ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

David Marimón Sanjuán

Design and Implementation of a UMA Environment

Diploma project

Advisors: Olivier Steiger
Professor Touradj Ebrahimi

Lausanne, EPFL
2002

Acknowledgments

En primer lloc dono les gràcies a la meva família pel seu suport i especial comprensió i paciència, durant la carrera.

I would like to thank Olivier Steiger for his advices. Thank to his clear idea of the project I have learned to separate objectives from implementation. Having worked with him made me understand the importance of independence and teamwork at the same time.

I am also grateful to Prof. Touradj Ebrahimi for his underline of the possibility to search for other solutions during the whole project. Our progress meetings made me review and plan my work periodically so we could together visualize the whole procedure and find the best way to reach the objectives.

Thanks to the whole LTS, because their sense of humour and humanity let me have a great time. Special mention also to its leisure equipment: "babyfut", "trotinettes" and ping-pong table.

Em veig en l'obligació moral de fer referències, per ordre d'antiguitat, a tots els que heu fet possible que avui estigui aquí: al grup de Segur, tant heterogeni i particular, a la gent del Cau (sempre us portaré a dins), als amics del CMOS United (romàntics exemplars) et à tous les gens internationaux de Lausanne. Faig una menció especial a l'Arnau, responsable de que hagi fet aquest Erasmus. Per tots ells una abraçada, us estimo.

No podria fer uns agraïments sense citar l'especial col·laboració de LA COMUNA. Cornerstone del meu Erasmus i referent indiscutible de l'amistat, el suport, el riure i la festa. Gracias chicos, sois muy grandes.

I owe my experience here to all of you, thanks.

Contents

List of Figures

List of Tables

Abbreviated terms

API: Application Program Interface
DOM: Document Object Model
DIA: Digital Item Adaptation
DIID: Digital Item Identification and Description
MDS: Multimedia Description Schemes
MPEG: Moving Picture Experts Group
PDA: Personal Digital Assistant
UMA: Universal Multimedia Access
XML: Extensible Markup Language

Abstract

The objective of *Universal Multimedia Access* (UMA) is to permit any user equipped with whatever device the access to multimedia information. To handle the problems of UMA, two different approaches are commonly used: store variations of the same content and send the most appropriate one, and store the original content and adapt it on-the-fly. In this project a UMA environment using a mixture of both approaches is proposed. The tools allowing to reach this goal are: an Annotation Tool which describes media using MPEG-7, and a Client-Server Application that takes all the steps for the browsing and retrieval of media.

After an overview of the designed UMA system, the function of the MPEG-7 annotation tool is explained. In particular, a descriptor list for content annotation is proposed. These descriptors are meant for content as well as for media feature description. The client-server application is then explained. Particular insight is given into the handling of user preferences and device capabilities. Finally, the UMA environment is tested on a Personal Computer simulating diverse devices and users. These tests show that the system behaves as expected and that possible extensions and improvements can be added.

Chapter 1

Introduction

Portable computer equipment, like Personal Digital Assistants (PDA) or cellular phones, is getting a steadily increasing amount of functionality. A user can connect to the Internet, read his/her e-mail and even access audiovisual databases with such equipment. However, portable equipment still has constraining hardware limitations in terms of storage capacity or processing power.
The variety of available computing devices is the context where the need for Universal Multimedia Access (UMA) emerged. The objective of UMA is to permit any user equipped with whatever device the access to multimedia information (Figure 1.1).

The most straightforward way to achieve this is referred to as the "Info-Pyramid". It is based on the concept of storing the multimedia data in different *variations* (e.g., different resolutions) and to retrieve the one that best matches the client device.

Another solution is the "Scalable Summary". Rather than selecting a specific file within a list or *pyramid*, the content is transformed according to the needs of the client (e.g., an audio clip will be transformed into plain text if the device has no speaker).

Although the scalable summary is thought to be the most generic approach, the object of this project is to implement a demonstrator of Universal Multimedia Access using a mix of both approaches. It is proposed to create a system capable of providing multimedia data according to client device capabilities and user preferences.

This work has been performed within the context of the EC-funded project PERSEO.

Figure 1.1: Universal Multimedia Access refers to the distribution of rich multimedia content to any kind of user and client device

1.1 Objectives and starting point

The goal of this project is to conceive and build a UMA system using open standards. In particular, this means select the properties and features that describe multimedia from the already existing media description standards, develop an environment capable of annotating and treating these descriptions, and finally retrieve the appropriate media to the user. The usage of open standards for these properties and features guarantees interoperability with upcoming applications.

What would merge the "Info-pyramid" solution to the "scalable summary" one, is the usage of automatic description techniques such as MPEG-7 Camera[11]. As EPFL has already worked on this technique, the basis for this extension would be the software already implemented. Another extension could be the "Scene Cut Detection"[12] which would give solution to those files that cannot be retrieved due to bit rate of the client device.

Description of media is annotated thank to the MPEG-7 and MPEG-21 standards. MPEG-7 uses a very powerful syntax to describe both media properties and content features. Three parts of the MPEG-7 standard are considered: the Visual, the Audio and the Multimedia Description Schemes. From MPEG-21, this project starts taking into account the Digital Item

Declaration(DID), the Digital Item Identification and Description (DIID) and the Digital Item Adaptation.

As there is no previous software implemented for the solution proposed, the development starts from scratch.

1.2 Organization of the report

As stated before, this work is based upon the MPEG standards for content description and management. For the ease of the reader next Chapter overviews these standards for a better comprehension of later explanations.

The specification of the overall system is described in Chapter 3. Chapters 4 and 5 give a detailed description of the annotation tool and the client-server application, respectively. In Chapter 6 the system is tested simulating diverse device capabilities and user preferences. Chapter 7 concludes this work.

Chapter 2

Theoretical background

The cornerstone of this project are the public standards for media content management and description. This chapter presents an overview of the purposes and the structure of these standards. Due to the considerable weight of XML in these standards and in the project implementation itself, a bare description of this syntax is set out first.

2.1 XML

The *eXtensible Markup Language* XML is a project of the WWWC [1]. It is designed to improve the functionality of the Web by providing more flexible and adaptable information identification.
XML is called *extensible* because it is not a fixed format like HTML (a single, predefined markup language). Instead, XML is actually a *metalanguage*, a language for describing other languages, which lets you design your own customized markup languages for limitless different types of documents. XML can do this because it is a subset of SGML, the international standard metalanguage for text markup systems (ISO 8879).
In this project, XML is used as a markup specification language: one can design ways of describing information (text or data), usually for storage, transmission, or processing by a program. XML permits to create arborescent-type structures to describe an element. As an example, for a cooking receipt we could use the description of Figure 2.1. The elements of this receipt are grouped hierarchically: the top element `<receipt>` has diverse "children" that group conceptual descriptions (a brief description, the needed ingredients and the preparation).

