tools - Based on MPEG-7

4.5.1 Introduction and Outline

We must admit that finding tools or applications based on MPEG-7 (Visual) which have currently been under development is a rather hard task. In the process of investigating the field we had contact with several groups that have been involved either in the standardization process of MPEG-7 and its reference software or in main software releases in the last few years. Prominent projects were for example the PRIMAVERA (Personalized Retrieval and Indexing of Media Assets in Virtual Environments for Real-Time Access) television archive, a Content Management System (CMS) with advanced features such as image similarity search, where the Austrian television broadcasting station ORF as

partner had a test case running for about a year, PicSOM[23] which is based on self-organizing maps for image retrieval with MPEG-7 content descriptors and already showed very good results in the recognition of defects in running paper and metal webs or SAMBITS (System for Advanced Multimedia and Broadcast Information Technology) that aimed to offer a new system architecture for creating, sending and receiving media broadcasts. But while a complete list of all 43 known applications under development in the year 2002 is given in [29], it is confirmed that no similar document summarizing the current work exists.

For research activity regarding current MPEG-7 projects and participating institutions and organizations, the author recommends either the *Multimedia Meta Data Community*[24], which wants to close the gap between academic research and industrial development by bringing together experts from research and industry in the area of multimedia metadata interoperability for collaborative working environments, or the *MPEG Industry Forum*[25] for research materials and tools.

It is obvious that there are fewer projects running at the moment, which - amongst several other reasons - may be attributed to a receding importance of the standard itself after a hype of expectations in its beginnings. However we must note that we neither have enough insight and literature in this area nor have done any research activity to follow this thesis, so that we are not capable of making a judgement.

In the following section we present a current overview of carefully selected tools and recently developed projects - using MPEG-7 (Visual) as base technology - reaching from annotation and CBIR (Content Based Image Retrieval) software, to a framework for building such tools, to finally two very interesting libraries that may be of interest when developing applications based on MPEG-7 metadata.

4.5.2 IBM Marvel - a Video Retrieval System

A prominent project based on MPEG-7, even granted the Wall Street Journal Innovation Award in the category multimedia in 2004, was developed by the Intelligent Information Management Department at IBM Research and is called MARVEL. This video retrieval application helps to organize multimedia data by automatically labeling its content. For that it is able to extract up to 200 different semantic concepts as "summer", "winter", "car" or "aeroplane" directly from video streams by using machine learning techniques. Because Marvel aims at the extraction of high level metadata it is impressive that the system only requires manual annotation of 1-5% of the content as training examples. The additional work is done by using multi-modal features (as speech

[23] Helsinki University of Technology, Labratory of computer and information science presents: PicSOM, a framework for content-based information retrieval http://www.cis.hut.fi/projects/cbir/

[24] Multimedia Meta Data Community, http://www.multimedia-metadata.info

[25] MPEG Industry Forum, http://www.mpegif.org/resources.php#section40

transcripts, visual, sound, etc.) to create statistical models accordingly to the training results. These models are then applied to large multimedia repositories for automatically annotating them (with associated confidence scores), what reduces the total costs of creating metadata as well as annotation errors. Performance in their retrieval engine is further improved by using ontologies to exploit semantic relationships. MPEG-7 in this video indexing tool is used in three different ways: first for producing/extracting MPEG-7 descriptions from video content, second for searching video content based on the MPEG-7 descriptions by text/key-word queries or by selecting examples of features or content, and third for adopting the digital material to the user's environment and preferences. An online software demonstration as well as further reading information is available. [21]

IBM already showed their MPEG-7 knowledge in a previous research project called VideoAnnex. This tool assists an author in the task of annotating video segments with MPEG-7 metadata in an interactive manner. After an automatic segmentation of the MPEG video sequence by the software one has the possibility to use static scene descriptions, key objects, event descriptions and other lexicon sets to annotate the video shots. The associated results are then stored in an output XML file, whereby the software is also capable of opening existing MPEG-7 descriptions and displaying the referenced source. [22]

4.5.3 MECCA - Multimedia Capturing of Collaborative Scientific Discourses

MECCA is a multi-dimensional video screening and classification platform, which is able to cover the requirements for a distributed collaborative multimedia knowledge management system. Due to its special design, that combines both: multimedia ontology and community vocabulary achieved by using the rich set of MPEG-7 tools for describing semantics, as well as for features of multimedia artifacts and collections, it is able to serve simultaneously as media classification and monitoring system for researchers (which e.g. may have diverse educational background knowledge, as well as different levels of profession) capturing scientific discourse and as an e-learning system for students by sharing the annotated media information. The basic ideas were taken from a former project called "Berliner sehen" developed at the MIT. We recommend viewing the detailed video demonstration on the project's web site for a first introduction. [27]

4.5.4 VizIR - a Framework for Visual Information Retrieval

The Institute of Software Technology and Interactive Systems at the Vienna University of Technology first introduced VizIR - a framework for visual information retrieval - in the year 2002. After having pointed out the major problems in the field of Content Based Image Retrieval (CBIR) and Content Based Video Retrieval (CBVR) they investigated several products (as e.g. IBM

QBIC, MARS, GIFT, etc.) and prototypes in the field and found out that most of them share a number of serious drawbacks in their development process and targets. [25] As a reaction VizIR was set up with the following goals:

1. To provide a *common basis for further research projects* by summarizing and integrating current knowledge in the topic of Visual Information Retrieval (VIR). This especially includes the areas of similarity measurement and modelling, feature extraction and query performance optimization. Therefore it offers an open class framework for methods of feature extraction, distance calculation and querying classes. It presents evaluated user interface components and prototypes for content-based visual retrieval as well as carefully selected evaluation sets for groups of features (e.g. color, texture, shape, etc.) with human rated co-similarity values. This model for similarity measurement was developed with the goal to be aware of subjective human perception (i.e. that the same/different person may judge or perceive features as color, texture, etc. differently).

2. Implementation of an open (by using Java, wrapper classes and standardized APIs for database and media access), extensible and well documented *framework for CBIR*. This includes methods for feature extraction, query mechanisms or user interfaces as well as benchmarks or test-cases which are released to the community at specified intervals. Additionally to the concept of modelling semantic descriptors a main effort lay in the evaluation of the MPEG-7 Visual Descriptors.

3. Cooperation with other research groups

VizIR is free and available under the GNU Public License and may be downloaded from:[26]. The tools Caliph and Emir, described below, include code of the VizIR project in their latest additions.

4.5.5 Caliph & Emir project - using MPEG-7 Descriptors

Photo and image annotation tools need to fulfill well defined requirements regarding topics of storing, indexing and retrieving multimedia content. Caliph and Emir are MPEG-7 based Java prototypes developed for the proof of concept of a graph-like annotation with semantic metadata and a content based image retrieval based on MPEG-7 descriptors. *Caliph* on the one hand makes use of already existing metadata sets which are encoded into digital photographs in the EXIF format and transforms this information to MPEG-7. But on the other hand it also allows the creation of new metadata - as semantic information - which is represented as directed graph with nodes reflecting objects, locations, agents, states, etc. and edges defining relationships between them. The provided MPEG-7 information consists of metadata descriptions, media, creation and textual data, semantics and visual features.

[26] Visual information retrieval project VizIR, Vienna University of Technology, http://vizir.ims.tuwien.ac.at/index.html

Emir the experimental metadata based image search tool supports multiple retrieval mechanisms for content based retrieval as MPEG-7 features (ColorLayout, EdgeHistogram and ScalableColor Descriptors are implemented) provided by samples, a key word retrieval using a Lucene search engine as well as a graph based retrieval with the possibility of specifying wild cards for semantic relations and objects. In the latter case it uses the FastMap algorithm to display a 2D visualization of the data repository (e.g. based on color information) which is described by [28] in detail. Because of the fact that the prototype only uses images for the visual representation of search results, it can easily be extended to handle all media items supported by MPEG-7. Emir for example is integrated in MEDPHYT, a database for plants of medical interest developed by the Beilstein Institute in cooperation with the Klagenfurt University of Technology, Institute for Information Technology. The two presented tools which are available under the GNU Public License as well as an online demonstration and further publications regarding their motivation and requirements may be found and downloaded at [26].

The Lucene Image REtrieval (*LIRE*) library which was developed as part of the Caliph & Emir project is an easy and light weight possibility for both, creating a Lucene index out of image features as well as enabling search of a given index. Note that the descriptors ScalableColor and Colorlayout were taken partly from the XM software while the origins of the EdgeHistogram implementations lies in the VizIR project. The library's main intent is to provide the CBIR features of Caliph & Emir to other Java projects.

4.5.6 MPEG-7 Library - a C++ API Implementation

Generally speaking the MPEG-7 Library, a current project at the Joanneum Research Institute of Information Systems & Information Management, consists of a set of C++ classes which implement the parts 3, 4 and 5 of the MPEG-7 (ISO/IEC 15938:2001) standard. Application developers who need to deal with MPEG-7 metadata may use this library to create multimedia content descriptions, manipulate them and serialize this metadata to and deserialize it (including validation) from XML. In that way it is neither necessary to struggle around with the complex XML DOM API when parsing the MPEG-7 facets - all hierarchical XML representations are made available in an object hierarchical class tree, nor one does one have to implement hundreds of MPEG-7 classes (the library already offers about 1200 classes) before adding MPEG-7 functionality. The source was originally designed for Windows 32-bit operating systems (but also compiles on UNIX) and is currently available as C++ implementation in version 2.0. However due to its design, which does not use multiple inheritance and templates it should easily be portable to C# and Java. Since the latest version, which was released in April 2006, it is now even possible to not only use XPath for accessing nodes but also for creating parts of a description using XPath statements. [23]

A second very interesting project which was launched in January 2004 at

the Joanneum Research, is called DIRECT-INFO - Media Monitoring and Multimodal Analysis of Time Critical Decisions. Along with eight important international partners, as Fraunhofer IGD Germany or Nielsen Media Research Italy, etc., they established a system which is capable of monitoring media automatically. A major goal, the unsupervised extraction of expressive semantic information, is achieved by the combination of different basic media analysis modules (like genre classification, speech and text analysis as well as topic detection, logo and object detection, etc. which are accessible as web service) to an integrated system. While in the first step logical entities are extracted (e.g. this is done by using pre-specific schemes for different kinds of information), the second step makes use of predefined knowledge to semantically correlate this data. The resulting consistent and meaningful semantic information is then of a quality level which is usable for humans for reasonable decision making. The software prototype at the moment is used for example to automatically monitor TV programmes and print media for sponsorship tracking [24]

5 Web Service Technology

In this chapter we give a short introduction of the web service technology and describe its main components. Afterwards Apache AXIS, one of the most prominent Java SOAP engines in the field as well as Soap with Attachments (SwA), a solution for attaching binary data to SOAP messages is presented.

What are Web Services? Generally speaking a web service can be depicted as a piece of business logic which is accessible through standard internet protocols. A consumer of a remote web service is not tied to the system directly but is offered an interface. This interface may even change over time without affecting the client's ability to interact with the service. By using XML as data representation layer for all web service protocols it provides interoperability and platform independency for exchanging data as well as complex documents.

The web service aspect assembles three main technologies which make up its core. These are *SOAP* (Simple Object Access Protocol), *WSDL* (Web Service Description Language) and *UDDI* (Universal Description, Discovery and Integration). SOAP is responsible for transporting XML messages over a variety of standard internet protocols. As it defines a structure of document exchange for executing Remote Procedure Calls (RPC) it is possible to let heterogeneous systems running on diverse platforms and/or even written in different programming languages (as e.g. Microsoft .NET and J2EE) interoperate with each other. For describing the service's interface in a standardized way WSDL is presented. It defines all input and output parameters for invoking methods and further contains the function's structure as well as the protocol binding. Finally making a service available to others is done by the UDDI, a worldwide registry for finding web services. Note, the W3C are the primary committees for standardizing the web service architecture and its technologies.

So the minimum requirements for interacting with a web service are that each participating party knows how to construct and deconstruct SOAP messages and that it is able to send and receive data via the HTTP protocol. [32]

5.1 The SOAP Engine Apache AXIS

5.1.1 Introduction - What is AXIS?

As web services build on XML their specifications focus on the XML aspects of the technology. However as it is not discussed how these technologies can be bound to certain programming languages (e.g. as Java) a number of industry solutions are proposed for facilitating Java web service development.

One of the most prominent ones in the field is Apache AXIS. AXIS stands for "Apache eXtensible Interaction System" and is an open source implementation of the web service standard SOAP. Its name however, can stand for almost everything as it was chosen at a time when the W3C had not yet agreed on the name for the SOAP technology. In 2000 the engine was totally restructured for making it more flexible and configurable as well as for adding support for

both W3C specifications: SOAP and the XML. AXIS currently is available in the third generation of the Apache SOAP project and has its roots at the IBM "SOAP4J". The current version is written in Java but a C++ implementation of AXIS's client side functionality is under development.

AXIS can essentially be characterized as a SOAP engine, i.e. a framework for creating clients, servers or gateways which above that is capable of hosting and running web services. The benefits from using the software to build a service or to implement a client for accessing a web service can be easily summarized. When developing web services with AXIS one neither has to care about encoding requests to a service or decoding the response messages, nor one needs to implement the program logic for handling message exchange. Developers may just write their application code and let AXIS do the rest. It includes tools and libraries for generating web service code from scratch - either from existing Java code or from a WSDL description file, for making the service available and for handling all SOAP, XML or JAX-RPC message exchange. A user is not required detailed knowledge on the standard anymore as he operates on local Java objects for getting a handle on the remote system and for invoking remote methods. [3]

A good tutorial presenting the first steps on how to develop web service with Apache AXIS is given in [33]. We will now present a short overview of the platform's supported key features.

5.1.2 Key Features

The main design approaches AXIS follows are: speed - it uses the event-based parsing mechanism SAX instead of DOM which was contained in former versions of Apache SOAP, flexibility - for extending the engine e.g. by customizing header processing and stability - as AXIS defines a set of interfaces which are slowly changed compared to the rest of AXIS. Its functionality can be shortly summed up as: [34]

- AXIS is not just a SOAP engine but also includes a stand-alone server as well a plugin for the web service engine Tomcat.
- It offers extensive support for WSDL and is capable of generating description files from the fly even when the service is up and running. This description file, containing the service's interface (its input and output parameters, a description of the function as well as the protocol binding, etc.) can then be directly used by a client to consume the service.
- By supporting a "drop-in" deployment of SOAP services (JWS) it is possible to write a Java application, drop it in the AXIS folder and see how it (its publicly offered methods) automatically become a web service.
- In section 7.2.3 the WDSL2Java tool for generating proxies and skeletons from WSDL input as well as the Java2WSDL tool are presented in detail. We made excessive use of their functionality in developing our software.

- Preliminary support for both Soap with Attachment (SwA) specifications is integrated into AXIS. An introduction to SwA and its two available implementations DIME and MIME Multipart see section 5.2.1)

5.2 Soap With Attachments (SwA) - for Transferring Binary Data

> "While XML and SOAP are very good at describing data, many kinds of application data aren't well-suited for XML—for example, a piece of binary data such as an image[...]. SOAP with Attachments (SwA) was born in recognition of this limitation. SwA combines the SOAP protocol with the MIME format to allow any arbitrary data to be included as part of a SOAP message. The model is exactly the same as the model used for including email attachments." [32]

Following the evolution of web services and their underlying protocols we see that the technology has gone from supporting requests with simple parameter types (as string, integer, boolean) to offering a full integrating of object-oriented languages by adding encoding rules for their complex data types. However the last step, adding improved support for binary data is coming up lately with SOAP with attachments.

SOAP which is used for inter-process communication in web services is built on XML and so inherits all of its inefficiencies i.e. requiring more bandwidth, more storage and more processing power, due to the fact that XML uses textual encoding instead of a binary format. "According to the W3C XML Schema specification, binary data should be encoded in base 64 or hexadecimal. Unfortunately, 64-encoded data is 50% larger than non-encoded data. Hexadecimal encoding doubles the size. This overhead is acceptable for small pieces of binary data, but it is clearly an issue for larger sets." [36]. The offered solution SwA therefore removes binary information from the XML payload and stores it directly as a multipart related MIME content into the HTTP request and further provides URI schemes for referencing them.

Currently not all available web service toolkits provide support for SOAP with Attachments but the number is strongly increasing - the W3C however is working on integrating support for attachments in SOAP version 1.2. A different solution which is based on DIME instead of MIME for attaching binary data to SOAP is presented by Microsoft. A short comparison of the two approaches is presented in the next section.