Up to now, several different solutions for XML validation have been developed. The two most important are the *Document Type Definition* (DTD)

```
<?xml version="1.0"?>
<receipt>
    <name>French receipt</name>
    <origine>France</origine>
    <time>few minuts</time>

    <ingredients>
        <element>Salt</element>
        <element>Pepper</element>
        <element>etc.</element>
    </ingredients>

    <preparation>
        <etape>Prepare</etape>
        <etape>Mix</etape>
        <etape>Make cook</etape>
    </preparation>
</receipt>
```

Figure 2.1: Cooking receipt description using XML

and the *XML Schema*. The former, lets one describe the markup (elements and other constructs) available in any specific type of document. However, the design and construction of a DTD can be complex. The latter lets one describe the hierarchy of elements and their type (dates, numbers, ...) using the XML syntax itself. There are several tools that can parse and validate this syntax. Therefore, the fact that DTDs are not written in standard XML syntax limits its usage. XML Schema offers several features which are required for data processing applications, such as a sophisticated set of basic data types including dates, numbers and strings.

2.2 MPEG-7

The steadily increasing amount of available audiovisual data brought up the need for efficient data indexing, browsing and retrieval methods. With the *Multimedia Content Description Interface* MPEG-7, the Moving Picture Experts Group proposes solutions to these issues. MPEG-7 is a standardized description of multimedia information. This description is associated with the content itself, to allow searching for material that is of interest to the user.

The basis elements of MPEG-7 are:

Description Scheme (DS) a description tool that describes entities or relationships pertaining to multimedia content. DSs specify the structure and semantics of their components, which may be Description Schemes, Descriptors, or datatypes.

Descriptor (D) a description tool that describes a feature, attribute, or group of attributes of multimedia content.

Descriptor Value instantiation of a D for a group of precise data.

Datatype a basic reusable datatype employed by Description Schemes and Descriptors.

Description Tool (or tool) refers to a Description Scheme, Descriptor, or Datatype.

Description Definition Language (DDL) Language that permits the creation of new D/DS or the modification/extension of existing DS and D. The DDL forms the core of MPEG-7 and is based on the XML language.

MPEG-7 is divided into eight parts. The following parts are used within this project:

Part 2. Description Definition Language specifies the XML-based syntax to be used when building new MPEG-7 compliant D and DS.

Part 3. Visual specifies a set of low-level feature Descriptors for natural and synthetic visual content. [5]

Part 4. Audio normalizes Descriptors for audio signals including music, speech, etc.

Part 5. Multimedia Description Schemes (MDS) gives a wide variety of high-level Descriptors and DS for multimedia content. Audio and visual Descriptors can then be inserted into this MDS framework. [4]

The standard does not comprise the extraction of descriptions/features. Nor does it specify the search engine (or any other program) that can make use of the description.

2.3 MPEG-21

Today, many elements exist to build an infrastructure for the delivery and consumption of multimedia content. However, the different content handling tools have not been integrated into a consistent framework so far. The MPEG-21 *Multimedia Framework* provides normalized system tools for the management of multimedia content. The intent is that the framework covers the entire multimedia content delivery chain encompassing content creation, production, delivery and trade. Universal Multimedia Access tools are one of the important parts of MPEG-21.

The elementary architectural concept in MPEG-21 is the *Digital Item*. Digital Items are structured digital objects, including a standard representation and identification, and metadata.
Basically, a Digital Item is a combination of resources (such as videos, audio tracks, images, etc.), metadata (such as MPEG-7 descriptors), and structure (describing the relationship between resources). Figure 2.2 shows the most important elements within this model, and how they are related.

MPEG-21 is divided into seven parts, most of them still in their early days. The following parts are used within this project:

Part 2. Digital Item Declaration (DID) gives tools for the hierarchical declaration of digital items (video, pictures, audio,...).[7]

Part 3. Digital Item Identification and Description specifies how to uniquely identify digital items using MPEG certified identification systems such as ISBN, EAN, etc.[8]

Part 7. Digital Item Adaptation gives descriptions and format independent mechanisms that provide support for adaptation in terms of resource adaptation, descriptor adaptation, and/or Quality of Service management. [9]

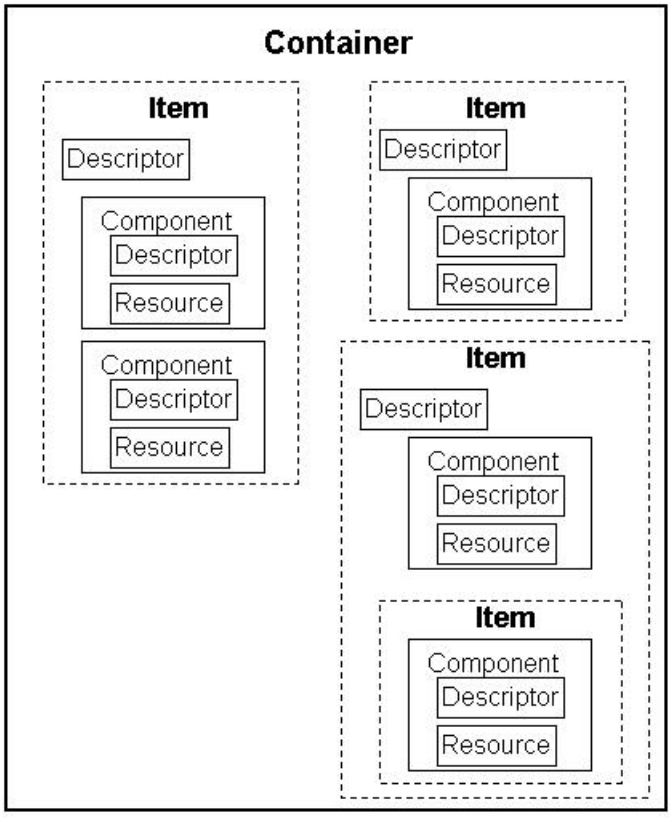

Figure 2.2: Relationship between MPEG-21 elements

Chapter 3

System Overview

The proposed UMA System is composed of diverse applications that work altogether toward a unique objective: the delivery of multimedia files depending on user preferences, client device capabilities and channel behavior. These different applications are: a content annotation tool and a client-server application. The annotation tool is developed in order to annotate content features and media properties in a standardized way throughout all the files stored in the system. The client-server application is the application that deals with all the media delivering and playback procedures. This Chapter outlines these applications, along with the objectives and usage scenarios of the system.

3.1 Objectives

Nowadays, many media delivery systems give very limited information to the user. The system proposes to use *content features* (title, summary, ...) to index media. Browsing media is only interesting if the client is able to know a little bit more than just the name of the file.
Also, these systems often give the opportunity to browse through media that is not playable by user's device, due to hardware incapabilities like insufficient resolution. Therefore, *media properties* (bandwidth, resolution, audio, ...) should also be annotated. By doing so, the system would only let the user play what he/she can. Both content features and media properties descriptions, can be handled using the MPEG-7 Description standard.

Another need of the system is to deduce users preferences from the diverse requests of the client. This can be achieved by analyzing repetitive user requests (e.g., if a client repetitively asks for weather forecasts, this type of

content should be showed first).

The system will use a simple browser for demo. Research on complex search engines is out of the scope of this project; however simple mechanisms to find specific content will be implemented.

3.2 The UMA Environment

The whole procedure systematizes content delivery. First, raw media files are annotated with a content annotation tool. This gives a MPEG-7 description of a media file. The different variations of media properties (resolution, bitrate,...) that have the same content, can be grouped in a MPEG-21 container: each variation represents a resource with the same content. With the collection of MPEG-21 files, the system has media indexed. Furthermore, the system needs users to describe their preferences. When preferences are organized, the server can start with the delivery of media. The system manages this delivery accessing both user preferences and content databases. It also performs file retrieval and updates user preferences. (Figure 3.1).

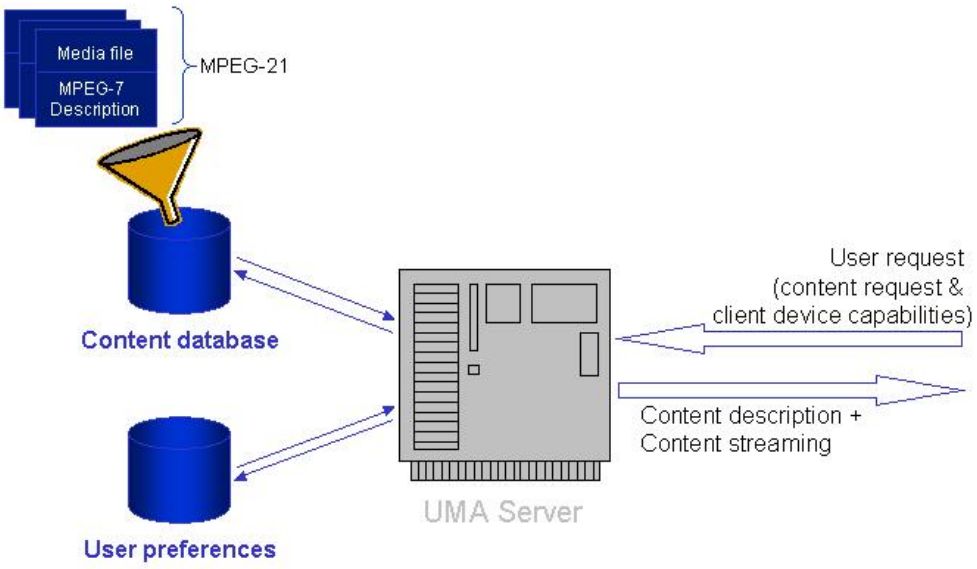

Figure 3.1: Overview of UMA Environment

3.2.1 Content annotation tool

For efficient browsing and retrieval, there is a need for annotation of multimedia content. The goal of the annotation tool is get this efficiency by annotating multimedia content using MPEG-7.

In this system, the annotation tool annotates content features and media properties using MPEG-7 wherever possible.

3.2.2 Client-server application

Since all media is stored in centralized databases and each device independently requests for media, there is an established client-server relation. As a matter of fact, there cannot be any delivery of media without a server application.

Without scratching much the surface of the server development, the intention is not only to retrieve playable content but also to deal with user preferences. The obvious objective of the interaction with the user is to make browsing and retrieval as transparent as possible, leaving him/her off the tough task of dealing with hardware characteristics.

Taking into account user preferences benefits not only the user but also the server. With the "ability" of previewing client behaviors by analyzing of user preferences, the system has a better performance for the user, as a critical part of the connection is the precious time the client looses trying to find what he/she is searching for. Also, less connection time is needed if the user is able to pick up from top of lists instead of navigating through endless submenus. To achieve this goal, *User Preferences* are structured as a list of content preferences selected by the user. In addition, the system lets the administrator have some control over preferred content. This is solved defining *Server Preferences*: list of content preferences selected by the administrator.

On the other side, the client has to be able to contact, request and follow up the steps set by the server. This is not only to browse through indexed media, but also to inform the server of its capabilities. To perform this step each device is prepared with a *Client Device Capabilities* description which lets the server discern which media can be played.

A typical content retrieval scenario looks like this:

1. **User Identification**
   The user visits a web site that initiates the procedure. The user authen-

ticates him/herself. If identification is positive, the procedure takes the next step.

2. **Send Device Capabilities**
   The capabilities of the device are sent to the server.

3. ***List of files* generation**
   At this moment the server has three sources of information: the user preferences, the client device capabilities and the server preferences. Matching them altogether the server builds the list of files the user will be able to select. Therefore, the server sorts seamlessly which content the client may desire and is capable of displaying. The files are displayed in the order the user is supposed to prefer.

4. **Content selection**
   The user chooses the media he/she prefers.

5. **Content retrieval**
   The server starts the application that will play the media in the client's side, then finds the exact file in the content database and streams it. Finally, the User Preferences Database is updated.

6. **Streaming**
   The streaming server controls necessary transfer bit rate and buffering of the file at the client side.

Chapter 4

Annotation Tool

Nowadays, a great amount of media is stored with no description attached. For a better performance when browsing media files, it is interesting to also store its description. In this case, in addition to easying the search the user can do manually, the system can look up for media that suits user preferences. Moreover, one of the proposals of UMA system is to match media files with device capabilities. The annotation tool solves exactly this problem: annotates an MPEG-7 description of media in terms of *content features* and *media properties*. Both terms are explained in this chapter. The annotation tool is a stand-alone application that plays media and stores a description of this media.

One of the context for Universal Multimedia Access is to use MPEG standards to guarantee consistence with MPEG-7 content descriptions and interoperability with upcoming applications. One of the goals of the annotation tool is to let a user use MPEG-7 without needing a deep knowledge of the standard.

4.1 Content Description

Content annotation notably facilitates the browsing and retrieval from huge multimedia databases. Two approaches are proposed for content description: the manual and the automatic approach. The manual content description is generated by a human typing the information in a form. By watching or listening, a human being can extract features that a machine cannot (e.g., geographical location, actors, ...). The other approach, the automatic content description, represents a big advantage in case a machine can do the

work (in terms of speed, cost,...) and human can keep their precious time (e.g., summarization of video surveillance). Figure 4.1 shows the procedure from raw to annotated content.

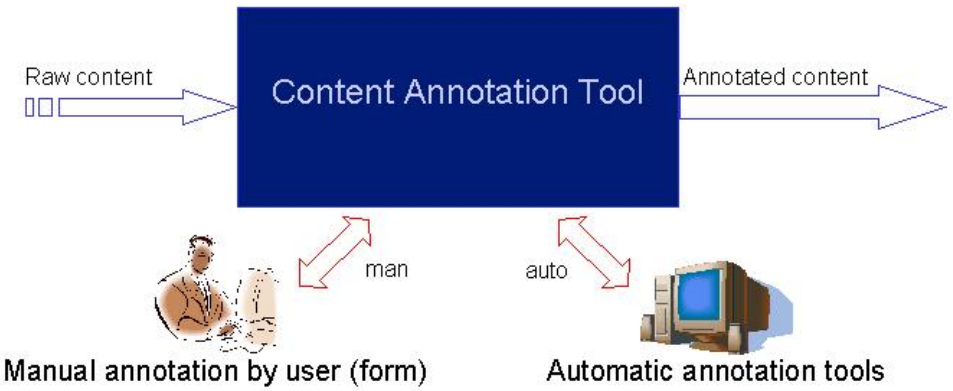

Figure 4.1: Content annotation procedure

4.1.1 Manual Content Description

Most content features cannot be extracted automatically. This concerns in particular high-level features like title, summary, geographical location, etc. On the other hand, they are often easy to find for human observers.