Apache AXIS (since version 1.2) supports both methods of SwA and makes attachments available to Java developers through JAX-RPC (the Java API for XML-based RPC) and SAAJ (SOAP with Attachments API for Java). The main distinction between the two libraries is their level of abstraction they offer but not their capabilities. Additionally required components are the JavaMail and the JavaBeans Activation Framework which must be available to AXIS.

5.2.1 Performance Evaluation of SOAP SwA DIME/MIME

Differences between SwA DIME and SwA MIME The Microsoft Implementation of adding support for binary data for SOAP is called DIME which stands for Direct Internet Message Encapsulation and presents a packet mechanism for streaming arbitrary data simultaneously within SOAP messages. The distinguishing features of DIME can be summarized as follows:

- Like DIME, MIME Multipart may contain a number of data records, each including a header part and a certain payload. However, instead of serving the total length of a record for specifying where to find the next header section MIME uses ASCI delimiters to separate them. Therefore when a message is received its data must completely be analyzed for determining simple things like e.g. the number of attachments contained in a message.
- Every data record contains a self-describing header which can be used by a parser to interpret the message
- When sending DIME attachments the system does not have to know its total data size as it is streamed. This avoids inefficiencies on the receiving side (e.g. buffer size management). Some MIME implementations however react to this limitation by adding a "content-length" headers to the MIME data. This solves the problem only partially as the system therefore must wait until the total length is known before it can start with the data transmission. Using delimiters therefore is in both situations not the best solution.
- It contains a "chunking" mechanism

To summarize, both available approaches for using SwA define an abstract model for SOAP attachments and contain mechanism for encapsulating them - in either a MIME or DIME format - into a SOAP message.

Summarizing the Results of the Performance Evaluation Study The presented study evaluates the different SOAP variants: standard SOAP, SwA-MIME and SwA-DIME in terms of efficiency, data size and speed. The main point of interest therefore was to give a statement on the performance improvement which is achieved by using the binary extensions compared to standard SOAP for transferring binary data. It was measured how long it takes to do a round-trip process i.e. sending a message from a client to the server and receiving its response again, which message size was transmitted over the wire and how large the overhead on serializing and deserializing SOAP calls is, as some performance studies pointed out that XML serialization and deserialization are the bottlenecks in SOAP processing. The results of the study can be summarized as follows: [36]

- Every 1 byte increase of size of the raw data will result in an increase in size of the standard SOAP message of about 7.2 bytes which costs another

0.12 ms in serialization. SwA compared with standard SOAP only has an increase of 1.1 bytes in the SOAP message's size. So it is clear that both SwA approach are able to avoid large amounts of XML encoding/decoding for improving SOAP performance

- SwA-DIME is always a faster and more efficient message processing approach than MIME although they provide the same functionality. Specifying the data record's length in the header rather than using a string delimiter at the start and the end of each record makes parsing faster and memory allocation more efficient. It takes about 10 times longer to perform the serialization for SwA-MIME for a 128x128 floating-point matrix and 6 times longer when communicating two of them.

- DIME contains a small set of fixed headers which makes it easier for tools to support DIME. This decreases the risk of interoperability and also makes processing fast and efficient.

- The results of investigating the overheat of serializing and deserializing for communicating 5000 data structs state that SOAP has a significant deserialization overheat which is responsible for up to 75% of its latency time. This derives from adding several tags for every element of the matrix after encoding it for representing its complex data structure. However each of these tags is required to provide its own definition of encoding-style and namespace which then must be interpreted every time when deserializing the SOAP message.

To conclude, the study indicates that SOAP performance can drastically be improved by using SwA when dealing with complex data types and/or a large amount of data. The time and memory consuming task of encoding and decoding XML can thereby be reduced to a minimal level. For detailed information on the study's design, the test-environment as well the results see [38].

Part II
The XMFE - a MPEG-7 Feature Extraction Service

6 Introduction to the System Components

As it was already mentioned in the introduction of this paper there are several reasons why METIS has a need for supporting basic structures of CBIR. With the constructed software therefore it is possible to extract MPEG-7 descriptions from image data and to integrate them into a METIS data model. These can then be used as basis for building multimedia description schemes for automatically classifying data, for retrieving images on feature basis or just for sorting out duplicate media objects. All of the results are stored as well-formed XML files which can be validated against their MPEG-7 scheme. Therefore already existing algorithms in the field are applicable without having to restore the original descriptions from an proprietary format. However, note that it is not part of this thesis to implement a similarity measurement on top of the extracted descriptions.

The system architecture can be separated logically as well as physically into two independent parts: On the one hand the XM Server, offering the feature extraction functionality on the basis of the XM framework as web service and on the other hand the METIS plugin XMFE and its corresponding semantic pack XMBasic for consuming the service and integrating MPEG-7 support into METIS. A general overview of the arrangement of constructed components, their interaction as well as the technologies they are built on is depicted in Figure 14.

This design was chosen due to the fact that the MPEG-7 reference software, its external libraries as well as the developed Java wrapper, in the context of the multi-user server application are bound to a specific operation system and its infrastructure. Beyond that however, it also makes sense to physically separate the feature extraction functionality from the METIS plugin for performance enhancement (e.g. higher hardware requirements for the XM) as well as for its reusability and availability to other systems (e.g. this follows the BRICKS approach - a distributed, service-oriented architecture for European cultural institutions).

The standard procedure is activated when the plugin receives an event of a changed media item. After verifying the service's support for this type of data, the client handle on the service contained in XMFE is used to upload the data to the client's remote workspace and to set the features that are supposed to get extracted from the service, according to the plugin's configuration. After the actual call for extraction is made, the plugin requests the results and attaches them to the attributes of the XMBasic's media type of the corresponding SMO container object.

Due to time limitations it was not possible to give a quality of service characterization of the implemented software and especially the eXperimentation Model, as this surprisingly was not done in the evaluation process of the MPEG-7 reference software. Further questions that still should be dealt with include a performance characterization of the underlying XM component, whether time and resources usage rise constantly with the size of the input data and/or if it shows significant differences for certain types of visual information (e.g. black/white, etc.). This data can then be used to identify a suitable machine in terms of storage space, CPU and RAM for hosting the multi-client web service. Finally it is important to evaluate the delay which is caused by the usage of the web service technology due to its message overheat especially for transferring binary data.

To conclude, all created software is well documented (by using Javadoc) and provides a clearly structured architecture so that its components may be used in further projects e.g. using our given reference implementation of client's web service code. Although the implementation was restricted to a basic set of MPEG-7 visual features the client and the server side Java code can be easily extended for supporting adding additional descriptors. Above that it is even possible to include video and audio, as the presented template mechanism is highly flexible for including all available applications of the XM framework. The main effort however is to make additional descriptors available within the eXperimentation Model. Its default settings are known to not deliver very reasonable results and therefore one needs to analyze the header files of the descriptor classes for finding likely input parameters. All available information concerning this point is summarized in annex 8.3.2.

All developed products, their created documentation as well as an installation guideline with valuable hints for setting up the system are contained on the CD of the Diplomarbeit. We will now have a detailed look at the design and the functionality of the different components.

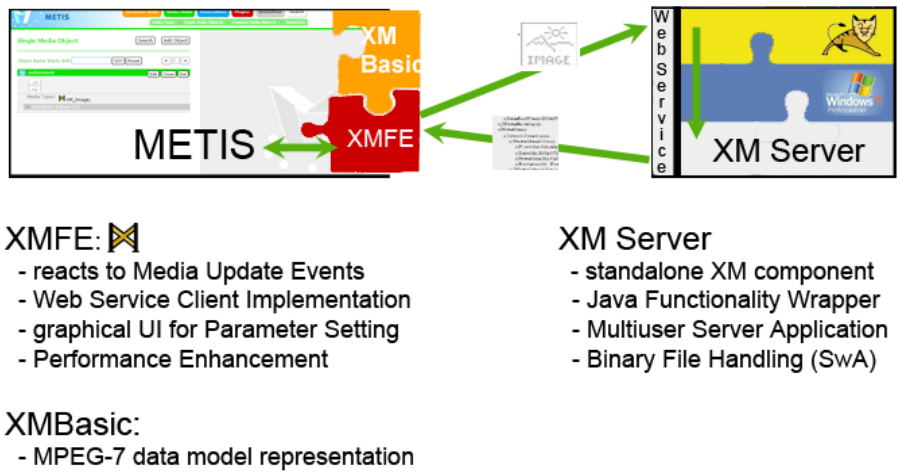

Figure 14: Simple Illustration of the System - Global Architecture and its Components

7 The XM Server - offering MPEG-7 Feature Extraction as Web Service

7.1 Requirements Description

The main goal of the XM Server solution is to make the visual part of the XM, i.e. the feature extraction of still images with the basic descriptors for color, shape and texture, easily available for a number of various applications. It is intended to be as comfortable to use as just uploading the images and receiving the description results of one's choice.

Therefore the first stage of the implementation process requires a Java wrapper for the eXperimentation Model which offers methods for configuring the tool, for automatically setting up its infrastructure (as parsing the input data, creating parameter, batch and list files, taking care of the file structure, etc.) and finally for launching the extraction process. Beyond mapping the XM functionality to a Java interface, a multi-user server architecture is necessary to enable the autonomous and concurrent usage of the software. It must be capable of administrating the clients, their data and settings as well as handling the extraction results. The web service technology is used as glue to guarantee a platform independent service, which on the one hand allows to automatically offer standardized interface descriptions of available methods and parameters (WSDL) and on the other hand to execute remote procedure calls by using SOAP for exchanging XML data.

7.2 Software Architecture

A machine running as XM Server assembles the functionality of three different software components: A preconfigured and stand-alone instance of the MPEG-7 reference software implementation XM, a Java wrapper of the software embedded in a multi-user server architecture and finally Apache AXIS for automatically generating the server skeleton (as well as the client stubs) making the system available as web service and handling the SOAP message communication. The main difficulty is to externaly couple the independently working XM framework to the Java wrapper software, which is done by making use of a template mechanism that we will present later on.

In this section we give an overview of the systems design by providing the main design ideas and software paradigms the implementation was based on and outline the architecture by using UML diagrams (Unified Modelling Language) for describing the source.

For an installation guideline of the software and its required components including hints on starting the server see appendix 8.3.2. Because mpeg7Extraction web service is deployed with the scope of an application on the server machine - and not on session basis, in the context of Apache AXIS, it is necessary to execute a small setup program (the SetupXM.SetupServerProperties.java) creating the property file the web service is going to listen to, before the service is launched for the first time. At the moment the only configuration which is done by this helper class is to set up the path on the local file system where the XM is located. However, because this information is mandatory to the mpeg7Extraction service, it will terminate, throwing a corresponding error message, until this step was performed.

7.2.1 Modifying the XM File Structure

Generally speaking the infrastructure on the local file system is rearranged to provide a higher degree of structure for the XM software and its components and to enable the wrapper classes to build a multi-user client implementation on top of it. Therefore a second top-level element is added to the directory tree next to the mpeg7 one which hosts the MPEG-7 reference software implementation. It is called pluginMETIS and contains sub-folders for input data, the extraction callers, the MPEG-7 Visual Descriptor configuration settings as well as one for capturing the resulting outputs of the extraction process. For all available descriptors, parameter and batch file templates have been created. These files are parsed by the web service which uses this data to setup the input for the feature extraction process in an - to the calling client - adapted form. This simple solution has proven as very valuable due to the fact that it provides a well defined and modular interface from the underlying XM to the service wrapper which is adaptable to changes in the underlying software (e.g. the number of supported descriptors) very quickly. On the other hand by always parsing the current templates, the service output can automatically be modified

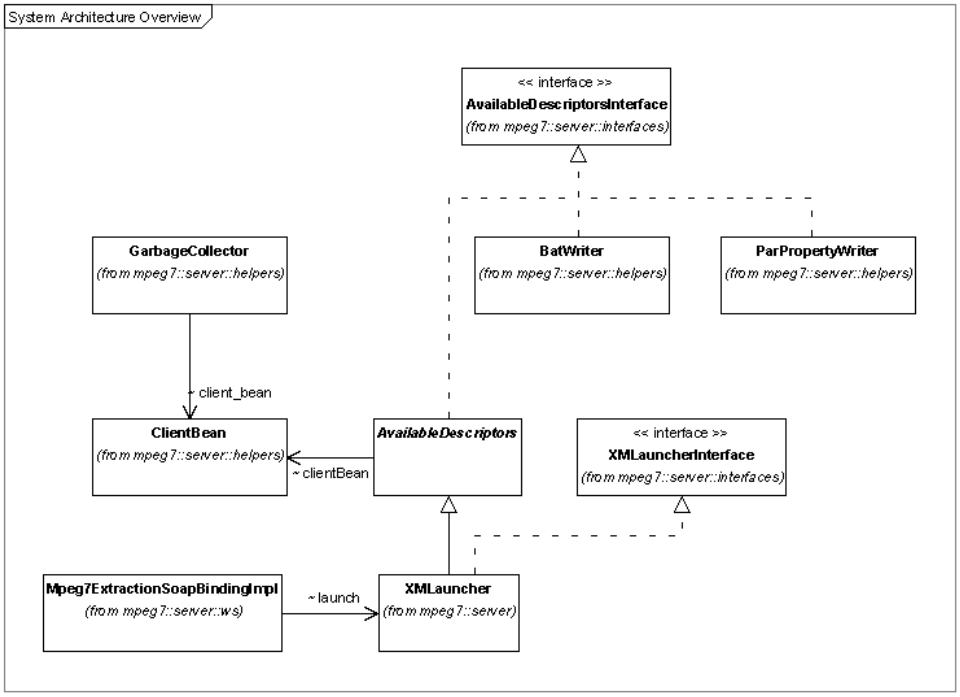

Figure 15: General System Architecture of the XM Web Service Wrapper - an overview using an UML class diagram

(e.g. which logging information is offered to the client) without having to change the source code of the XM Server.

For further details on how this file structure is used by the web service wrapper see section 7.2.2.5. Information regarding the XM setup as well as its input parameters (including an interpretation) for launching XMMain is given in 4.3.3.

Note, that although the resulting structure is optimized for external resource and user management, it still contains a complete standalone compilation of the XM, which may be used independently from the web service. However when the created batch file templates for triggering the feature extraction applications are used separately, the depth of the pointers has to be modified. All of the pointers are currently adjusted to the XM root directory.

7.2.2 The XM Java Wrapper

The design of the wrapper implementation follows the approach of being clearly structured and easily extendable. Therefore its architecture is split up into four packages *server*, *server.helpers*, *server.interfaces* and *server.ws*. The core

methods for images handling, descriptor setting, feature extraction as well as for returning the extraction results are located in the *XMLauncher* Java file in the server package. Because the XM is an external application, which is configured and setup independently of the *mpeg7Extraction* web service, it is important to keep track of changes and to offer a structure allowing modifications to the wrapper software very easily if necessary. This is done by providing the abstract interface *AvailableDescriptorsInterface* which is implemented by all classes that depend on the underlying XM. The steps which are necessary to extend the software for further MPEG-7 descriptor support are presented in the software's documentation in detail.

An overview of the system's main components as well as their dependencies is given in Figure 15 by using an UML class diagram.

7.2.2.1 Registering a Client to the Service When a client accesses the service for the first time, it is requested to register itself to the server so that an individual working space can be created. As key, a randomly chosen number of eight digits is generated, which also names the client's root directory. As long as a client is registered to the service - this only has to be done once - it has access to all of its methods for uploading images, configuring the applied descriptors or triggering an extraction process. Note that the implemented XMFE METIS Plugin, which is designed as a client instance for the current web service version 0.96 by offering fully support for the methods in the release, automatically takes care of the initial handshake procedure and the administration of the clients identification key which is a mandatory input parameter for most operations. Section **8.2** deals with the description of the XMFE architecture and gives further information on its design goals.

7.2.2.2 Uploading Images and remote File Handling Due to the fact, that using the XM requires to having the image data locally available, the mpeg7Extraction web service provides three possibilities for transferring data to the server machine. Two of them, the *receiveAndStoreFile* and the *receiveAndStoreDir* method, directly upload image data to the server's workspace by making use of Soap With Attachments (SwA), an efficient solution for including binary data in SOAP messages. Therefore a client must wrap the image source into a javax.activation. DataHandler object, which is the format that AXIS uses to attach data to a javax.xml.soap. SOAPMessage. After the message is sent over the wire the binary attachments are then automatically extracted from the message and stored in the WEB-INF/attachments folder in the context of the web service container (e.g. Apache Tomcat) by AXIS. Both methods are responsible on the one hand for moving the file from the service container into the XM infrastructure on the file system (called pluginMETIS\input_images) and on the other hand for renaming and sorting the file according to the information the client passed along. The third possibility of transferring data into the client's workspaces is *receiveAndStoreHTTPFile*, where instead of actively uploading data, a string object is handed over, representing the Uniform Resource

Locator (URL) of a remote file. After conformance testing the java.net.URL class is used for opening an incoming stream and downloading the source to the client's directory.