From the huge amount of descriptors available from the MPEG-7 standard, some have been selected because they mostly cover general user interests. Table 4.1 shows these features and their corresponding MPEG-7 Descriptor (features in **bold** are mandatory for the system to work).

Feature	Purpose	Descriptor	Document.
ID	Identify each media or segment in a unique way	EntityIdentifier	[4] §8.2.2.3
Location	Describe the origin (Web Site, Movie, . . .)	MediaUri	[4] §6.5.2.3
Start Time(*)	Specify the beginning point of a scene	MediaTimePoint	[4] §6.4.11.3
Duration(*)	Specify the duration of the whole movie clip or of a scene	MediaDuration	[4] §6.4.12
Title	Content title	Title	[4] §9.2.2.3
Summary	Describe the content briefly	Abstract	[4] §9.2.2.3
Creator	Specify the content creator (e.g., Ana Blanco, spanish news TV presenter)	Creator	[4] §9.2.2.3
Place	Describe geographical location of content	Location	[4] §9.2.2.3
Date of Creation	Describe when it was created	Date	[4] §9.2.2.3
Actors	Enumerate people or playing roles appearing (e.g., the spanish soccer player Raul)	Who	[4] §7.1
Form	Describe the production type (film, news, magazine, documentary,. . .)	Form	[4] §9.2.3.2
Genre	Describe what the multimedia content is about (sports, weather forecast, . . .)	Genre	[4] §9.2.3.2
Language	Describe media's language	Language	[4] §9.2.3.2
Copyright	Information about copyright holder	Copyright String	[4] §9.2.3.2
Extra Features	Describe special features (e.g., Parental Rating)	FreeTextAnnotation	[4]

Table 4.1: Features for Manual Content Annotation (*: Video only)

4.1.2 Automatic Content Description

Manual annotation is generic and powerful, but also lengthy and expensive. Therefore, it is useful to find features which that can be extracted automatically.
Current video segmentation systems permit to find, track and describe deforming objects in simple video sequences with static backgrounds [11]. Scene Cut Detection is also performed automatically [12].
The features proposed in Table 4.2 can be extracted in a reasonable time while providing sufficient information for scalable sequence reconstruction and the retrieval of individual video scenes [10].

Feature	Purpose	Descriptor	Docs.
Object Shape	Box or polygon shape description	Region Locator	[5] §10.2
	Describe a closed contour shape	Contour shape	[5] §8.3
Object Color	Set up to 8 dominant colors	Dominant color	[5] §6.4
	Spatial distribution of colors	Color Layout	[5] §6.6
Object Texture	Perceptual texture description	Texture browsing	[5] §7.3
	Structural texture description	Homogeneous texture	[5] §7.2
Object Motion	Perceptual motion Descriptor	Motion activity	[5] §6.6
	Single-point motion	Motion trajectory	[5] §9.3
	Describe moving regions	Parametric motion	[5] §9.4
	Specify 3-D camera motion parameters	Camera motion	[5] §9.2
Keyframes	Specify a group of AV summaries	HierarchicalSummary	[4] §13.2.4
Scene Location	Specify a video segment	VideoSegment	[4] §11.4.8

Table 4.2: Features for Automatic Content Annotation

4.2 Media Properties Description

As seen in Chapter 3, media properties must be classified and standardized within UMA, so all media files and devices can be matched by the system.

A media property is a descriptor that describes the information related to the file format or the coding parameters of the media profile. Media Properties are used for content streaming. Comparing then with client device capabilities permits to discern if media can be played and if so, decide which is the most adequate variation.

From the MPEG-7 standard, some descriptors have been selected because of their direct relation with device capabilites. Table 4.3 shows these MPEG-7 Descriptors (properties of table in **bold** are mandatory for the system to work).

Property	Purpose	Descriptor	Document.
Target bit rate	Specify the necessary overall target bitrate for streaming (bits/s)	Bitrate	[4] §8.2.4.3
Audio	Specify whether there is speech, music (HQuality, LQ), both or none	audioRating	Custom
Colored	Describes whether media is colored, graylevel or B&W	Format (colorDomain)	[4] §8.2.4.3
Color importance	Describes relevance of color: Unimportant, Useful, Important, Essential	colorImportance	Custom
Resolution	Specify image resolution	Frame (height and width)	[4] §8.2.4.3
Frame rate	Specify the number of frames per second (Hz)	Frame (rate)	[4] §8.2.4.3
File Size	Specify file size or scene size (in videos) in bytes	FileSize	[4] §8.2.4.3
Format	Specify the coding format (e.g., jpeg, mpeg, ...)	FileFormat	[4] §8.2.4.3

Table 4.3: Media Properties and corresponding MPEG-7 Descriptors

4.3 Description Example

To understand the final structure of an MPEG-7 Description, this Section presents an example (Figure 4.2). Note that descriptions used in Annotation Tool refer to all media as if it was "AudioVisualType", no matter if they are still images, only audio or only video.

```
<?xml version="1.0" encoding="utf-8"?>
<Mpeg7>
 <Description xsi:type="ContentEntityType">
   <MultimediaContent xsi:type="AudioVisualType">
     <AudioVisual>
       <MediaInformation>
           <MediaIdentification>
               <EntityIdentifier organization="MPEG"
               type="MPEG7ContentSetId">animals</EntityIdentifier>
           </MediaIdentification>
           <MediaProfile>
               <MediaFormat>
                   <FileFormat href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:">
                       <Name>avi</Name>
                   </FileFormat>
                   <FileSize>160688144</FileSize>
                   <BitRate/>
                   <VisualCoding>
                       <Format colorDomain="color"/>
```

```
                <Frame height="240" width="320" rate="26">
                </Frame>
                <colorImportance>Important</colorImportance>
            </VisualCoding>
            <AudioCoding>
                <audioRating>Speech and Music LQ</audioRating>
            </AudioCoding>
        </MediaFormat>
    </MediaProfile>
</MediaInformation>

<MediaTime>
    <MediaTimePoint>T00:00:00</MediaTimePoint>
    <MediaDuration>PT0H9M14S</MediaDuration>
</MediaTime>

<CreationInformation>
    <Creation>
        <Title>Animals</Title>

        <Abstract>
            <FreeTextAnnotation>
                The talking goldfish describes animal life.
            </FreeTextAnnotation>
            <StructuredAnnotation>
                <Who>
                    <Name></Name>
                </Who>
            </StructuredAnnotation>
        </Abstract>
        <Creator>
            <Role href="urn:mpeg:mpeg7:cs:RoleCS:2001:ANCHOR">
                <Name></Name>
            </Role>
            <Agent xsi:type="PersonType">
                <Name>
                    <GivenName></GivenName>
                    <FamilyName></FamilyName>
                </Name>
                <Icon>
                    <MediaUri/>
                </Icon>
            </Agent>
        </Creator>
        <CreationCoordinates>
            <Location>
                <Name></Name>
                <Region>es</Region>
                <AdministrativeUnit></AdministrativeUnit>
```

```
                    </Location>
                    <Date>
                        <TimePoint>1998-10-03T14:13+01:00</TimePoint>
                    </Date>
                </CreationCoordinates>
                <CopyrightString/>
            </Creation>
            <Classification>
                <Form href="urn:mpeg:mpeg7:cs:FormatCS:2001:">
                    <Name>Documentary</Name>
                </Form>
                <Genre href="urn:mpeg:mpeg7:cs:ContentCS:2001:">
                    <Name>Information</Name>
                </Genre>
                <Language>en</Language>

            </Classification>
        </CreationInformation>
        <TemporalDecomposition>
            <AudioVisualSegment></AudioVisualSegment>
        </TemporalDecomposition>

        <FreeTextAnnotation/>

      </AudioVisual>
    </MultimediaContent>
  </Description>
</Mpeg7>
```