For details on the difficulties and restrictions of transmitting binary data via SOAP along with the suggested solution on the basis of Soap With Attachments (SwA) see section 5.2. It gives an introduction to the topic as well as a comparison to other alternative implementations.

In the process of evaluating the MPEG-7 reference software, we tested the XM and its libraries to see if it properly worked with files of the type *jpg, gif, png, tiff and bmp*, which therefore are accepted as incoming attachments by the server. Other files - or images of the right type though using other extensions - are discarded by the methods for data handling automatically. However all client implementations are strongly recommended to use the *getAllowedFileExtensions* methods to request a list of supported data types for already filtering out invalid material on the client's side. This saves needless traffic as well as valuable server resources.

As the server implementation is meant to be capable of reusing uploaded image material, a simple infrastructure for remote file handling is required. Although the solution only needs to offer basic commands, like grouping and deleting, a flat file structure would not suit the requirements. Therefore the idea of a keyword classification system is implemented , which creates clear structures because it only needs a single hierarchical directory level and at the same time perfectly fits the requirements for remote file handling by making its methods for listing, grouping and deleting accessible in an simple and very efficient manner. When a client uploads a file it adds a classifier along with the data (e.g. animals) that from then on is used in combination with the file name to uniquely identify it. Methods for remote data handling mainly cover removal operations like *removeDir*, *removeFile* as well as *removeFiles,* which allows specifying a list of images which can even be located in different subfolders. Note that the connection of a classifier[i] and a filename[j] will not be deleted. Finally *getClientsFileList* is used to request a view of the remote file structure, whereby the classifier may be used as qualifier to restrict the resultset. Since data handling on the multi-user server only had secondary importance in the software development process because of the fact that the XMFE plugin due to its requirements can hardly make use of it (in most cases it is reasonable to calls *removeAllFiles*), the server implementation only offers a basic set of mandatory file operations. Due to the chosen design approach, however, it is possible to adapt and extend the functionality very easily.

7.2.2.3 The usage of a Client Bean To achieve a strict separation between the model representation and the controlling instance (an object of the XMLauncher class) the approach of Java Beans is introduced. A single object of the *ClientBean* Java class is instantiated in the AvailableDescriptors class and from then on lives as long as the application is up. The bean is responsible for

encapsulating all client specific information like e.g. the selected descriptor set being applied in the extraction process or the timestamp information when a certain client last registered to the system or performed an action on it. Other objects that want to make use of this data are offered setter and getter methods to obtain or modify the bean's information. The internal implementation of the bean makes use of the java.util.Hashtable to store the information in a efficient way. Methods which have to deal with concurrent data access are monitored by using the Java keyword "synchronized".

7.2.2.4 Adding and Removing Descriptors The current version of the XM Server offers support for the MPEG-7 Visual features of ColorLayout, ColorStructure, DominantColor, EdgeHistogram, HomogeneousTexture, RegionShape and ScalableColor. TextureBrowsing however is supported but not recommended to be used due to its instable implementation in the eXperimentation Model. A client is offered this broad spectrum of MPEG-7 Visual Descriptors for image characterization as well as an interface enabling it to decide which features get applied when "extract" is called. Therefore every client may make use of the method *getAvailableDescriptorList* to ask which features are currently offered by the web service and then in- or exclude certain ones by using the descriptor's add or removal methods. The second possibility is to call *addDefaultDescriptors* which groups a set of recommended descriptors to a combination for easily handling them. The settings are stored in the client bean and are therefore persistent for the client's lifetime on the server. Therefore the selection only has to be made once and from then on gets applied for every call of "extract". Nevertheless it still is adjustable at any time. The current configuration can always be retrieved using the *getAddedDescriptorList* method.

Note that the implemented software only manages the descriptor handling for its clients. Making new descriptors available in the XM infrastructure is an autonomous task. We will give a short explanation on how to do this later on in this chapter and show how easy it is to extend the wrapper software for supporting new features.

7.2.2.5 Extracting MPEG-7 Features Along with the functionality for uploading images and data handling, the MPEG-7 feature extraction is located in the server.XMLauncher Java class. Note that due to the design of the XM software, which is confusingly structured, rarely commented and widely spread on the one hand and implemented (besides in a generic language) in plain C++, it is not able to directly map the algorithms to Java, nor to directly attach to an interface of the software. Therefore the decision was made to view the XM as a homogeneous system. For offering the XM functionality as a Web Service, a proper wrapper implementation is built around the "black box", operating on the local file system to set up the required infrastructure for triggering and monitoring the extraction procedure. The steps, which are necessary to launch an MPEG-7 feature Extraction from the web service by using the underlying

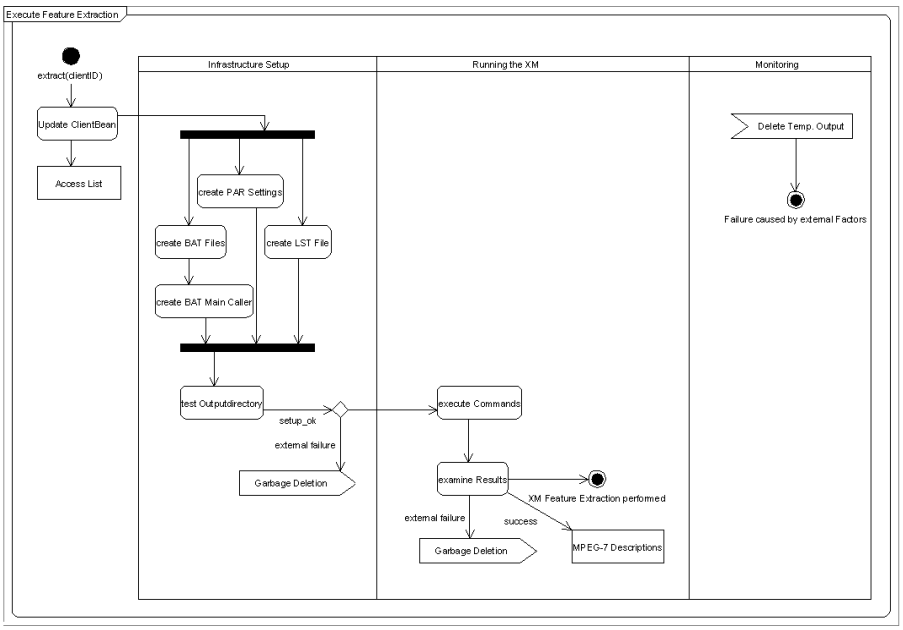

Figure 16: Sequence of Acivites when calling the MPEG-7 Feature Extraction - using an UML Activity Diagram

and pre-configured eXperimentation Model, are shown in Figure 16 and are now discussed in detail.

When the method *extract* is called, the first thing to do is to update the client's access list, which is necessary to inform the server when a registered client requested a service for the last time - this is done by all available public operations. Afterwards the client's workspace is cleaned from former description results and parameter settings before user specific infrastructure for calling the XM feature extraction is set up. The setup procedure involves the following points:

1. *Creating The .par Settings:* The configuration of the underlying software is an essential task since the resulting quality of the descriptions is highly dependent on it. As described in section 4.3 the available documentation dealing with this point is rare and a far from providing a complete picture of its possible settings. Due to the complexity of this task the decision was made to either use the given, predefined settings or expert knowledge to configure the infrastructure on the server machine, but not to offer the normally unexperienced users a generic interface, which would be necessary to fit the requirements for changing these settings.

The software makes use of a *template mechanism* for integrating these settings. Therefore a template, following the conventions for naming, must be available in the "default_desc_settings" directory for every descriptor that shall be accessible via the web service. This has the advantage that the configuration can be changed at any time without having to go into the wrapper code and in addition it immediately affects the results even when the service is up and running. A private method of the XMLauncher class works thorugh all registered descriptors using Java reflection to add the selected ones. Therefore it uses an instance of the ParPropertyWriter Java class, which is located in the mpeg7.server.helpers package, to parse the templates, to find and replace the client specific information in the setting (i.e. the name and location of the input images and the file names for the resulting output) and to automatically write the modified data to the client's *parameter (.par) files*. These records now contain pointers to the client's workspace and consequently can be used as input for the extraction process later on.

2. *Creating The .bat Files:* The next step in the infrastructure setup is to combine the correct parameters with the requested XM feature extraction applications, which are both input parameters of the XMMain.exe. Due to the fact that no direct connection between the wrapper implementation and the eXperimentation Model exists - as both are independat of eachother and external components, logging is a very important task to determine the success of an extraction procedure. So, very similar to the template structure of the parameter settings, a default arrangement of commands for all available descriptors is created in form of *batch files*. On the one hand they handle the formation and redirection of the logs and on the other hand contain the actual program calls, combining the proper extraction applications with their corresponding parameter configuration. Batch files in Microsoft Windows are characterized with the file extension ".bat" and make it possible to give a lists of commands and/or to specify programs to be launched once the batch file has been executed.

   These template documents are read when *createBatFiles*, as part of the extract method, is called. Their references are then adopted to the client's needs and directory structure by using the BatWriter helper object from the mpeg7.server.helpers package. The resulting data is written to the client's workspace using the naming convention "extract_" followed by the descriptors name. Due to the fact that feature extraction is considered and implemented as a definite operation based on a single thread structure per client, it is necessary that all requested descriptions are finally connected by a master batch file, forming a processing chain.

3. *Gathering the Input Data:* As last part of the infrastructure the XM framework requires a list of images on which the feature extraction shall be performed. This file contains a list of included file names (relative to the root directory), separated by a new-line delimiter. As this information

is made use of in the parameter settings - a reference already was created in step one - the method *createLSTFile*, which iterates on the local file system collecting the visual material, is located in the ParPropertyWriter helper class. All files of the client's workspace are automatically included for feature extraction, when their file extension is supported by the XM-Launcher class. Currently the web service in the version 0.96 was tested with jpg, gif, png, tiff and bmp.

4. *Creating a Shell and Invoking the Feature Extraction:* After the setup of all required data structures for the client's requested descriptors is completed, the data is available on the local server's operating system in form of files. Its structure exists in a form which also could be executed by hand to run the XM feature extraction process - and actually nothing else is done by the web service wrapper. Therefore it invokes a shell, moves to the working directory of the XM service and runs the master batch file, to launch the MPEG-7 feature extraction. The software uses a java.lang.Process object for executing commands on the system's runtime environment. As we described above the extraction is implemented as a definite operation, based on a single thread architecture. This is achieved by opening an input stream for reading the frameworks shell outputs and logging them simultaneously in a temporary file. This has the advantage that the independent feature extraction process running on the server's operating system is synchronized - the opened output stream forces the object to wait until a definite result, i.e. the MPEG-7 feature descriptions, exists. Although this possibility for tightly coupling the two independently working systems on the server side is very important for excluding interference and for checking extraction results for completeness, we however noticed that it is probably a good idea to either use a multi-threading client architecture or to support asynchronous notification by offering a push service. In both cases the time for waiting on the server to perform the feature extraction could be skipped.

Reliability is a priority - therefore at every stage of the infrastructure setup the outputs are monitored. Only if each step is carried out successfully, the method proceeds to the next one. After the XM finishes its work the resulting descriptions are examined if all requested features are successfully available or whether a time-out occurred somewhere. Finally, it must be mentioned that the system's architecture is built to support multiple simultaneously running extractions processes by different clients and thereby is only limited by the underlying power of the server machine and the capability of the eXperimentation Model and its components. The weaknesses and problems of this solution are presented later.

7.2.2.6 Returning the MPEG-7 Description Results Because all information the feature extraction process depends on, that is the descriptor configuration and the list of images the XM applications are applied to, as well

as the extracted results themselves, are stored as records on the server's file system, the client can access this data even if the descriptor settings were altered or image material was uploaded or deleted in the meantime. Logically this corresponds to the idea of a transaction. Although the resulting descriptions are available until the method extract is called again, there is unfortunately no level in the server's architecture that deals with caching of results, as this was assumed to be implemented by the clients. However as running the XM is a costly process in terms of occupied resources capacities and needed time - compared to the relative modest requirements for storing an image hash as key along with its corresponding textual MPEG-7 descriptions - it definitively is reasonable to add a caching functionality in the next software release.

The output of an XM extraction process is stored in XML files, containing all requested descriptions for a certain feature in form of collections. As the framework groups results per descriptor and not per image (i.e. when image A requests descriptor D1 as well as D2 to get applied, then the results for image A are stored in files called D1.xml as well as in D2.xml and not in a single document containing both features for A) the MPEG-7 *DescriptorCollectionType* is used to include various results.

Currently the software offers two methods for requesting MPEG-7 image characterizations - both are located in the XMLauncher Java class. They either return extracted data of a certain descriptor (like *getExtractionResult*) and therefore take its qualifying string name as input parameter or ask for all available results (like *getExtractionResults*) which are packed into a java.util.Hashmap to transfer the data to the client. In the latter case the qualifying names are used as keys in the hashtable to retrieve their corresponding data. The qualifying descriptor names are located in the mpeg7.server.AvailableDescriptors class which is extended by XMLauncher.

Retrieving extraction result(s) per image query (e.g. per classifier and filename or per source) is currently not supported as there was no need to provide this feature for the METIS client plugin. However, when offering the service to a broader community this would probably be the best possibility for requesting specific description results because on the one hand it is comfortable in its handling and on the other hand connects data that logically belongs together in a transparent way. When this idea is taken further, it probably ends in an operation implementing the idea of a one-stop-shop, taking an image object and the applied descriptor names as input parameters and returning an aggregated description. As this is only interesting for a limited number of clients which do not take advantage of remote file handling and multiple image extraction, it must be discussed whether to implement this feature on the server or rather on the client side.

Example 1 presents a code snippet of a simple client implementation where all typical steps from of uploading data, to starting an extraction process and finally requesting the extraction results are shown. However for some application scenarios this may be satisfactory - clients that want to make excessive use of

the systems remote file handling or multiple image description extraction are recommended to add a further abstraction layer, dealing for example with the persistency of the clientID or the separation of the returned XML data, to extend the server's low level functionality in these points.

Example 1 *{*

```
// Create a service object
    mpeg7.cl.XMLauncherService service =
            new mpeg7.cl.XMLauncherServiceLocator();
// Now use the service to get a stub to the service
    mpeg7.cl.XMLauncher xmserver = service.getmpeg7Extraction();
.
    System.out.println("Basic mpeg7Extraction Client");
//Register to the web service - workspace is created
    String s_clientID = xmserver.generateClientID();
.
//request allowed file extension
    file_types = xmserver.getAllowedFileExtensions();
    [...]
//upload data
    boolean b_success = xmserver.receiveAndStoreHTTPFile(
          "http://domain.at/ferrar1.jpg","cars",s_clientID);
                //b_success = xmserver.receiveAndStoreDir(..);
                //b_success = xmserver.receiveAndStoreFile(..);
    System.out.println("upload successfully?: "+ b_success);
.
//add the features you want to have extracted
    xmserver.addDefaultDescriptors(s_clientID);
    xmserver.removeColorLayout(s_clientID);
.
//trigger extraction procedure
    b_success = xmserver.extract(s_clientID);
   System.out.println("MPEG-7 feature extraction successfully?: "+ b_success);
.
//retrieve the results
   String[] desc_names = xmserver.getExtractedDescriptorNames(s_clientID);
   String s_DomColorXML = xmserver.getExtractionResult(desc_names[1]);
    String[] s_from_files = xmserver.getExtractedFileInputList(s_clientID);
//display the extraction result
    System.out.println(s_DomColorXML);
.
//execute remote file handling operations
    String[] list = xmserver.getClientsFileList("vacation",s_clientID);
    xmserver.removeAllFiles(s_clientID);
}
```

7.2.2.7 Error Handling As the Java wrapper builds an interface to an external software system, there are many critical points where errors can occur. Therefore it is important that all methods which manipulate data are designed to be definite. When errors or faults occur it must be possible for an affected user to determine the exact problem and to judge whether its nature is termporary or heavy weight (e.g. enabling the software to be monitored remotely or to respond with a problem-specific reaction).

Generally speaking errors have influence on the boolean outcome of a method, failures and exceptions on the other hand are separately dealt with by using the org.apache.axis.AxisFault for throwing exceptions. It is not necessary to derive specialized objects from that class for different failure situations as the AxisFault - which generates a SOAP fault message - offers sufficient possibilities for characterizing them. The mpeg7Extraction web service implementation mainly made use of enclosing their occurrence with a textual depiction including the location and environment it was determined.