Figure 4.2: MPEG-7 Description Example

4.4 Annotation Tool Implementation

The goal of this application is to develop an environment suitable for different types of media such as image, sound and video. In addition, the interface separes the user from the complexity of the MPEG-7 syntax. That is solved by giving the user the name of the feature instead of the MPEG-7 Descriptor wherever possible.
One of the objectives followed while working on this application was to keep it as flexible as possible, so that the descriptors selected in 4.2 and 4.1 can be easily replaced without affecting the code.

The application uses a tree that follows the XML structure of the MPEG-7 description. To implement this tree, World Wide Web Consortium's DOM

[3] was adapted using Java API. On the other half of the window space, there is a custom media player (Figure 4.3). All the treatment of media (display and extraction of some media properties) inside the application used the QuickTime Java API (ver. 5) developed by Apple.

In order to make the application as user-friendly as possible, the application uses a XML pseudo-schema to translate some MPEG-7 Descriptors to more understandable names like MediaTimePoint → Start Time. Also, this pseudo-schema sets the hierarchical relation of the elements and the datatype of each one.

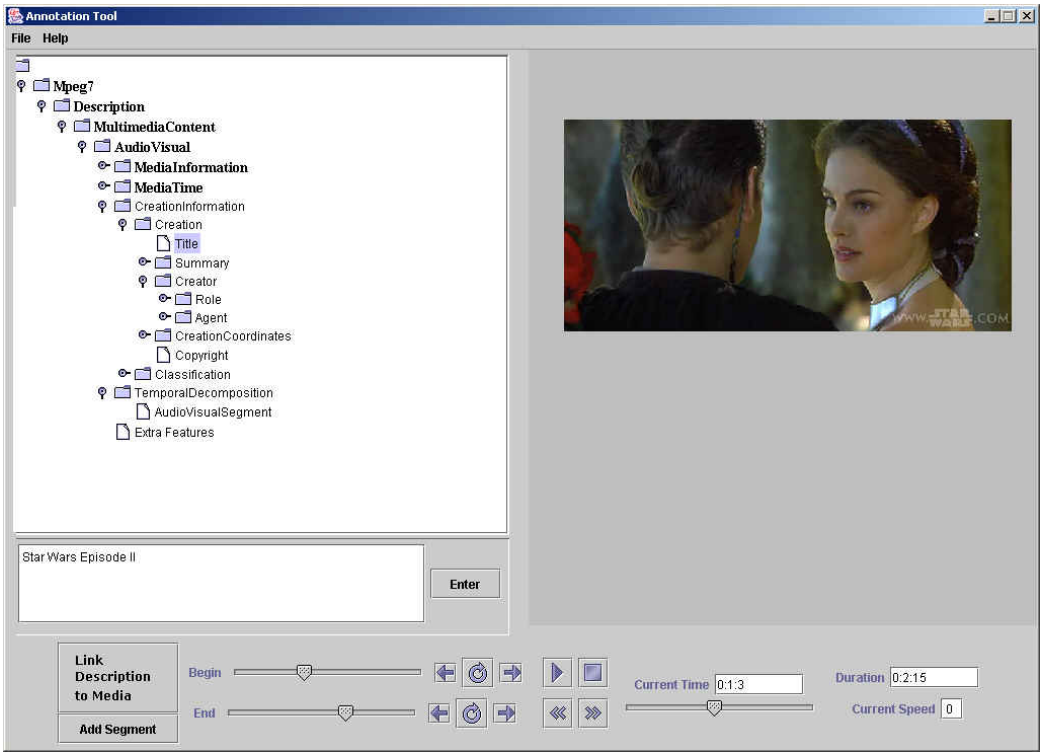

Figure 4.3: The MPEG-7 Annotation Tool

Chapter 5

Client-Server Application

This Chapter describes the implementation of the client-server application. In particular, both sides, the client and the server, are described and the algorithms used to treat Client Device Capabilities and user/server preferences, are addressed.

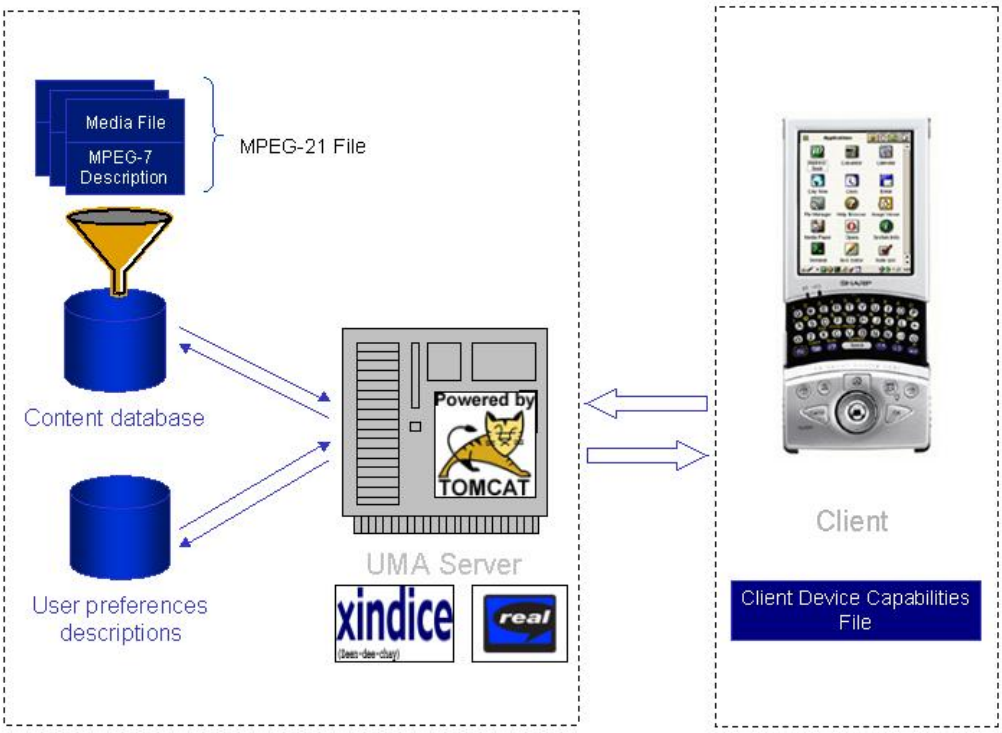

Figure 5.1: Client-Server Application

5.1 Organization of the Server

From the control of databases to how the server answers client requests, the server has to be organized so as to perform the whole retrieval procedure. This Section presents the databases used to organize media and user information, along with the software required to fulfill the whole procedure.