7.2.2.8 Garbage Collection A prominent example of an object making use of the ClientBean is the *GarbageCollector* Java class. In the decision process of offering the mpeg7Extraction web service with the scope of an application rather than on session basis, the following considerations were taken into account. First of all the task of uploading binary data to the server machine, where the web service resides, is assumed to be a very resource and cost intensive task in terms of bandwidth consumption and time (e.g. due to the data and message overheat which is needed for SOAP communication - the results of the performance measurement give a proof 5.2.1). Further, on the one hand depending on the client's infrastructure or application domain it may not be possible to have the image data cached for further feature extraction (e.g. to resource limitations) or on the other hand caching description results may not be desired due to the provided possibilities of easily requesting specific results and combining various media data into an extraction process. In both situations it is desirable to save network as well as server traffic and to store the client's settings, extraction results as well as its uploaded image data for a certain amount of time.

Therefore garbage collection is a very essential task to keep the system alive. The GarbageCollector class which is contained in the server.helpers package is responsible for cleaning up the file system from image material of expired clients, from automatically created .bat and .par files, which are used to set up the infrastructure for running the XM, and finally from extraction results and log files. By extending the TimerTask the GarbageCollector operates as an independent Java thread - an initial startup cleanup is performed every time the web service gets restarted (e.g. after rebooting the web service container) where all leftovers from previous sessions are discarded. A repeating cleanup procedure is started every 30 minutes,.removing clients that did not access the service during the last three days, as well as their workspace,

7.2.2.9 Summary The created XM Server makes the feature extraction part of the eXperimentation Model (XM) available as web service - called mpeg7Extraction - to provide its users with the broad spectrum of MPEG-7 Visual descriptors for image characterization. Therefore it assembles the functionality of three different components: the MPEG-7 reference software, the implemented Java wrapper code and Apache AXIS. The Java wrapper software on the one hand makes use of the preconfigured and underlying standalone component, the XM and its descriptor applications, by automating the required infrastructure setup. On the other hand it offers multi-user functionality e.g. for image handling, feature configuration, garbage collection etc. The created template mechanism permits defining settings (e.g. for logs, parameter configuration) independently for every client and at the same time enables a local expert configuration of the XM without having to change any web service code. As third component Apache AXIS is used for separating the wrapper code from SOAP, as AXIS manages all message exchange and object stub generation. For uploading binary data, Soap with Attachments (SwA) is used to enhance performance. The WSDL file, describing and defining the service's interface, may be used by a client to automatically implement its stubs for accessing the remote service. The MPEG-7 conform extraction results are valid XML documents, containing the feature characterizations of the requested images.

7.2.3 Using AXIS for Automatic Stub Generation and SOAP Traffic Handling

The SOAP engine Apache AXIS does not only offer libraries which are capable of handling the complete message creation and exchange procedure, but also provides a set of helpful tools for creating a web service from scratch. The software is capable of taking any piece of Java code, wrapping it up as a web service and deploying it to the AXIS system. A remote client can then automatically generate the stubs for accessing the service, by only requiring its WSDL file. For background knowledge on Apache AXIS as well as the web service technology see chapter **5**.We will now give a short introduction on how these tools may efficiently and easily be used for creating web service code and its description documents by looking at our implemented mpeg7Extraction service. [33]

7.2.3.1 Java2WSDL - generate a WSDL description from existing Java code This program which comes in the "org.apache.axis.wsdl" package, is able to generate a WSDL description from any given piece of Java code. It must be called with the following parameters: the name of the resulting WSDL file, the URL where the web service shall be reachable, the target namespace for the WSDL file, a mapping between the Java package and the specified namespace and finally the fully qualified class name itself. Remark 2 gives an example on how the call of Java2WDSL would look like for the mpeg7Extraction application. The public methods of the XMLauncher class are mapped to the platform independent WSDL file, describing them in XML syntax. Although it is not difficult to manually construct the WSDL description as the structure

is simple, the tool though drastically reduces the time effort for creating the 1483 lines of XML data which are necessary for characterize our service (it is depicted in annex 8.3.2).

Remark 2 *java org.apache.axis.wsdl.Java2WSDL -o XMServer.wsdl -l"http://localhost:8080/axis/services/mpeg7Extraction" -n urn:mpeg7Extraction -p"mpeg7.server" urn:mpeg7Extraction mpeg7.server.XMLauncher*

7.2.3.2 WDSL2Java - for automatically generating Web Service code

After the service was defined in step one, the next task is to generate the code for deploying and accessing it. The input parameters for the org.apache.axis.wsdl. WSDL2Java class are: the base output directory, the scope the service is deployed as (e.g. application, request or session), whether server or client side stubs are generated, the Java package to place the code and finally the name of the input WSDL file. Note that the mpeg7Extraction web service is deployed with the scope of an application. It is a good idea to choose a different output package for clearly separating the application from the web service code. The resulting data structure after running the tool is depicted in Figure 17. The ws.XMLauncher class presents a remote interface to the XMLauncher system, as the methods of the original class must be able to throw remote exceptions. The XMLauncherService.java contains the service interface of the web service and therefore is implemented by the XMLauncherServiceLocator for retrieving a handle on the system. All server side stub code is contained in the MPEG7ExtractionSoapBindingStub class.

However all these files are generated automatically and do not have to be dealt with at all. The only code which needs to be modified is the ws. Mpeg7ExtractionSoapBindingImpl containing the implementation code for our web service. Since we must connect our developed XM wrapper software, i.e. the mpeg7.server. XMLauncher.java, to its binding class, all incoming requests are forwarded to the identically named methods of an instantiated XMLauncher object which then processes the request and returns its results to the binding class. An extract of the Mpeg7ExtractionSoapBindingImpl.java file, demonstrating this simple linking mechanism is given by the following example. However note that the presented steps must be performed anew, every time the method's interface of the wrapper software is modified because no direct coupling between the generated web service code and the application code exists.

Example 3 .

```
/**
* Mpeg7ExtractionSoapBindingImpl.java
*
* This file was auto-generated from WSDL
* by the Apache Axis 1.2alpha Dec 01, WSDL2Java emitter.
*/
.
```

```
package mpeg7.server.ws;
.
import mpeg7.server.XMLauncher;
.
public class Mpeg7ExtractionSoapBindingImpl implements mpeg7.server.ws.XMLauncher{
        XMLauncher launch = new XMLauncher();
  public boolean extract(java.lang.String s_clientID) throws java.rmi.RemoteException{
        return launch.extract(s_clientID);
   }
.
   public boolean receiveAndStoreFile(javax.activation.DataHandler dh,
      java.lang.String s_sentFileName, java.lang.String s_projectName,
     java.lang.String s_clientID) throws java.rmi.RemoteException{
       return launch.receiveAndStoreFile(dh,s_sentFileName,s_projectName,s_clientID);
   }
.
  public boolean receiveAndStoreDir(javax.activation.DataHandler[] dh, java.lang.String[]
      s_sentFileName, java.lang.String s_projectName, java.lang.String
     s_clientID) throws java.rmi.RemoteException{
       return launch.receiveAndStoreDir(dh,s_sentFileName,s_projectName,s_clientID);
   }
[...]
  public java.lang.String getAvailableDescriptorList() throws java.rmi.RemoteException{
        return launch.getAvailableDescriptorList();
   }
}
```

7.2.3.3 Deploy the service to AXIS Finally to complete the procedure of generating the web service code, we need to hand over a deployment descriptor to the AXIS system for setting up the service. This may either be done by executing the helper class java org.apache.axis.client.AdminClient, after the AXIS server is up and running, or the java org.apache.axis.utils.Admin. Both methods require the "deploy.wsdd" - an automatically created product resulting from calling WSDL2Java and therefore is found in the ws package - as input data.

7.2.3.4 Summary Summing up, Apache AXIS and its included tools are capable of setting up a web service from scratch by offering stub generation from existing Java code, the creation of WSDL service description files and taking care of all message handling. A user taking advantage of the toolset does not require detailed knowledge of the SOAP specification anymore and therefore can fully concentrate on the actual application development.

As hint for easily making use of AXIS in an round-trip engineering process, we recommend organizing all input parameters for calling the offered helper classes into batch files, so that the steps of extracting the description file, generating the web service code, compiling, packing and moving the created .jar

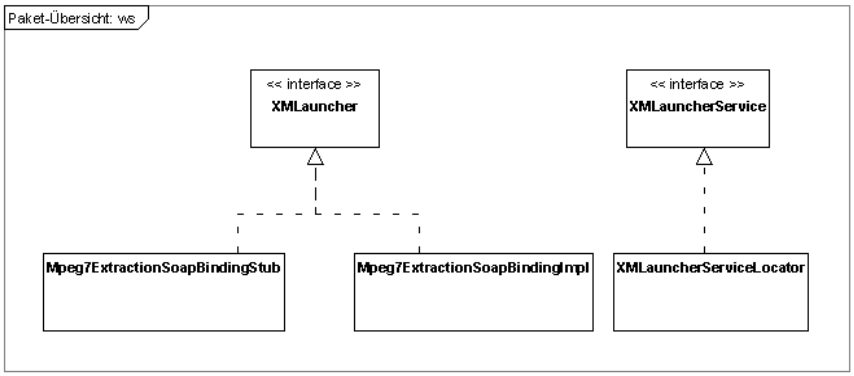

Figure 17: Using the AXIS tool WSDL2Java for Generating Web Service Code - an overview on the resulting class structure

file into the AXIS library as well as finally deploying the service can be done by just simply executing these files.

7.2.4 Problems and Limitations

Although we have already pointed out several limitations and problems earlier in the paper that the solution currently suffers from, we want to explicitly summarize the most important ones and present conceptional solutions for supporting and improving further implementations.

Performance Improvements and Monitoring the XM One of the main advantages, namely the implementation as definite state machine, is actually at the same time responsible for a lack of performance. As the Java wrapper software is based on an external component, with no direct connection between these elements, this architecture is essential for excluding interferences. However since using the XM, for applying the descriptors to image data, is a very cost intensive task in terms of computation time, it is not a good decision to let the client wait for the results. Therefore either a push architecture on the server side or a multi threading client implementation would achieve better performance.

The idea of a client side cache, as it is introduced in the XMFE plugin architecture, can furthermore be an applicable solution for certain application scenarios to improve performance at very modest costs compared to running a remote extraction process. In certain situations even a server side cache could deliver reasonable results, depending on whether the clients are using domain specific image material, if a client performs feature extraction several times on the same files (e.g. in different collections) or requests a subset of already

extracted descriptors. However, the task is to evaluate the optimal size of the cache, to choose the best suited form of persistency and to use Java objects that work efficiently on a large amount of data (as e.g. java.util.Hashtable does).

Due to the fact that the underlying eXperimentation Model and its contained libraries (e.g. for image handling) cannot be classified as highly reliable (e.g.Texture Browsing is generally very slow and unnstable for large images), the garbage collector functionality should be extended to monitor time-outs and terminate the invoked shell. As no quality-of-service parameter for the XM exists, it further probably is a good idea to install a pool of maximum simultaneously running extraction procedures to limit the occupied resources. However this point is highly dependent on the application scenario the web service gets deployed on. Due to our requirements definition this has not been a problem yet.

Extending remote File Handling At the moment all image data contained in the client's workspace is included in an extraction call, as long as its file types are supported by the software. This is definitively not a good idea and violates the design approach of offering remote file handling based on a classifier system, for saving network resources and offering support for clients with restricted hardware facilities (e.g. terminals or PDAs). Therefore mechanisms for restricting the query input, as they are offered e.g. for requesting a remote view on the file system, are demanded.

Even more important is to discuss the idea of a one-stop-shop solution, taking image data as well as a set of descriptors that are supposed to be applied to the data as input parameters and returning the extracted results to the requesting party. At the moment the web service offers methods to request all required data for assigning description results to its corresponding data for separating the MPEG-7 collection type. The application scenarios taking advantage of a query-by-image solution or whether an extended client implementation should provide this functionality were discussed in this chapter.

Operation System dependent Code All tools and software technologies the server as well as the client side are built on (e.g. the Apache tools and Java), are available in a platform independent form and resp. publically available under a license of the open-source software community - only the compilation of the XM framework and its libraries (e.g. for importing images) are platform dependent. The eXperimentation Model is available for Microsoft Windows 32-bit operation systems (OS) as well as for Linux and must be modified accordingly for supporting the service's directory architecture. Because our wrapper solution uses Java file objects to set up the required infrastructure on the server machine for launching different applications of XMMain.exe, batch files are used - a solution that is available on every operating system. Therefore the only OS specific mechanism contained in the wrapper code deals with invoking a shell and executing operations on it in a platform dependent syntax.

However the architecture of using a template mechanism for configuring the XM framework externally allows easily adapting the wrapper software to platform specific conditions.

Security Finally note that adding support for security features has not yet been a topic. However as the solution identifies its clients by using a randomly generated ID number this mechanism could easily be tampered with and the client's workspace including its sensitive data could get altered. When offering the service to a broader community this topic definitively has to be dealt with e.g. by handing out some kind of certificate.

8 Adding MPEG-7 Support for METIS Objects

8.1 Requirements Description

Content based image retrieval (CBIR) in METIS is supposed to be supported in terms of either identifying duplicate media instances attached to SMOs or offering a search-by-image query system for retrieving similar multimedia content, spread over the media repository. However, building an automatic or semi-automatic CBIR system (e.g. by using MPEG-7 description schemes that includes an arrangement of carefully selected feature characteristics getting weighted according to a given environmental situation - and by offering a satisfactory interface for user interaction) is a non-trivial task and was defined to be out of the scope of this work. Therefore the requested implementation is restricted to providing support for a basic set of MPEG-7 Visual Descriptors. The solution shall therefore on the one hand contain a client implementation for the developed web service with extended functionality for data handling and retrieval and on the other hand meet the requirements of a visual METIS plugin. It subscribes and reacts to events for triggering a remote feature extraction process by not violating the transaction mechanism and finally attaches the results to a SMO container object. This software development includes the creation of an METIS heavy-weight plugin, and the construction of a semantic pack - a model, capable of capturing and integrating the extraction results by corresponding data types.

8.2 XMFE Plugin - an extended Client Implementation

8.2.1 Basic Introduction to important METIS Classes

The design approach of the multimedia middleware platform METIS and its main architectural components was already described in chapter **2**. We will now give an overview of the technical background, which is necessary to understand the XMFE (eXperimentation Model Feature Extraction) plugin architecture.

In short, plugins are used for extending the core functionality of METIS and eventually its web-interface. Plugins are METIS Named Persistent Objects (MPO's) which are required to have certain metadata (e.g. a namespace, which is used as identifier, information about the author, an icon and they exist in a certain version) and include the actual plugin classes, Cocoon XSP pages and 3rd party libraries in their file structure. Further plugins may depend on a certain underlying data structure (i.e. a semantic pack) or even on other plugins and may be enabled/disabled during runtime without having to be uninstalled. For communication with users or for administration purposes it is possible to extend the METIS web-inferface. These so called visual plugins are allowed to use the METIS GUI resources (as images, stylesheets, logic/gui tags, etc.) and define their own XSP pages and Cocoon sitemaps which then automatically get mounted. Other communication is either done by using the implemented

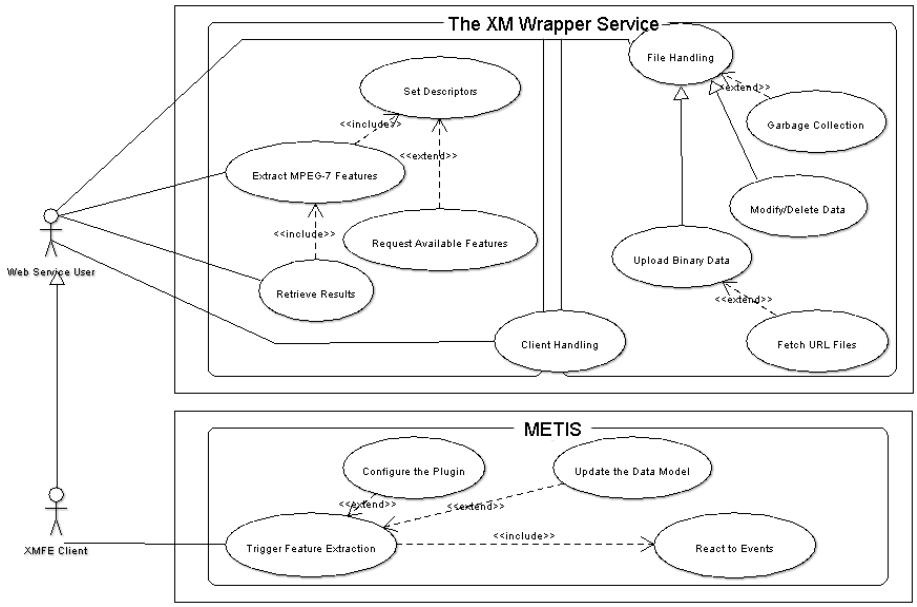

Figure 18: The XM Wrapper Service and its METIS Client XMFE - a use case overview

event mechanism, the METIS API or an own interfaces for external systems communication.

The METIS PluginManager is responsible for importing (e.g. unpacking JARs, parsing plugin metadata, etc.) and administrating all plugins on a METIS instance. Therefore all plugins in METIS must extend the Abstract-Plugin class which provides a large amount of inherited functionality and is responsible for defining a structure the plugins have to implement. Besides the register and unregister methods, which are used to make a plugin available to the manager, invoke is probably the most important methods as it forms the interface between METIS and the plugin code.

8.2.1.1 Making use of METIS-PDE In order to motivate users to extend the core functionality of METIS by plugin development, the Research Studios aim at making the building procedure as comfortable as possible. Therefore a development system for creating new and for compiling existing METIS plugins and semantic packs is provided. It is called METIS PDE and based on Apache Maven, which is similar to Ant, with the difference that it offers a "standard way to build the projects, a clear definition of what the project consisted of, an easy way to publish project information and a way to share JARs across sev-

eral projects"[27] The maven targets which are offered by METIS PDE include the creation of new sempacks/plugins by automatically setting up the required directory structure and template files. It further supports the generation and compilation of plugin/sempack jars from existing projects (even of interdependent plugins) and the execution of plugin classes for testing purposes.

8.2.2 Software Architecture

The plugin's architecture can actually be divided into two relatively independent tasks, shown in the use case diagram in Figure 18. The first one is to offer a client implementation for integrating the mpeg7Extraction web service functionality into the plugin. Therefore it makes use of Apache AXIS for automatically generating its stubs. Further, Java classes are presented, containing a reference implementation for extending the service's operations by e.g. encapsulating the wrapping of required data structures or additional accessing. On the other hand the plugin's main classes and files need to follow the structure and requirements for METIS plugins for being able to communicate with the system using the event mechanism.and manipulating its elements through the API.

We will now have a look at the different architectural points in detail. For an installation guideline for seting up the system including hints for modifying the software see annex 8.3.2.

8.2.2.1 Stub Generation and Extended Client Functionality The XMFE plugin acts as a client to the mpeg7Extraction web service, requesting MPEG-7 feature extraction from uploaded image data. Apache AXIS and its WSDL2Java helper tool are used for generating the web service's client code including the stubs for accessing the remote service. Because the resulting client implementation is consequently dependent on the AXIS libraries and classes, they must be provided in the plugin's lib directory. All jars which are included get automatically added to the classpath when the plugin is compiled. The setup procedure of the client's remote infrastructure takes a WSDL file containing an interface description of the service as input parameter. This file can be retrieved very easily as AXIS is capable of generating it on the fly from a running system. Therefore just contact the service's endpoint address and attach "?WSDL" at the end of the URL for receiving the description. For detailed information about executing the WSDL2Java class see section 7.2.3.2. The generated package architecture, including the classes for invoking methods remotely is contained in mpeg7.client.cl and is found in the plugin-XMFE directory. However note, that AXIS is not necessarily required on both, the server and/or the client side, due to the fact that the web service technology offers a platform independent interface and uses standardized formats for exchanging messages. It is an autonomous decision we made to use AXIS on both application sides due to its abilities for automatic code generation and for hiding all dreadful SOAP or RPC code from the developers.

[27] Apache Maven Project, http://maven.apache.org/what-is-maven.html

Although the resulting client code of calling WSDL2Java can be used directly, without having to perform any modifications for creating a service object from the ServiceLocator class and for calling all available methods on its remote handle, it however is a good idea to add an additional abstraction layer. The implemented mpeg7.client.*XMLauncher* Java class is therefore responsible for hiding the ID management (e.g. the initial handshake procedure, providing it for every remote call and making it persistent on the client's system by using a java.util.Properties file), for correspondingly mapping the method names and server exceptions to the client's need and finally for encapsuling additional functionality or creating complex data structures (e.g. for wrapping a file into a DataHandler object). This is mainly made use of by the methods for uploading binary data to the service. The additional abstraction layer allows providing a simple interface for accessing remote methods and their functionality by just expecting a file or directory name as well as a classifier as input parameters and then automatically takes care of all data wrapping. This has the advantage that a mechanism for prefiltering images according to the remotely supported data types can be installed before the data gets uploaded to the service. Other functionality extending the service is for example offered by the "print" methods for displaying received answers of the service in a human readable form.

As we already pointed out earlier on in this work, it probably is a good idea to further provide advanced methods for grouping and arranging description results, e.g. in form of a one-stop-shop which makes it possible to logically connect the input data with its resulting description results.

8.2.2.2 The XMFE Plugin Structure and Event Mechanism The actual plugin structure by extending the AbstractPlugin and implementing the MetisEventListener is realized in the Java class *XMFeatureExtractionPlugin,* which is contained in the mpeg7.client package. When the plugin gets imported into the system and registered at the PluginManager, its "register" method first performs a check whether the required XM Basic semantic pack was already installed, as it provides the underlying data structure to store the MPEG-7 description data.

The workflow of the plugin triggers the extraction process for Media Instances (MI) that are attached to Single Media Objects (SMOs) carrying the "XM_Image" MediaType. Therefore the register method makes use of the METIS event mechanism - it creates an instance of the EventManager and subscribes to *MediaInstanceUpdateEvents*, TransactionCommited and TransactionRolledBack events.

The event of main interest however is the MediaInstanceUpdateEvent. It indicates that the data corresponding to the associated MediaInstance of this event has changed. Every time this event is thrown by the system (or a system component) our plugin is informed through its implemented notify method. Note, that a MediaInstance is just a logical instance of a media object. Hence

when this situation occurs the plugin needs to request the source of the event. It checks whether the data belongs to the supported mime types and finally adds the transaction ID to a list of monitored transactions. After a transaction is committed, the plugin gets active again and passes the data source to the protected helper method handleMediaInstance for finally performing the remote MPEG-7 feature extraction. procedure by using an object of the XMWrapper class.

For clearly structuring the software, the actual plugin code is separated from the description data handling. Therefore a helper class called *XMWrapper*.java is used. It is contained in the "plugin" package next to the XMFeatureExtractionPlugin and is responsible for mapping the configuration settings from the user interface to the client's web service methods. As the plugin only must treat a single file per process, it is not intended to make excessive use of the service's remote data management abilities. However the mpeg7.client.XMWrapper class is a good example of a one-stop-shop for offering MPEG-7 feature extraction with an simple interface and methods for easily retrieving its results.

Because of a missing possibility of getting a handle on media items of the type "database", the XMFE plugin currently offers support for the location schemes "http" and "file" only, as their reference can directly be resolved and transferred to the web service.

The resulting MPEG-7 description data is stored in the corresponding attributes slots of the underlying "XM_Image" MediaType of the XMBasic semantic pack. It however is not necessary to manually add these attributes to the SMO before attaching a MediaInstance because the software was designed to automatically create new values or overwrite existing ones with current data.

8.2.2.3 A Graphical User Interface for Configuring Applied Descriptors Although the XMFE plugin is constructed to work autonomously without requiring any user interaction, it however offers a simple graphical user interface for enabling basic configurations. A visual plugin, extending the METIS web interface is therefore required to provide logic- and stylesheets. A template of these files is created when METIS-PDE is used for building the skeleton of a plugin. These files can then be enriched with logical tags, representing application data (e.g. calling a Java object and retrieving some information from it) and visual tags (e.g. buttons). In the METIS rendering pipeline these tags are incrementally replaced with Java logic and XHTML code by using XSLT stylesheets, also called logicsheets. The resulting XSP can then be processed by Cocoon and gets displayed as HTML code in the web browser. [2]

The main possibilities for configuring the XMFE plugin via web interface are offered for including or excluding certain descriptors for being executed on the image data. Other information displayed is the location of the endpoint the client is working on and the possibility to enable GUI messages, informing the user of the success or status of the extraction procedure. All of this information is encapsulated in the *XMFeatureExtractionPlugin* class, which offers getter and

setter methods for accessing and modifying its objects. The XSLT file is used as the link between the Java program logic and the GUI interface for displaying and configuring the corresponding Java objects. As we described above, it allows invoking the plugin's Java methods (e.g. getWebServiceLocation, getAListOfSetDescriptors) and including object data into the stylesheet.

When the METIS action equals "configure" - this is triggered when the user submits data via the web interface - an instance of the Pluginanager is fetched for invoking arbitrary methods on the plugin classes. The modified configuration data which is contained in the received HTTP Request is parsed - here the mapping between the HTML fields and the corresponding descriptors is done - and inserted as input parameters for the plugins setAListofSetDescriptors and setGuiMessagesDisplayed operations. Finally note that the plugin only contains a single method for being invoked and this requests an ArrayList containing the configuration of set descriptors as single input parameter.

8.2.2.4 Caching Description Results There are two main reasons why the XMFE plugin needs to implement a layer for caching MPEG-7 description results and its corresponding media instances. On the one hand the remote task of executing an MPEG-7 feature extraction process is very costly in terms of computation time the XM requires for applying the descriptors to the image data and in terms of resource handling, as every file has to be transferred to the server machine. On the other hand METIS, unfortunately, does not currently distinguish between a MediaInstanceUpdateEvent, which is thrown every time e.g. the SMO container object is modified, and a MediaInstanceAlteredEvent to inform that the attached resource itself has changed. This leads to an excessive and needless usage of the plugin which must be prevented.

As reaction to this problem the idea of a local cache is introduced. It is located in the main class of the plugin and captures all extracted description data in hashtables objects. Every time an extraction is requested by the system, the cache is examined first whether the image data was already analyzed by the plugin and if all requested descriptors are locally available. If one of these points fails, the remote procedure is launched to extract all requested characteristics anew, otherwise the results are directly returned from the local cache to update the SMO's attribute values.

For caching the descriptors the efficient structures of java.util.Hashtable are used, as the class does extremely well in efficiently handling huge amounts of data without suffering performance deficits. As key for uniquly identifying images, a hash value is computed from the attached resource. However as the current METIS build lacks a hash-checking functionality to distinguish whether a media instance was altered or just the container object, the plugin implementation makes use of the MediaInstance.getHashValue() methods, although it is slow. Further performance improvement can be reached for example by implementing the plugin on basis of a multi-threaded architecture, waiting on the extraction to be performed and running the update procedure of the data space as background operation.

However note that a cache, included in the plugin infrastructure is definitively not the best solution because all the information of the object cache is lost when the plugin gets disabled or uninstalled, although its data is still used in the system in form of attribute values. So after loading the plugin again it would either be necessary to parse this information from a locally stored data record or to iterate over all SMOs having the corresponding media type attached because otherwise all MediaInstanceUpdateEvents would again result in contacting the remote service. In the process of discussing this problem we agreed an that handling hash-values is something the METIS core should care about.

8.2.2.5 Error Handling and Logging Errors that occur in the process of retrieving data resources, extracting the MPEG-7 features (including remote exceptions of the web service) or updating the SMOs, are handled with a rollback of the affected transaction.

Users are informed of exceptions either by events of the PluginException class (e.g. displaying graphical dialogs) or are advised to have a look at the logs, as all "debugging", "error" and "warning" messages are redirected to the log4j category "at.researchstudio.dme.metis.plugins".

Finally the unregister method of the XMFeatureExtractionPlugin class follows the recommendations for creating a clean plugin implementation, as it unregisters itself from the manager and its subscribed events.

8.2.3 Problems and Suggestions

METIS library problem We experienced massive problems and spent lots of debugging time on the task of integrating the XM feature extraction web service functionality into the METIS Plugin. Although each component, i.e. the plugin containing a client implementation to the offered mpeg7Extraction web service by making use of the AXIS libraries for stub generation and SOAP message handling, as well as METIS works fine when it is viewed independently. After loading the plugin in the context of METIS however it showed up unexpected program termination and Nullpointer references, but did not throw any exceptions. We were able to trace the failure back and noticed that it occurs when the client implementation delivers own versions of the required AXIS libraries in its plugin/lib folder, when METIS already contains an older version of these libraries in its repository (\metis\WEB-INF\lib). The tricky point is that the plugin compiles without problems as METIS PDE is configured to use the plugin's supplied libraries to perform this step - at runtime the system falls back to use the versions contained in its repository because they are added into the classpath first. As a result the software is forced to use a different version of AXIS as it was compiled with and therefore communication fails and the software terminates unexpectedly.

This topic however points out an important limitation for plugin development in METIS as there is no way to determine which libraries are already

contained in the system's repository or whether other components are dependent on them. However this can further be characterized as a general JAVA problem too because the platform does not offer any support for ranking libraries, distinguishing them by version or signing JARs.

As a reaction to this limitation the parts of the server and client side code which build on AXIS were downgraded in their functionality to support axis-1.2-alpha, saaj-1.1 and wdsl4j in order to properly work with the delivered libs. However note that all of them exist in newer versions and for example axis-1.3 (release October 2005) adds more stable support for Soap with Attachments.

8.2.3.1 Summary The implemented XMFE, a heavy-weight visual plugin for METIS, has the goal to integrate support for MPEG-7 into the system and to offer feature extraction for attached MediaInstances. Therefore the plugin contains extended client functionality for the mpeg7Extraction web service (e.g. as methods for building complex data structure, or by following a one-stop-shop implementation). The decision to use AXIS on the client side for generating its stubs and taking care of all SOAP message handling, is made autonomous from the server side. The XMFE gets notified by the system when update events occur and is responsible for launching the remote MPEG-7 feature extraction process. Due to the fact that all images must be transferred to the service's repository as well as the fact that applying the descriptors to the image data is a costly process, a client side cache is installed. The plugin depends on the XM Basic semantic pack as it offers the underlying data structures the description results are written to. As the XMFE runs autonomously from user interaction, a minimal graphical user interface, extending the METIS web frontend, is sufficient for configuring the plugin.

8.3 XM Basic Semantic Pack - Presenting the Data Model

8.3.1 Introduction and Requirements Description

Semantic packs in METIS have two main purposes: firstly they can be viewed as sub-models for uniquely distinguishing different elements (similar to namespaces in XML) and secondly, at the same time provide a kind of contract, for guaranteeing the availability of a well defined set of data types and attributes in the model after the semantic pack was installed. Therefore plugins can be dependent on semantic packs i.e. on its provided media types and attributes - but not vice versa.

The XMBasic semantic pack is mandatory for the implemented plugin and therefore provides a data set, satisfactorily covering the basic MPEG-7 visual elements. As the XM produces valid XML documents containing the results of a feature extraction process, the structure of the model elements must either be able to store the complete XML file or offering facets for capturing the element's structure and attributes as well as their actual values. Example 4 depicts a possible feature extraction result created by the XM framework, showing its data

structure that is required for wrapping a ColorLayout descriptor. Note that although the XM only makes use of a small set of simple MPEG-7 tags for creating media collections, the results are well-formed XML documents, following the MPEG-7 Schema specifications and may be validated against them.

Example 4 .