5.1.1 Server Databases

To store the information in the server, different databases are maintained. How to control these databases will be explained in Section 5.4.1. For the ease of implementation, information has been divided into four groups: user preferences descriptions, MPEG-7 media descriptions, MPEG-21 variation containers and media files.

User preferences descriptions A collection of XML files storing user preferences. There is one file per client.

MPEG-7 media descriptions Each media file has a description annotated in MPEG-7 as explained in Chapter 4.

MPEG-21 variation containers In each MPEG-21 description there is a reference to at least one MPEG-7 file. Each reference is a variation of media properties but not of content. The tool used to arrange media resources is MPEG-21 Digital Item Declaration [7]. Figure 5.2 is a sample of a MPEG-21 file used by the system. The `<Descriptor>` element describes the content of the *Digital Item* by referencing to a MPEG-7 file ("FamilyAnimalsHH.xml"). For this *Item* there are three variation referenced using the `<Resource>` element: the first one is colored and has Low Quality Music, the second one is also colored and has High Quality Music, and the third one is graylevel and has High Quality Music.

Media files These are the files that are retrieved by the client.

5.1.2 Server Applications

The management of client requests demands a platform of server applications. For the purposes of the UMA System a Web Server with basic functions is not enough. In fact, such simple servers do not handle databases or stream media files. The needs for this system concern three different areas:

```
<?xml version="1.0" encoding="UTF-8"?>
<DIDL>
  <Declarations>
    <Descriptor>
      <Statement type="text/xml">
        <mpeg7:Mpeg7>
          <mpeg7:Description xsi:type="mpeg7:ContentEntityType">
            <mpeg7:MultimediaContent xsi:type="AudioVisualType">
              <mpeg7:AudioVisual>
                <mpeg7:MediaLocator>
                  <mpeg7:MediaUri>FamilyAnimalsHH.xml</mpeg7:MediaUri>
                </mpeg7:MediaLocator>
              </mpeg7:AudioVisual>
            </mpeg7:MultimediaContent>
          </mpeg7:Description>
        </mpeg7:Mpeg7>
      </Statement>
    </Descriptor>
  </Declarations>
  <Item>
    <Component>
      <Resource ref="FamilyAnimalsHL.xml"></Resource>
    </Component>
    <Component>
      <Resource ref="FamilyAnimalsHH.xml"></Resource>
    </Component>
    <Component>
      <Resource ref="FamilyAnimalsHBW.xml"></Resource>
    </Component>
  </Item>
</DIDL>
```

Figure 5.2: MPEG-21 File Example

Web Browsing As our proposed UMA System is based on a web server, there has to be an application running that listens to the requests made by the client's browser.

Database Management It is impossible to manipulate a great amount of data without having an application that permits browsing through it. Moreover, due to the direct relation between the XML syntax and the system's data files, a native XML database is needed.

Streaming Management A crucial step in the retrieval of media is the streaming of files. To accomplish this, a streaming application is necessary. This application opens the connection with the client's device

to stream data, controls the stream while loading, buffering and displaying the content at the user's side, and finally closes the stream.

5.2 Organization of the Client

There is no need for specific software implementation of the client side as all the procedure is done by web browsing. However, special plug-ins to manage interoperability with the server, such as a media player, are needed.

Furthermore, one should not forget the specification of the Device Capabilities which is stored in the client device (Section 5.2.1).

5.2.1 Client Device Capabilities

One of the objectives of UMA is to let different devices connect with the system. Different devices usually have different capabilities. This is why a standardized way to describe device capabilities is needed. As seen in Section 4.2 these descriptions will be matched with media properties descriptions to discriminate these files that cannot be played. Table 5.1 lists the chosen device capabilities descriptors.

Capability	Purpose	Descriptor
Bandwidth	Specify available BW (kBit/s)	BitRate
Sound capabilities	Specify frequency range (Hz)	FrequencyRange (minValue and maxValue)
Color depth	Sets how many colors the display can show	ColorDepth
Screen resolution	Sets the resolution of the device's screen (pixels)	Screen (width and height)
Maximum Frame Rate	Sets highest rate of frame for the display	MaxFrameRate

Table 5.1: Table for Client Device Capabilities and corresponding custom descriptors

Taking advantage of the fact that the Database Management is XML compliant, Client Device Capabilities are stored in a XML file. Figure 5.3 gives a sample of a device capabilities file.

```
<?xml version="1.0"?>
<ClientDeviceCapabilities>
    <BitRate>128</BitRate>
    <FrequencyRange>
        <minValue>0</minValue>
        <maxValue>12000</maxValue>
    </FrequencyRange>
    <ColorDepth>256</ColorDepth>
    <Screen>
        <width>800</width>
        <height>600</height>
    </Screen>
    <MaxFrameRate>30</MaxFrameRate>
</ClientDeviceCapabilities>
```

Figure 5.3: Client Device Capabilities File Example

5.3 Content Retrieval

This Section gives a detailed description of how the system performs retrieval. Particular attention is given to content filtering and sorting, according to client device capabilities and user preferences.

5.3.1 Client Device Capabilities Filter

One objective of the UMA System is to sort out files that cannot be played by the client device. This Section describes the algorithm followed to perform this filtering.
To achieve this operation two inputs are used. On one side, we have the media files description in terms of media properties such as Target bit rate, Audio, ... (Section 4.2). On the other side, there is the specification of the device in terms of Bit Rate, Color Depth, ... (Section 5.2.1). In order to generate the content list that matches the client device best, the CDC Filter has to relate both inputs together (Figure 5.4). This is done as follows:

Bitrate This is compared directly with the available Bit rate of the client device.

audioRating This descriptor specifies what kind of audio is contained in the media. audioRating takes one of these values: *speech only*, *speech and low quality music*, *speech and high quality music*, *low quality music*, *high*

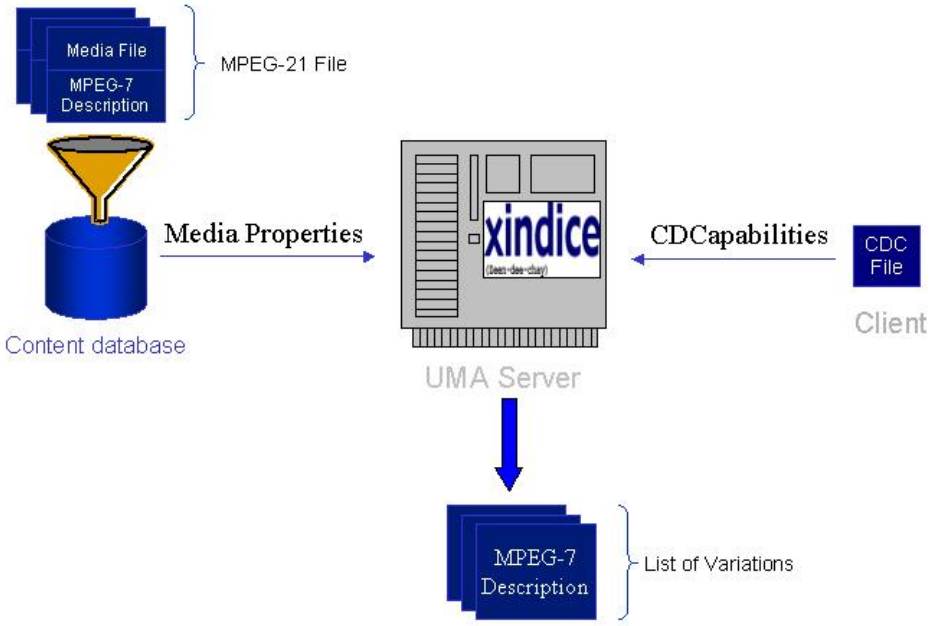

Figure 5.4: CDC Filter relates Media Properties and Client Device Capabilities to output the filtered list of variations