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Mpeg7 xmlns="http://www.mpeg7.org/2001/MPEG-7_Schema"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance">
    <DescriptionUnit xsi:type="DescriptorCollectionType">
        <Descriptor xsi:type="ColorLayoutType">
            <YDCCoeff>15</YDCCoeff>
            <CbDCCoeff>40</CbDCCoeff>
            <CrDCCoeff>22</CrDCCoeff>
            <YACCoeff5>16 18 15 13 9 </YACCoeff5>
            <CbACCoeff2>22 21 </CbACCoeff2>
            <CrACCoeff2>8 8 </CrACCoeff2>
        </Descriptor>
    [...]
</Mpeg7>
```

Because the implemented plugin is highly dependent on the underlying data structure offered by the XMBasic semantic pack, it must examine whether the semantic pack is already installed on the METIS distribution - if not the user is informed to do so, as otherwise the plugin cannot be loaded. This task is performed in the plugin's register method.

8.3.2 Model Elements

The decision was made not to implement proprietary data types for capturing the visual descriptors because it seems unreasonable to pack the extracted XML results into properly constructed METIS data types as this would manipulate their representation. That is an important consideration due to the fact that a number of algorithms or tools exist which are configured to run on the XML representation. Altering the representation would either require methods for recomposing the original file out of the METIS data types or to adapt the algorithms to work on the alternative structures.

Unfortunately METIS does currently not offer an XML representation, so the results are applied to the string data type which comes in the "Basic" semantic pack. However an XML type is strongly requested for METIS as this would offer specific operations for accessing parts of a document, as certain element tags or data values. It could further perform syntactical checks, when a scheme or DTD is provided, or even validate its attached data.

The created model of the *XMBasic* semantic pack , which is currently available at version 0.62, contains a single MediaType with the name "XM_Image",

located in the category "Image". The description provided in the model states: "The XM_Image represents a 'MPEG-7 Visual' representation. The offered attributes are standardized descriptions of images using the MPEG-7 XM visual-descriptors for color, texture, shape,...They give a characteristic fingerprint which can be used in MPEG-7 schemes for image retrieval and search".

For every implemented descriptor supported by the plugin, a MDAttributes exists which is named by assembling "XM" followed by its qualifying name and an added "XML" at the end (e.g. XMDominantColorXML). The attributes provides a short description summarizing their ISO/IEC 15938-3:2002 characterization, which gets displayed in the web frontend. Beyond that they also supply a default description, outlining the structure of the descriptions. All attributes in an XM_Image type may occur in the range from 0 to 1.

A Installation Guidelines

A.1 Server Side Components

Installationguide_v1.0 23.01.2006

Installationguidline

XM Feature Extraction Web Service v0.94 **Server Side Config**

1. Abstract:

This installation guideline describes the setup procedure of a server machine, hosting the eXpermimantal Model (XM) and deploying the AXIS web service (mpeg7Extraction). All of the needed files for this step should be provided in the working dir “Server Side Files” on the CD. It is important to notice that if you plan to edit, modify or extend one of the components (doesn’t matter if client or server side) you will have to follow the “extended setup procedure” described in the written Diplomarbeit.

2. Requirements:

The installation of the server side components, the Feature Extraction Web Service (mpeg7Extraction) and the eXperimental Model (XM) requires the following software to be installed and properly configured on the server machine.

Apache Tomcat – tested with version jakarta-tomcat-5.0.28
Apache Axis – tested with version 1_2_alpha (see point 4 for troubleshooting)
javaRTE (v.1.3.1 or higher)

Operating system *Windows XP*
The added and pre-compiled version of the eXperimental Model (XM) depends on the Microsoft Windows XP operating system because some of its components use system libraries as well as the mpeg7Extraction web service that calls XP specific shell commands.

3. Software Setup

The setup of the software should be followed in the order described in this section.

3.1 Installation of the eXperimental Model + web service working environment

The preconfigured and compiled files of the eXperimental Model + all of the needed tools (e.g. like ImageMagick) for handling images, etc. are contained in the “XM_Files.zip”.

- extract the files contained in „XM _Files.zip“ (recommendation:) under the root directory "C:\“. It should then exist a file structure similar to „C:\temp\pluginMETIS“ and „C:\temp\mpeg7“.

- create a Windows system variable with the name "MP7JRS_HOME" and add the value “C:\temp\Mp7Jrs\\" to it.

- Add the following string to the Windows system variable „path“: "c:\temp\mpeg7\xml\lib;c:\temp\mpeg7\ImageMagick\bin"

Hint: Don’t forget to modify the directory paths when using another location than “C:\” for extraction of the XM_Files.zip.

3.2 Apache Axis

Tomcat must not be running when executing the next steps.

author: Andrew Lindley
e-mail: andrew.lindley@qse.ifs.tuwien.ac.at

- To add support for binary data to Apache AXIS (using SOAP with attachments – SwA) the folder „%Tomcat_home%/webapps/ axis/WEB-INF/lib“ must contain the java libraries activation.jar and javamail.jar.

You'll find a copy of all needed AXIS libs in the file „axis_libs.zip“. If you plan to use the mpeg7Extraction web service with METIS you have to ensure that the libs exist in the same version that METIS uses.

Hint: To see if AXIS has access to all of the mandatory jar libraries and if they are configured properly you can call the point "validation" under http://localhost:8080/axis

- extract the „ XM Feature Extraction Web Service_v0.94.zip“, which contains the actual mpeg7Extraction Web Service .jar to „%Tomcat_home%/webapps/axis/WEB-INF/lib“.

- Now startup Tomcat

- As next step we need to deploy the web service functionality. AXIS offers a command line tool which executes the (un)deploy tasks. Therefore extract the files „deploy_webservice.bat“, undeploy_webservice.bat“, „deploy.wsdd“ und „undeploy.wsdd“ from the "deploy_mpeg7Extraction_v0.94.zip" to a arbitrary directory and then run "server_deploy_webservice.bat". This performs the following commands:

```
% java org.apache.axis.client.AdminClient deploy.wsdd
<admin>Done processing</admin>
```

Hint: You'll find in the class in the axis-1.2-alpha.jar archive (which you can extract from "axis_libs.zip"). Make sure the .bat file has access to the .jar – the best solution is to add the jar to the System Classpath.

The mpeg7Extraction web service now should be found at the endpoint http://localhost:8080/axis/services

- Finally we only have to configure the mpeg7Extraction service to read certain properties from the system like the XM home directory. Therefore extract the setup utility from „configure_mpeg7Extraction_v0.94.zip" and start the java program by calling „java setupXM.SetupServerProperties“ and add the pluginMETIS installation directory (e.g. "C:\temp\pluginMETIS" that you specified under point 3.1) as the only argument. This will create a properties file which is absolute mandatory for the functionality of the web service.

4. additional hints and trouble shooting

How to test if the XM Framework is working

A sample configuration (for testing purpose) should exist in the directory "C:\temp\pluginMETIS". The files used by this example carry the ClientID 84180223. You can now just start the Feature Extraction for the images: "Knowme.jpg" and "D1001_jpeg_small.jpg" by calling "extract 84180223.bat". The extraction results should then be found in the "desc_extracted" directory – a log of the extraction process can be found in the "extract_84180223.txt file.

author: Andrew Lindley
e-mail: andrew.lindley@qse.ifs.tuwien.ac.at

Note that the first time you use the mpeg7Extraction web service the file system will get cleared by the garbage collector and the example will get deleted.

Problems running METIS and the mpeg7Extraction web service on the same webserver

METIS and the XM web service have problems running on the same webserver container e.g. Tomcat. The problem occurs when sending binary data from one hosted tomcat application to another hosted application - like METIS to the XM web service. This problem only involves binary data transferred from METIS to XM web service and not external programs communicating with AXIS.

The XMFE METIS Plugin terminates when uploading files to the web service

Check if the installed Apache AXIS uses the same versions of following libraries as METIS does: axis-1.2-alpha, saaj-1.1, wsdl4j-1.4.

The problem is that the web service client which is used in the METIS XMFE-plugin also depends on the AXIS libraries that are used in METIS (\metis\WEB-INF\lib). When delivering newer versions of the libraries in the plugin/lib folder of the METIS XMFE Plugin, the metis-pde compiles the software but METIS still falls back to the elder own versions at runtime and so communication fails.

XM Framework is missing the Msvcr71.dll system library.

It is possible that the Msvcr71.dll – a Microsoft Windows system library – is missing on a brand new Windows XP installation. This library is used by ImageMagick – MPEG-7 eXperimental Model uses ImageMagick to perform loading and decoding of still and animated images. This missing file can be downloaded here or just install Adobe Acrobat Reader which also provides this dll.

Additional information concerning the usage of the eXperimental Model

- feature extraction is performed in "C:\temp\pluginMETIS"
- "extract+ClientID.bat" extracts MPEG-7 image features from the images named in " list_of_pictures+ClientID.lst" (paths relative to pluginMETIS) using the descriptor parameters in “desc_settings/"<descriptorname>+ClientID.par" (e.g. ColorLayout.par).
- extracted descriptions are currently stored in " desc_extracted".
- sample descriptions can be performed when calling “extract84180223.bat”

some hints:

- store images as jpegs, not as gifs although gif is also supported
- if you want to extract descriptions from video clips (video input in the XM is a nuisance...): use a tool (e.g. matlab) to extract frames as jpegs and store them in a separate directory with framenumber as name
- extraction of Texture Browsing is generally very slow, and instable for large images.
- Homogeneous Texture uses only the upper left 128xs128 pixels of an image for extraction
- descriptor source files are stored in "C:\temp\mpeg7\newsrc\Descriptors".
-"XMMain" is used for extraction. "s" is just a tool to redirect STDOUT.
- all XM documentation can be found in "C:\temp\mpeg7\newsrc\Doc"

author: Andrew Lindley
e-mail: andrew.lindley@qse.ifs.tuwien.ac.at

A.2 Client Side Components

Installationguidline

XMFE METIS-Plugin v0.1 **Client Side Config**

1. Abstract:

This installation guideline describes the setup procedure of the XM Feature Extraction (XMFE) Plugin for METIS. We recommend to follow the "extended setup procedure" for the client – only use the "quick setup" if you know that the server is running the mpeg7Extraction web service v0.94. All of the needed files for this step should be provided in the working dir "Client Side Files" on the CD. It is important to notice that if you plan to edit, modify or extend one of the components (doesn't matter if client or server side) you will have to follow the "extended setup procedure" anyway.

2. Requirements:

The installation of the client side components, the XMFE-Plugin for METIS and the integrated web service client requires the following software to be installed and properly configured on the client machine.

METIS - having BasicSempack installed
metis-pde
javaRTE (v.1.3.1 or higher)

3. Software Setup

The setup of the software should be followed in the order described in this section. For all of these steps we assume that the server side functionality has already been setup (if not – do it now) and the service is running.

3.1 XMBasic Sempack for METIS

- Upload the XMBasic-Sempack to METIS ("sempack-XMBasic-01092006.jar") which covers the data model for the supported description results of the mpeg7Extraction web service v0.94.

3.2 XMFE Plugin for METIS

The plugin requires the user to know the location of the XM web service location. If you're not sure which version the server is running or the version != 0.94 then follow the "extended setup procedure".

3.2.1 quick setup procedure

Hint: We recommend to test the XM web service with the described testing class before using the XMFE-Plugin.

-extract the "plugin-XMFE_v01.zip" into your metis-pde src folder.
-go to the "plugin-XMFE/src" folder and open the "XMLLauncherServiceLocator.java" file edit the value of the public static java.lang.String mpeg7Extraction_address =" http://192.168.0.6:8080/axis /services/mpeg7Extraction" to the Endpoint where the mpeg7Extraction service instance is running.
- use metis-pde to build the plugin-XMFE.jar. e.g. call "maven build:plugin –Dname=XMFE"
- Now upload the plugin jar to METIS and configure it.

author: Andrew Lindley
e-mail: andrew.lindley@qse.ifs.tuwien.ac.at

3.2.2 extended setup procedure

Hint: We recommend to test the XM web service with the described testing class before using the XMFE-Plugin.

- extract the "axis_libs.zip" to a folder of your choice and add the jar files to the system Classpath.
- extract the files contained in the "extended_setup_files_v0.94.zip" to a directory of your choice. (referred to as "setup_home_dir")

- go to the endpoint and ask AXIS to return you the WSDL description file for the mpeg7Extraction web service containing its callable methods and parameters. AXIS returns the WSDL description by calling the URL of the Endpoint+"?WSDL". e.g. http://localhost:8080/axis/services/mpeg7Extraction?WSDL
  Save this file as "XMServer.wsdl" to your setup_home_dir.
- Now call the "client_wsdl2java.cmd" – which uses the WSDL file as input to automatically generate the client stubs. As a result there should now exist a mpeg7\client\cl directory

- extract the "plugin-XMFE_v01.zip" into your metis-pde src folder.
- delete all files in the "plugin-XMFE/src" directory except of "XMWrapper.java" and "XMFeatureExtractionPlugin.java"
- copy all of the files from your "setup_home_dir/mpeg7/client" and "setup_home_dir /mpeg7/client/cl" without the folder structure to your "plugin-XMFE/src" directory.
-go to the "plugin-XMFE/src" folder and open the "XMLLauncherServiceLocator.java" file. Edit the variable "mpeg7Extraction_address" to "public static java.lang.String" and change its value to the Endpoint where the mpeg7Extraction service instance is running. e.g. public static java.lang.String mpeg7Extraction_address =" http://192.168.0.6:8080/axis /services/mpeg7Extraction"
- use metis-pde to build the plugin-XMFE.jar. e.g. call "maven build:plugin –Dname=XMFE"
- Now upload the plugin jar to METIS and configure it.

4. additional hints and trouble shooting

How to test if the XM web service and the client are working.

Extract the Test1.java and the mpeg7 client files from the "web_service_test.zip" to a folder of your choice and edit the XMLauncherServiceLocator.java as described above. This Test file contains a very intuitive usage of the XMLauncherClient that – by the default settings – tries to upload a file "C:/grafiktest/Knowme.jpg" and a URL to the webservice, sets the standard descriptors, calls the extraction of the features and returns true if it was carried out successfully. Of course you may alter the location of the image.

Note that the java compiler must find the axis_libs on your Classpath to execute the program.

The XMFE METIS Plugin terminates when uploading files to the web service

Check if the METIS instance and the XMFE Plugin (plubin-XMFE/lib directory) are using the same versions of following libraries as the XM web service does: axis-1.2-alpha, saaj-1.1, wsdl4j-1.4.

The problem is that the web service client which is used in the METIS XMFE-plugin also depends on the AXIS libraries that are used in METIS (\metis\WEB-INF\lib). When

author: Andrew Lindley
e-mail: andrew.lindley@qse.ifs.tuwien.ac.at

delivering newer versions of the libraries in the plugin/lib folder of the METIS XMFE Plugin, the metis-pde compiles the software but METIS still falls back to the elder own versions at runtime and so communication fails.

When following the extended setup procedure the XMLauncherClient doesn't compile

When you experience errors in the XMLauncherClient.java or the mpeg7.client.cl Stubs when compiling the source code it probably has to with a changed functionality at the server side. Compare the XMLauncherClient methods with the WSDL description of the mpeg7Extraction supported services.

author: Andrew Lindley
e-mail: andrew.lindley@qse.ifs.tuwien.ac.at

B Additional Descriptor Settings

Horst Eidenberger, PhD, Associate Professor of the Vienna University of Technology
For Andrew Lindley

MPEG-7 – Parameter

Color Structure

Name	Bedeutung	Beispiel
ColorQuantSize	Color quantization resolution (32, 64, 128, 256)	32

Dominant Color

Name	Bedeutung	Beispiel
ColorSpace	Color space (0...RGB, 1...YCrCb, 2...HSV, 3...HMMD, 4...LinearMatrix, 5...Monochrome)	0
Component0	first color component (0...R, 3...Y, 6...H)	0
Component1	second color component (1...G, 4...CB, 7...S, 12...Sum)	1
Component2	third color component (2...B, 5...Cr, 8...V, 11...Diff)	2
BinNumber0	quantization for bin 1 (number of values)	256
BinNumber1	quantization for bin 2	256
BinNumber2	quantization for bin 3	256
ColorSpacePresent	0...Component* wird im Parfile gesetzt, 1...defaults benutzen (abgeleitet von ColorSpace)	0
ColorQuantizationPresent	1...BinNumber*-Werte aus Parfile benutzen 1...Defaultwert benutzen (32=5bit)	0
VariancePresent	1...varianzen berechnen, 0...defaultwerte benutzen (0.0)	0
SpatialCoherency	1...Spatial Coherency berechnen, 0...defaultwert benutzen (0)	0

Homogeneous Texture

Name	Bedeutung	Beispiel
layer	anzahl der zu berechnenden komponenten (0...base-layer 32-components, 1...full-layer 62-components)	1
option	default matching option for client (n...normal matching, r...rotation invariant matching, s... scale invariant matching, rs...rotation and scale invariant matching)	n

Scalable Color

Name	Bedeutung	Beispiel
NumberOfBitplanesDiscarded	quantisierung für die koeffizienten der haar-transformation. wert=anzahl der wegzuschneidenden bits von einem integer (0, 1, 2, 3, 4, 6, 8)	3
NumberOfCoefficients	anzahl der histogramm-einträge (16, 32, 64, 128, 256)	64

Texture Browsing

Name	Bedeutung	Beispiel
layer	anzahl der zu berechnenden komponenten (0...basic-layer 3-components, 1... full-layer 5-components)	0

extraction from: mpeg7/newsrc/wiTests/parameter_description.doc of the XM precompiled package

C WDSL Interface Description of the XM Web Service

Mit dieser XML-Datei sind anscheinend keine Style-Informationen verknüpft. Nachfolgend wird die Baum-Ansicht des Dokuments angezeigt.

```
− <wsdl:definitions targetNamespace="urn:mpeg7Extraction">
  − <!--
     WSDL created by Apache Axis version: 1.2alpha
     Built on Dec 01, 2003 (04:33:24 EST)
    -->
  − <wsdl:types>
    − <schema targetNamespace="http://xml.apache.org/xml-soap">
        <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
      − <complexType name="mapItem">
        − <sequence>
            <element name="key" nillable="true" type="xsd:string"/>
            <element name="value" nillable="true" type="xsd:string"/>
          </sequence>
        </complexType>
      − <complexType name="Map">
        − <sequence>
            <element maxOccurs="unbounded" minOccurs="0" name="item"
            type="apachesoap:mapItem"/>
          </sequence>
        </complexType>
      </schema>
    − <schema targetNamespace="urn:mpeg7Extraction">
        <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
      − <complexType name="ArrayOf_xsd_string">
        − <complexContent>
          − <restriction base="soapenc:Array">
              <attribute ref="soapenc:arrayType" wsdl:arrayType="tns2:string[]"/>
            </restriction>
          </complexContent>
        </complexType>
      − <complexType name="ArrayOf_apachesoap_DataHandler">
        − <complexContent>
          − <restriction base="soapenc:Array">
              <attribute ref="soapenc:arrayType" wsdl:arrayType="apachesoap:DataHandler[]"/>
            </restriction>
          </complexContent>
        </complexType>
      </schema>
  </wsdl:types>
  − <wsdl:message name="getExtractedDescriptorNamesResponse">
      <wsdl:part name="getExtractedDescriptorNamesReturn" type="impl:ArrayOf_xsd_string"/>
    </wsdl:message>
  − <wsdl:message name="extractRequest">
      <wsdl:part name="in0" type="xsd:string"/>
    </wsdl:message>
  − <wsdl:message name="getExtractionResultRequest">
      <wsdl:part name="in0" type="xsd:string"/>
      <wsdl:part name="in1" type="xsd:string"/>
    </wsdl:message>
```

```
- <wsdl:message name="getAllowedFileExtensionsResponse">
    <wsdl:part name="getAllowedFileExtensionsReturn" type="impl:ArrayOf_xsd_string"/>
  </wsdl:message>
- <wsdl:message name="receiveAndStoreFileRequest">
    <wsdl:part name="in0" type="apachesoap:DataHandler"/>
    <wsdl:part name="in1" type="xsd:string"/>
    <wsdl:part name="in2" type="xsd:string"/>
    <wsdl:part name="in3" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="getAddedDescriptorListResponse">
    <wsdl:part name="getAddedDescriptorListReturn" type="impl:ArrayOf_xsd_string"/>
  </wsdl:message>
- <wsdl:message name="receiveAndStoreDirRequest">
    <wsdl:part name="in0" type="impl:ArrayOf_apachesoap_DataHandler"/>
    <wsdl:part name="in1" type="impl:ArrayOf_xsd_string"/>
    <wsdl:part name="in2" type="xsd:string"/>
    <wsdl:part name="in3" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="removeDefaultDescriptorsRequest">
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="getExtractionResultsResponse">
    <wsdl:part name="getExtractionResultsReturn" type="apachesoap:Map"/>
  </wsdl:message>
- <wsdl:message name="getExtractionResultResponse">
    <wsdl:part name="getExtractionResultReturn" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="removeFileRequest">
    <wsdl:part name="in0" type="xsd:string"/>
    <wsdl:part name="in1" type="xsd:string"/>
    <wsdl:part name="in2" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="removeDominantColorResponse">
    <wsdl:part name="removeDominantColorReturn" type="xsd:boolean"/>
  </wsdl:message>
- <wsdl:message name="addEdgeHistogramResponse">
    <wsdl:part name="addEdgeHistogramReturn" type="xsd:boolean"/>
  </wsdl:message>
- <wsdl:message name="removeTextureBrowsingResponse">
    <wsdl:part name="removeTextureBrowsingReturn" type="xsd:boolean"/>
  </wsdl:message>
- <wsdl:message name="receiveAndStoreDirResponse">
    <wsdl:part name="receiveAndStoreDirReturn" type="xsd:boolean"/>
  </wsdl:message>
- <wsdl:message name="getExtractedDescriptorNamesRequest">
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="removeFilesRequest">
    <wsdl:part name="in0" type="impl:ArrayOf_xsd_string"/>
    <wsdl:part name="in1" type="impl:ArrayOf_xsd_string"/>
    <wsdl:part name="in2" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="addDominantColorRequest">
    <wsdl:part name="in0" type="xsd:string"/>
```



```
  </wsdl:message>
- <wsdl:message name="receiveAndStoreHTTPFileResponse">
    <wsdl:part name="receiveAndStoreHTTPFileReturn" type="xsd:boolean"/>
  </wsdl:message>
- <wsdl:message name="removeFileResponse">
    <wsdl:part name="removeFileReturn" type="xsd:boolean"/>
  </wsdl:message>
- <wsdl:message name="addHomogeneousTextureRequest">
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="addHomogeneousTextureResponse">
    <wsdl:part name="addHomogeneousTextureReturn" type="xsd:boolean"/>
  </wsdl:message>
- <wsdl:message name="removeColorLayoutResponse">
    <wsdl:part name="removeColorLayoutReturn" type="xsd:boolean"/>
  </wsdl:message>
- <wsdl:message name="receiveAndStoreHTTPFileRequest">
    <wsdl:part name="in0" type="xsd:string"/>
    <wsdl:part name="in1" type="xsd:string"/>
    <wsdl:part name="in2" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="removeFilesResponse">
    <wsdl:part name="removeFilesReturn" type="xsd:boolean"/>
  </wsdl:message>
- <wsdl:message name="removeRegionShapeRequest">
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="generateClientIDRequest"> </wsdl:message>
- <wsdl:message name="addColorLayoutRequest">
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="removeAllFilesResponse"> </wsdl:message>
- <wsdl:message name="removeDirResponse">
    <wsdl:part name="removeDirReturn" type="xsd:boolean"/>
  </wsdl:message>
- <wsdl:message name="getClientsFileListResponse">
    <wsdl:part name="getClientsFileListReturn" type="impl:ArrayOf_xsd_string"/>
  </wsdl:message>
- <wsdl:message name="addColorStructureRequest">
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="removeTextureBrowsingRequest">
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="addDefaultDescriptorsRequest">
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="removeHomogeneousTextureResponse">
    <wsdl:part name="removeHomogeneousTextureReturn" type="xsd:boolean"/>
  </wsdl:message>
- <wsdl:message name="isClientKnownResponse">
    <wsdl:part name="isClientKnownReturn" type="xsd:boolean"/>
  </wsdl:message>
  <wsdl:message name="removeClientResponse"> </wsdl:message>
```



```
- <wsdl:message name="addDominantColorResponse">
    <wsdl:part name="addDominantColorReturn" type="xsd:boolean"/>
  </wsdl:message>
- <wsdl:message name="removeScalableColorResponse">
    <wsdl:part name="removeScalableColorReturn" type="xsd:boolean"/>
  </wsdl:message>
- <wsdl:message name="getAddedDescriptorListRequest">
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="addScalableColorRequest">
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="addScalableColorResponse">
    <wsdl:part name="addScalableColorReturn" type="xsd:boolean"/>
  </wsdl:message>
- <wsdl:message name="getExtractedFileInputListResponse">
    <wsdl:part name="getExtractedFileInputListReturn" type="impl:ArrayOf_xsd_string"/>
  </wsdl:message>
- <wsdl:message name="addRegionShapeRequest">
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="removeColorLayoutRequest">
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="addColorStructureResponse">
    <wsdl:part name="addColorStructureReturn" type="xsd:boolean"/>
  </wsdl:message>
- <wsdl:message name="removeEdgeHistogramRequest">
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="getAllowedFileExtensionsRequest">
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="addColorLayoutResponse">
    <wsdl:part name="addColorLayoutReturn" type="xsd:boolean"/>
  </wsdl:message>
- <wsdl:message name="removeEdgeHistogramResponse">
    <wsdl:part name="removeEdgeHistogramReturn" type="xsd:boolean"/>
  </wsdl:message>
  <wsdl:message name="getAvailableDescriptorListRequest"> </wsdl:message>
- <wsdl:message name="removeDominantColorRequest">
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="removeClientRequest">
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="addRegionShapeResponse">
    <wsdl:part name="addRegionShapeReturn" type="xsd:boolean"/>
  </wsdl:message>
- <wsdl:message name="addEdgeHistogramRequest">
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="removeDefaultDescriptorsResponse"> </wsdl:message>
- <wsdl:message name="getExtractedFileInputListRequest">
```



```
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="receiveAndStoreFileResponse">
    <wsdl:part name="receiveAndStoreFileReturn" type="xsd:boolean"/>
  </wsdl:message>
- <wsdl:message name="removeScalableColorRequest">
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="addTextureBrowsingResponse">
    <wsdl:part name="addTextureBrowsingReturn" type="xsd:boolean"/>
  </wsdl:message>
- <wsdl:message name="addTextureBrowsingRequest">
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="addDefaultDescriptorsResponse"> </wsdl:message>
- <wsdl:message name="removeDirRequest">
    <wsdl:part name="in0" type="xsd:string"/>
    <wsdl:part name="in1" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="generateClientIDResponse">
    <wsdl:part name="generateClientIDReturn" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="isClientKnownRequest">
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="removeColorStructureRequest">
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="extractResponse">
    <wsdl:part name="extractReturn" type="xsd:boolean"/>
  </wsdl:message>
- <wsdl:message name="removeColorStructureResponse">
    <wsdl:part name="removeColorStructureReturn" type="xsd:boolean"/>
  </wsdl:message>
- <wsdl:message name="getClientsFileListRequest">
    <wsdl:part name="in0" type="xsd:string"/>
    <wsdl:part name="in1" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="removeAllFilesRequest">
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="getExtractionResultsRequest">
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="getAvailableDescriptorListResponse">
    <wsdl:part name="getAvailableDescriptorListReturn" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="removeHomogeneousTextureRequest">
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
- <wsdl:message name="removeRegionShapeResponse">
    <wsdl:part name="removeRegionShapeReturn" type="xsd:boolean"/>
  </wsdl:message>
- <wsdl:portType name="XMLauncher">
```

```
- <wsdl:operation name="extract" parameterOrder="in0">
    <wsdl:input message="impl:extractRequest" name="extractRequest"/>
    <wsdl:output message="impl:extractResponse" name="extractResponse"/>
  </wsdl:operation>
- <wsdl:operation name="getExtractionResult" parameterOrder="in0 in1">
    <wsdl:input message="impl:getExtractionResultRequest" name="getExtractionResultRequest"/>
    <wsdl:output message="impl:getExtractionResultResponse"
    name="getExtractionResultResponse"/>
  </wsdl:operation>
- <wsdl:operation name="getExtractionResults" parameterOrder="in0">
    <wsdl:input message="impl:getExtractionResultsRequest"
    name="getExtractionResultsRequest"/>
    <wsdl:output message="impl:getExtractionResultsResponse"
    name="getExtractionResultsResponse"/>
  </wsdl:operation>
- <wsdl:operation name="getExtractedDescriptorNames" parameterOrder="in0">
    <wsdl:input message="impl:getExtractedDescriptorNamesRequest"
    name="getExtractedDescriptorNamesRequest"/>
    <wsdl:output message="impl:getExtractedDescriptorNamesResponse"
    name="getExtractedDescriptorNamesResponse"/>
  </wsdl:operation>
- <wsdl:operation name="getExtractedFileInputList" parameterOrder="in0">
    <wsdl:input message="impl:getExtractedFileInputListRequest"
    name="getExtractedFileInputListRequest"/>
    <wsdl:output message="impl:getExtractedFileInputListResponse"
    name="getExtractedFileInputListResponse"/>
  </wsdl:operation>
- <wsdl:operation name="receiveAndStoreFile" parameterOrder="in0 in1 in2 in3">
    <wsdl:input message="impl:receiveAndStoreFileRequest"
    name="receiveAndStoreFileRequest"/>
    <wsdl:output message="impl:receiveAndStoreFileResponse"
    name="receiveAndStoreFileResponse"/>
  </wsdl:operation>
- <wsdl:operation name="receiveAndStoreDir" parameterOrder="in0 in1 in2 in3">
    <wsdl:input message="impl:receiveAndStoreDirRequest"
    name="receiveAndStoreDirRequest"/>
    <wsdl:output message="impl:receiveAndStoreDirResponse"
    name="receiveAndStoreDirResponse"/>
  </wsdl:operation>
- <wsdl:operation name="receiveAndStoreHTTPFile" parameterOrder="in0 in1 in2">
    <wsdl:input message="impl:receiveAndStoreHTTPFileRequest"
    name="receiveAndStoreHTTPFileRequest"/>
    <wsdl:output message="impl:receiveAndStoreHTTPFileResponse"
    name="receiveAndStoreHTTPFileResponse"/>
  </wsdl:operation>
- <wsdl:operation name="getAllowedFileExtensions" parameterOrder="in0">
    <wsdl:input message="impl:getAllowedFileExtensionsRequest"
    name="getAllowedFileExtensionsRequest"/>
    <wsdl:output message="impl:getAllowedFileExtensionsResponse"
    name="getAllowedFileExtensionsResponse"/>
  </wsdl:operation>
- <wsdl:operation name="getClientsFileList" parameterOrder="in0 in1">
    <wsdl:input message="impl:getClientsFileListRequest" name="getClientsFileListRequest"/>
    <wsdl:output message="impl:getClientsFileListResponse"
```



```
    name="getClientsFileListResponse"/>
  </wsdl:operation>
- <wsdl:operation name="generateClientID">
    <wsdl:input message="impl:generateClientIDRequest" name="generateClientIDRequest"/>
    <wsdl:output message="impl:generateClientIDResponse" name="generateClientIDResponse"/>
  </wsdl:operation>
- <wsdl:operation name="isClientKnown" parameterOrder="in0">
    <wsdl:input message="impl:isClientKnownRequest" name="isClientKnownRequest"/>
    <wsdl:output message="impl:isClientKnownResponse" name="isClientKnownResponse"/>
  </wsdl:operation>
- <wsdl:operation name="removeClient" parameterOrder="in0">
    <wsdl:input message="impl:removeClientRequest" name="removeClientRequest"/>
    <wsdl:output message="impl:removeClientResponse" name="removeClientResponse"/>
  </wsdl:operation>
- <wsdl:operation name="removeAllFiles" parameterOrder="in0">
    <wsdl:input message="impl:removeAllFilesRequest" name="removeAllFilesRequest"/>
    <wsdl:output message="impl:removeAllFilesResponse" name="removeAllFilesResponse"/>
  </wsdl:operation>
- <wsdl:operation name="removeFile" parameterOrder="in0 in1 in2">
    <wsdl:input message="impl:removeFileRequest" name="removeFileRequest"/>
    <wsdl:output message="impl:removeFileResponse" name="removeFileResponse"/>
  </wsdl:operation>
- <wsdl:operation name="removeFiles" parameterOrder="in0 in1 in2">
    <wsdl:input message="impl:removeFilesRequest" name="removeFilesRequest"/>
    <wsdl:output message="impl:removeFilesResponse" name="removeFilesResponse"/>
  </wsdl:operation>
- <wsdl:operation name="removeDir" parameterOrder="in0 in1">
    <wsdl:input message="impl:removeDirRequest" name="removeDirRequest"/>
    <wsdl:output message="impl:removeDirResponse" name="removeDirResponse"/>
  </wsdl:operation>
- <wsdl:operation name="getAvailableDescriptorList">
    <wsdl:input message="impl:getAvailableDescriptorListRequest"
    name="getAvailableDescriptorListRequest"/>
    <wsdl:output message="impl:getAvailableDescriptorListResponse"
    name="getAvailableDescriptorListResponse"/>
  </wsdl:operation>
- <wsdl:operation name="getAddedDescriptorList" parameterOrder="in0">
    <wsdl:input message="impl:getAddedDescriptorListRequest"
    name="getAddedDescriptorListRequest"/>
    <wsdl:output message="impl:getAddedDescriptorListResponse"
    name="getAddedDescriptorListResponse"/>
  </wsdl:operation>
- <wsdl:operation name="addDefaultDescriptors" parameterOrder="in0">
    <wsdl:input message="impl:addDefaultDescriptorsRequest"
    name="addDefaultDescriptorsRequest"/>
    <wsdl:output message="impl:addDefaultDescriptorsResponse"
    name="addDefaultDescriptorsResponse"/>
  </wsdl:operation>
- <wsdl:operation name="removeDefaultDescriptors" parameterOrder="in0">
    <wsdl:input message="impl:removeDefaultDescriptorsRequest"
    name="removeDefaultDescriptorsRequest"/>
    <wsdl:output message="impl:removeDefaultDescriptorsResponse"
    name="removeDefaultDescriptorsResponse"/>
  </wsdl:operation>
```