quality music or *none* (no sound at all). To compare this with the audio frequency ranges of the client device there is a need for quantification. Figure 5.5 from the Internet Sound Institute [13], gives an idea of these limits. Table 5.2 lists the values chosen for the system.

colorDomain and colorImportance colorDomain can be either *colored*, *gray level* or *black and white*. colorImportance is a subjective descriptor that attributes an importance value to color. Possible values are *Unimportant*, *Useful*, *Important* and *Essential*. Both descriptors are matched using the quantification showed in Table 5.3.

Frame width and height As with Bit rate, this is compared directly with the available resolution (Screen width and height) of the client device.

Frame Rate is compared with the MaxFrameRate of the client device.

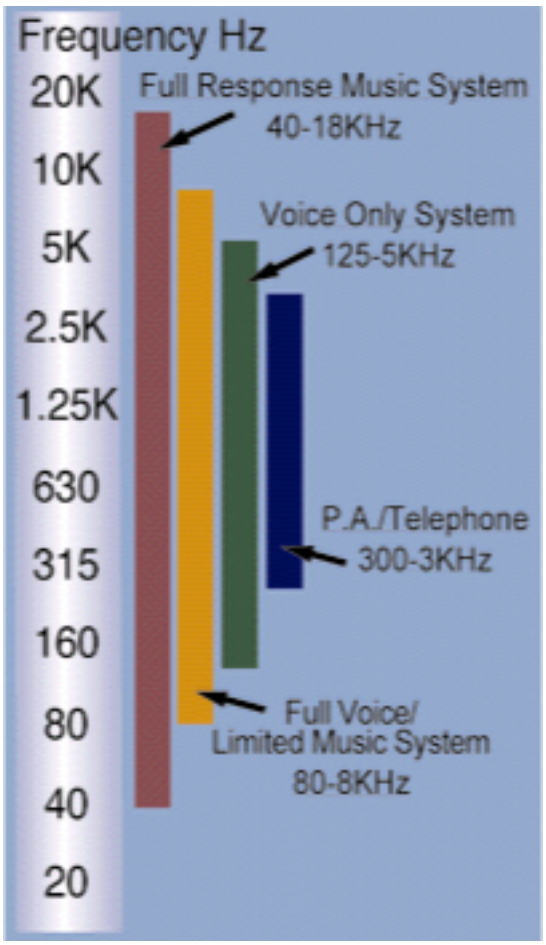

Figure 5.5: Voice and Music Frequency Ranges [13]

Next is described how the filter weights these descriptors. In each MPEG-21 file there are one or more references to MPEG-7 descriptions which represent media properties variations of the same content (Section 5.1.1). The filter has to select the "best" variation. Two steps are taken to achieve this: first, the variations that are not compliant with the limitations of the device are supressed; next each valid variation is "rated". Depending on which element is rated, a different punctuation is given (e.g., if media is colored +20 points are added, while if audio is of low quality only one point is added). Table 5.4 shows the punctuation given to each element. The variation with the highest punctuation is the one chosen.

Figure 5.6 shows an example with three variations a, b and c: while variation b is excluded directly because frequency range does not reach the

audioRating	FrequencyRange	
	minValue (Hz)	maxValue (Hz)
Speech only	125	5000
Speech and low quality music	125	8000
Low quality music	125	8000
Speech and high quality music	125	18000
High quality music	125	18000

Table 5.2: Relation between audio properties of media and necessary available sound bandwidth of client device

colorDomain	ColorDepth
binary	2
graylevel	256
colorImportance	**ColorDepth**
Unimportant	16
Useful	16
Important	256
Essential	65536 (16-bit)

Table 5.3: Relation between color properties of media and necessary color depth of client device

Descriptor	Value	Points
colorDomain	color	+20
	graylevel	+10
	B&W	+0
resolution		$\frac{Frame\_height}{Device\_height} + \frac{Frame\_width}{Device\_width}$
rate		$\frac{Frame\_rate \cdot 2}{Device\_frame\_rate}$
audioRating	High Quality	+2
	Low Quality	+1
	Speech only	+0

Table 5.4: Rating values for each descriptor

bandwidth of High Quality Music (Table 5.2) the other ones are rated. Finally, variation c is chosen (it's rate is calculated as follows: 20 (color) +1 (Music LQ) $+\frac{400}{800}$ (width) $+\frac{300}{600}$ (height) $+\frac{15 \cdot 2}{30}$ (frame rate) = 23).

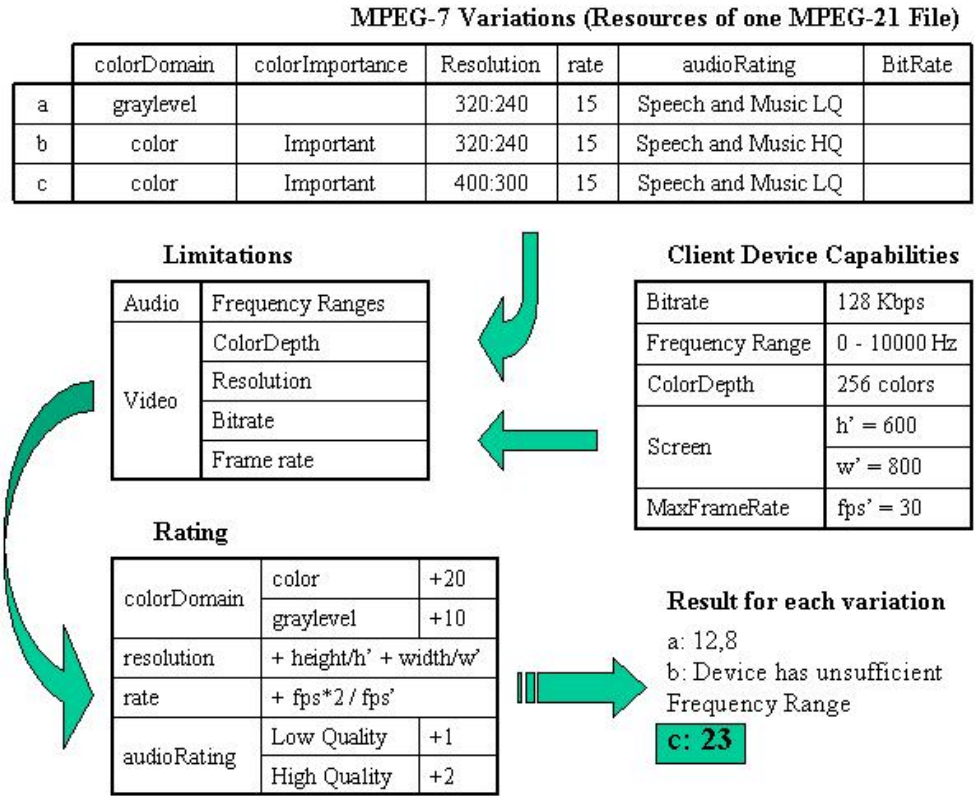

Figure 5.6: Example of Client Device Capabilities Filtering: with media properties and CDCapabilities, the filter extracts the variations that are not compliant with the device limitations and then rates the valid variations.