```
- <wsdl:operation name="addColorLayout" parameterOrder="in0">
    <wsdl:input message="impl:addColorLayoutRequest" name="addColorLayoutRequest"/>
    <wsdl:output message="impl:addColorLayoutResponse" name="addColorLayoutResponse"/>
  </wsdl:operation>
- <wsdl:operation name="addColorStructure" parameterOrder="in0">
    <wsdl:input message="impl:addColorStructureRequest" name="addColorStructureRequest"/>
    <wsdl:output message="impl:addColorStructureResponse"
    name="addColorStructureResponse"/>
  </wsdl:operation>
- <wsdl:operation name="addDominantColor" parameterOrder="in0">
    <wsdl:input message="impl:addDominantColorRequest" name="addDominantColorRequest"/>
    <wsdl:output message="impl:addDominantColorResponse"
    name="addDominantColorResponse"/>
  </wsdl:operation>
- <wsdl:operation name="addEdgeHistogram" parameterOrder="in0">
    <wsdl:input message="impl:addEdgeHistogramRequest" name="addEdgeHistogramRequest"/>
    <wsdl:output message="impl:addEdgeHistogramResponse"
    name="addEdgeHistogramResponse"/>
  </wsdl:operation>
- <wsdl:operation name="addHomogeneousTexture" parameterOrder="in0">
    <wsdl:input message="impl:addHomogeneousTextureRequest"
    name="addHomogeneousTextureRequest"/>
    <wsdl:output message="impl:addHomogeneousTextureResponse"
    name="addHomogeneousTextureResponse"/>
  </wsdl:operation>
- <wsdl:operation name="addRegionShape" parameterOrder="in0">
    <wsdl:input message="impl:addRegionShapeRequest" name="addRegionShapeRequest"/>
    <wsdl:output message="impl:addRegionShapeResponse" name="addRegionShapeResponse"/>
  </wsdl:operation>
- <wsdl:operation name="addScalableColor" parameterOrder="in0">
    <wsdl:input message="impl:addScalableColorRequest" name="addScalableColorRequest"/>
    <wsdl:output message="impl:addScalableColorResponse"
    name="addScalableColorResponse"/>
  </wsdl:operation>
- <wsdl:operation name="removeColorLayout" parameterOrder="in0">
    <wsdl:input message="impl:removeColorLayoutRequest"
    name="removeColorLayoutRequest"/>
    <wsdl:output message="impl:removeColorLayoutResponse"
    name="removeColorLayoutResponse"/>
  </wsdl:operation>
- <wsdl:operation name="removeColorStructure" parameterOrder="in0">
    <wsdl:input message="impl:removeColorStructureRequest"
    name="removeColorStructureRequest"/>
    <wsdl:output message="impl:removeColorStructureResponse"
    name="removeColorStructureResponse"/>
  </wsdl:operation>
- <wsdl:operation name="removeDominantColor" parameterOrder="in0">
    <wsdl:input message="impl:removeDominantColorRequest"
    name="removeDominantColorRequest"/>
    <wsdl:output message="impl:removeDominantColorResponse"
    name="removeDominantColorResponse"/>
  </wsdl:operation>
- <wsdl:operation name="removeEdgeHistogram" parameterOrder="in0">
    <wsdl:input message="impl:removeEdgeHistogramRequest"
```



```
    name="removeEdgeHistogramRequest"/>
    <wsdl:output message="impl:removeEdgeHistogramResponse"
    name="removeEdgeHistogramResponse"/>
  </wsdl:operation>
- <wsdl:operation name="removeHomogeneousTexture" parameterOrder="in0">
    <wsdl:input message="impl:removeHomogeneousTextureRequest"
    name="removeHomogeneousTextureRequest"/>
    <wsdl:output message="impl:removeHomogeneousTextureResponse"
    name="removeHomogeneousTextureResponse"/>
  </wsdl:operation>
- <wsdl:operation name="removeRegionShape" parameterOrder="in0">
    <wsdl:input message="impl:removeRegionShapeRequest"
    name="removeRegionShapeRequest"/>
    <wsdl:output message="impl:removeRegionShapeResponse"
    name="removeRegionShapeResponse"/>
  </wsdl:operation>
- <wsdl:operation name="removeScalableColor" parameterOrder="in0">
    <wsdl:input message="impl:removeScalableColorRequest"
    name="removeScalableColorRequest"/>
    <wsdl:output message="impl:removeScalableColorResponse"
    name="removeScalableColorResponse"/>
  </wsdl:operation>
- <wsdl:operation name="addTextureBrowsing" parameterOrder="in0">
    <wsdl:input message="impl:addTextureBrowsingRequest"
    name="addTextureBrowsingRequest"/>
    <wsdl:output message="impl:addTextureBrowsingResponse"
    name="addTextureBrowsingResponse"/>
  </wsdl:operation>
- <wsdl:operation name="removeTextureBrowsing" parameterOrder="in0">
    <wsdl:input message="impl:removeTextureBrowsingRequest"
    name="removeTextureBrowsingRequest"/>
    <wsdl:output message="impl:removeTextureBrowsingResponse"
    name="removeTextureBrowsingResponse"/>
  </wsdl:operation>
</wsdl:portType>
- <wsdl:binding name="mpeg7ExtractionSoapBinding" type="impl:XMLLauncher">
  <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
- <wsdl:operation name="extract">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="extractRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="extractResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="getExtractionResult">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="getExtractionResultRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
```

```
- <wsdl:output name="getExtractionResultResponse">
    <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    namespace="urn:mpeg7Extraction" use="encoded"/>
  </wsdl:output>
</wsdl:operation>
- <wsdl:operation name="getExtractionResults">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="getExtractionResultsRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="getExtractionResultsResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="getExtractedDescriptorNames">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="getExtractedDescriptorNamesRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="getExtractedDescriptorNamesResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="getExtractedFileInputList">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="getExtractedFileInputListRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="getExtractedFileInputListResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="receiveAndStoreFile">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="receiveAndStoreFileRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="receiveAndStoreFileResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="receiveAndStoreDir">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="receiveAndStoreDirRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
```



```
    </wsdl:input>
  - <wsdl:output name="receiveAndStoreDirResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="receiveAndStoreHTTPFile">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="receiveAndStoreHTTPFileRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="receiveAndStoreHTTPFileResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="getAllowedFileExtensions">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="getAllowedFileExtensionsRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="getAllowedFileExtensionsResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="getClientsFileList">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="getClientsFileListRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="getClientsFileListResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="generateClientID">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="generateClientIDRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="generateClientIDResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="isClientKnown">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="isClientKnownRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
```



```
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="isClientKnownResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="removeClient">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="removeClientRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="removeClientResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="removeAllFiles">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="removeAllFilesRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="removeAllFilesResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="removeFile">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="removeFileRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="removeFileResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="removeFiles">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="removeFilesRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="removeFilesResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="removeDir">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="removeDirRequest">
```



```
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="removeDirResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="getAvailableDescriptorList">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="getAvailableDescriptorListRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="getAvailableDescriptorListResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="getAddedDescriptorList">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="getAddedDescriptorListRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="getAddedDescriptorListResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="addDefaultDescriptors">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="addDefaultDescriptorsRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="addDefaultDescriptorsResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="removeDefaultDescriptors">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="removeDefaultDescriptorsRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="removeDefaultDescriptorsResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="addColorLayout">
    <wsdlsoap:operation soapAction=""/>
```

```
- <wsdl:input name="addColorLayoutRequest">
    <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    namespace="urn:mpeg7Extraction" use="encoded"/>
  </wsdl:input>
- <wsdl:output name="addColorLayoutResponse">
    <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    namespace="urn:mpeg7Extraction" use="encoded"/>
  </wsdl:output>
</wsdl:operation>
- <wsdl:operation name="addColorStructure">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="addColorStructureRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="addColorStructureResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="addDominantColor">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="addDominantColorRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="addDominantColorResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="addEdgeHistogram">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="addEdgeHistogramRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="addEdgeHistogramResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="addHomogeneousTexture">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="addHomogeneousTextureRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="addHomogeneousTextureResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="addRegionShape">
```



```
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="addRegionShapeRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="addRegionShapeResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="addScalableColor">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="addScalableColorRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="addScalableColorResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="removeColorLayout">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="removeColorLayoutRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="removeColorLayoutResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="removeColorStructure">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="removeColorStructureRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="removeColorStructureResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="removeDominantColor">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="removeDominantColorRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="removeDominantColorResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
```



```
- <wsdl:operation name="removeEdgeHistogram">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="removeEdgeHistogramRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="removeEdgeHistogramResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="removeHomogeneousTexture">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="removeHomogeneousTextureRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="removeHomogeneousTextureResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="removeRegionShape">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="removeRegionShapeRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="removeRegionShapeResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="removeScalableColor">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="removeScalableColorRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="removeScalableColorResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="addTextureBrowsing">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="addTextureBrowsingRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="addTextureBrowsingResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
```



```
  </wsdl:operation>
- <wsdl:operation name="removeTextureBrowsing">
    <wsdlsoap:operation soapAction=""/>
  - <wsdl:input name="removeTextureBrowsingRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:input>
  - <wsdl:output name="removeTextureBrowsingResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:mpeg7Extraction" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
- <wsdl:service name="XMLauncherService">
  - <wsdl:port binding="impl:mpeg7ExtractionSoapBinding" name="mpeg7Extraction">
      <wsdlsoap:address location="http://localhost:8080/axis/services/mpeg7Extraction"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

References

[1] R. King, N. Popitsch, and U. Westermann. METIS: A Flexible Database Foundation For Unified Media Management, in *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 744–745, New York, NY, USA, 2004. ACM Press.

[2] R. King, N. Popitsch, and U. Westermann. METIS: A Flexible Database Solution for the Management of Multimedia Assets. In *Proc. of the 10th International Workshop on Multimedia Information Systems (MIS 2004)*, 2004.

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

[4] Namespaces in XML, eds. T. Bray, D. Hollander, A. Layman**,** Jan. 1999, http://www.w3.org/TR/REC-xml-names/

[5] Resource Description Framework (RDF), eds. E. Miller, R. Swick, D. Brickley, Mar 2006, http://www.w3.org/RDF/

[6] F. Nack, A. T. Lindsay, "Everything You Wanted to Know About MPEG-7: Part1", in *IEEE Institute of Electrical and Electronics Engineers MultiMedia*, vol.6 no.3, p.65-77, July 1999

[7] Wo Chang, " ISO/IEC JTC 1/SC 29/WG 11 N6832 MPEG-A Multimedia Application Format Overview and Requirements", Palma de Mallorca, Oct. 2004, http://www.chiariglione.org/mpeg/standards/mpeg-a/mpeg-a.htm

[8] L. Chiariglione, The Moving Picture Experts Group (MPEG) Website, http://www.chiariglione.org/mpeg/standards.htm, last visited 23.05.2006

[9] L. Chiariglione, "ISO/IEC JTC1/SC29/WG11 N MPEG: achievements and current work", Nov. 2001, http://www.chiariglione.org/mpeg/mpeg_general.htm

[10] A. Thun, "Media Asset Management mit MPEG-7", Graz, Nov. 2002, http://www.iicm.edu/thesis/athun.pdf

[11] MPEG-7 Consortium, "What is MPEG-7", http://mpeg7.nist.gov/inf/inf_main.html, last visited 24.2.2006

[12] H. Kosch, "Distributed Multimedia Database Technologies Supported by MPEG-7 and MPEG-21", *CRC Press*, 2004

[13] J. M. Martínez, R. Koenen, F. Pereira, "MPEG-7 The Generic Multimedia Content Description Standard, Part 1", in *IEEE Institute of Electrical and Electronics Engineers MultiMedia*, vol.9, no. 2, p.78-87, Apr 2002

[14] J. M. Martínez, "ISO/IEC JTC1/SC29/WG11 N6828 MPEG-7 Overview (version 10)", Palma de Mallorca, Oct. 2004, http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm

[15] F. Nack, A. T. Lindsay, "Everything You Wanted to Know About MPEG-7: Part2", in *IEEE Institute of Electrical and Electronics Engineers MultiMedia*, vol.6 no.4, p.64-73, 1999

[16] ISO/IEC 15938, "Information technology - Multimedia content description interface", first edition, Jun. 2002

[17] E. van. der. Vlist, "XML Schema", *O'Reilly*, Jun. 2002

[18] J. M. Martínez, "MPEG-7 Overview of MPEG-7 Description Tools, Part 2", in *IEEE Institute of Electrical and Electronics Engineers MultiMedia*, vol.9 no.3, p.83-93, Jul. 2002

[19] N. Day, J. M. Martínez, "ISO/IEC JTC1/SC29/WG11 N4675 Introduction to MPEG-7 (version 4.0)", Jeju, March 2002

[20] H. Eidenberger, "What are the potentials of the metadata standard MPEG-7", SNML Digital Content Engineering Workshop, Salzburg, Austria, 2004

[21] IBM Research, "MARVEL: MPEG-7 Multimedia Search Engine", http://www.research.ibm.com/marvel/, last visited: 18.4.2006

[22] IBM Research, "VideoAnnEx Annotation Tool", http://www.research.ibm.com/VideoAnnEx/index.html, last visited: 20.4.2006

[23] Joanneum Research - Institute of Information Systems & Information Managemen, "MPEG-7 library: A complete API to manipulate MPEG-7 documents", http://iiss039.joanneum.at/cms/index.php?id=80, last visited 18.4.2006

[24] H. Rehatschek, "DIRECT-INFO: media monitoring and multimodal analysis for time critical decisions", in *Proceedings of 5th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS) 2004*, ISBN-972-98115-7-1, Apr. 2004 http://www.joanneum.at/cms_img/img2278.pdf

[25] H. Eidenberger and C. Breiteneder, "VizIR - A Framework for Visual Information Retrieval", in *Journal of Visual Languages and Computing*, Elsevier, Vol. 14, No. 5, pp. 443-469, 2003, http://www.ims.tuwien.ac.at/~hme/papers/jvlc2003.pdf

[26] Know-Center Graz, "Caliph&Emir: Semantic Annotation and Retrieval in Personal Digital Photo Libraries", http://www.semanticmetadata.net/features/, last visited 19.4.2006

[27] RWTH Aachen University, "MECCA: Multimedia capturing of collaborative scientific discourses about movies", http://www-i5.informatik.rwth-aachen.de/lehrstuhl/projects/MECCA/, last visited 20.4.2006

[28] M. Lux, M. Granitzer, "Retrieval of MPEG-7 based Semantic Descriptions", in *WebDB meets IR*, Proceedings of the 11. GI-Fachtagung für Datenbanksysteme in Business, Technologie und Web, Karlsruhe, Mar. 2005

[29] N. Day, "ISO/IEC JTC1/SC29/WG11 N4676 MPEG-7 Applications", Jeju, Korea, Mar. 2002

[30] H. Eidenberger, "Statistical analysis of content-based MPEG-7 descriptors for image retrieval", *ACM Multimedia Systems journal*, Springer, vol. 10, No. 2, pp. 84-97, 2004.

[31] H. Eidenberger, "How good are the visual MPEG-7 features?", in *SPIE & IEEE Visual Communications and Image Processing Conference*, Lugano, Switzerland, 2003.

[32] D. Chappell, T. Jewell, "Java Web Services", *O'Reilly*, Mar. 2002

[33] Dion Almaer, "Creating Web Services with Apache Axis", O'Reilly ONJava.com, Mai 2002, http://www.onjava.com/pub/a/onjava/2002/06/05/axis.html?page=1, last visited: 16.5.2006

[34] The Apache Software Foundation, "Axis User's Guide - version 1.2", http://ws.apache.org/axis/java/user-guide.html#WhatIsAxis, last visited 23.5.2006

[35] T. Bayer, "Die SOAP Engine Apache AXIS - oder: die Achse des Guten", in *Orientations in Objects*, Feb. 2003, http://www.oio.de/axis-soap.htm, last visited 23.5.2006

[36] B.Marchal, "Passing files to a Web service" in *IBM developerWorks*, Feb. 2004, last visited 23.05.2006, http://www-106.ibm.com/developerworks/xml/library/x-tippass.html

[37] M. Powell, "Grundlegendes zu DIME und WS-Attachments", in *MSDN*, Jan. 2003, last visited 23.5.2006, http://www.microsoft.com/germany/msdn/library/xmlwebservices/GrundlegendesZuDIMEUndWSAttachments.mspx?mfr=true

[38] Y. Ying, Y. Huang and D. Walker, "A Performance Evaluation of Using SOAP with Attachments for e-Science", School of Computer Science, Cardiff University, 2005, http://www.allhands.org.uk/2005/proceedings/papers/422.pdf