The filter actually works on all the list of MPEG-21 files. It outputs a list of "best" variations, so no more than one variation per content takes next step.

5.3.2 User/Server Preferences Filter

This Section explains how the content is sorted according to user and server preferences.

User and Server Preferences

The goal is to produce a list of files ordered according to user preferences. In this list, media files on top are more important than those beneath. The proposed solution is to consider the descriptors used when annotating content for preferences, but with a specific value. A *preference* is the couple composed by an MPEG-7 Descriptor and its Descriptor Value (e.g., Form: Bulletin, Genre: Sports). Hence, *importance* is here understood as the amount of coincidences between annotated media descriptions and preferences.
Once the concept of preference is defined, we can take one more leap: it is basic to be able to detect the files that have the preferred content, but also to find which are the "most" preferred files. With this objective in mind, a numeric *weight* attribute is defined for each preference. The higher the weight is, the more important the corresponding descriptor is considered.

For the ease of implementation, user preferences are stored in custom XML rather than MPEG-7. Figure 5.7 gives an example of the structure.

```
<List>
  <Language weight="5">es</Language>
  <Language weight="1">cat</Language>
  <Form weight="1">Magazine</Form>
  <Genre weight="2">Entertainment</Genre>
  <Genre weight="1">Sports</Genre>
  <Genre weight="10">Leisure</Genre>
</List>
```

Figure 5.7: *List of user preferences* example

Client-server applications give the administrator a certain control over the content that can be accessed. Instead of giving explicit control, the administrator has his/her own preferences. This list of preferences is similar to the user one's but in addition it has the *overwrite* attribute. This attribute is proposed to give special rights to the server administrator. This attribute is a selector (takes value 0 or 1) that overrides these preferences of the user which have the same descriptor no matter what their value is. Figure 5.8 shows what a server preferences list looks like. Note that the default value of the *overwrite* attribute is 0 (no override).

```
<List>
    <Form weight="7" overwrite="0">Interview</Form>
    <Genre weight="3">Daily News</Genre>
    <Language weight="5" overwrite="1">en</Language>
</List>
```

Figure 5.8: *List of server preferences* example

Ordering lists

The purpose of having defined user and server preferences is to order lists of media files. The algorithm set out below takes three separate steps:

1. Obtain the "total preferences" from the user and the server preferences. Total preferences are a list of preferences that take the following criteria into account:
   - If overwrite is activated (value of 1) for a given server preference, this preference is kept in the total preferences list, and none of the user preferences with the same descriptor (no matter what the value is) are selected.
   - If a given preference coincides (descriptor and value, the whole couple) in user and server preferences, the preference is added to total list but the weight is a mean of both weights, the user's and the server's.
   - If one descriptor (D) of a preference is in both user and server list, but with different value (which in fact represents different preferences), both the user preference (descriptor D, value_u and its weight_u) and the server preference (descriptor D, value_s and its weight_s) are included in total list.
   - If one descriptor from user preferences is not in server's, or vice-versa, the corresponding preference (descriptor, value and its weight) is included in total list.

   Figure 5.9 is an example of the results of this algorithm.

2. Rate each media file. The procedure to calculate the rate of each file is:
   (a) An item of total preferences list is taken. The descriptor-value couple is searched in the MPEG-7 description of the media file.

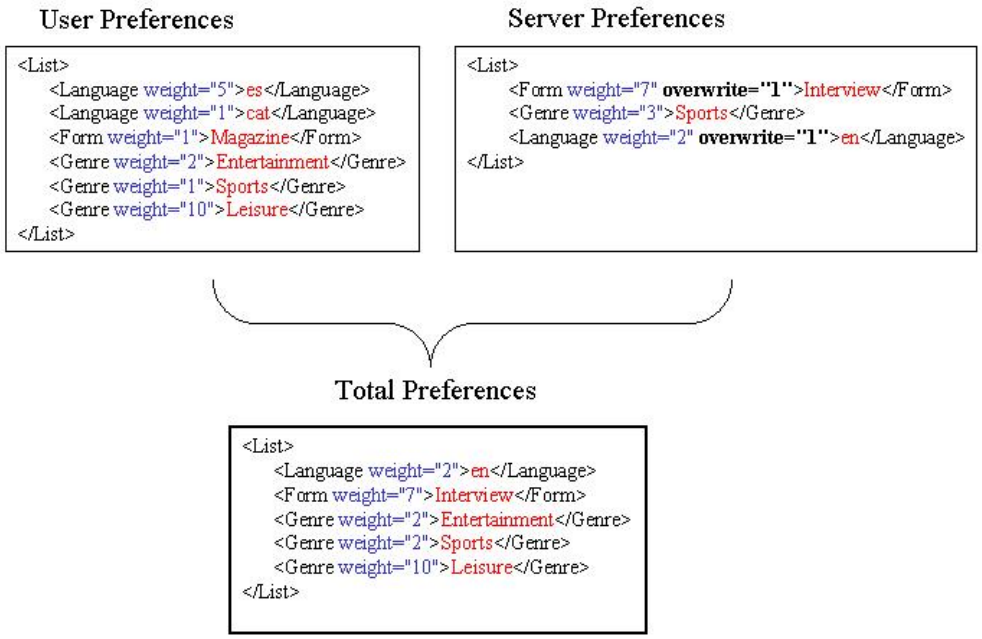

Figure 5.9: Example of creation of total preferences list: first, the server preferences that override are included in total list and the equivalents from user preferences are skipped (this is the case of Form="Magazine", Language="es" and Language="cat"); then, as Genre="Sports" coincides on both list of preferences, this preference is included but with a weight equal to the means of weights (weight=(1+3)/4=2); finally, the preferences that do not coincide like Genre="Entertainment" or Genre="Leisure", are included in total preferences list.

(b) If the couple coincides, a numeric value equal to the weight of the preference is added to *rate*. Otherwise the preference is skipped.

(c) Restart with next item until there are no more items to search.

3. Order list of media files from the best rate (greatest number) to the worst.

5.3.3 Form Filter

A client is used to navigate through the Internet using engines that facilitate the, sometimes tough, task of searching. The Form Filter is a pseudo-search engine.
Once the list of files is ordered and presented to the client, the systems lets him/her fill a form of "preferences". When submitting this form, the system reorders the presented list with the new preferences, ignoring the user and server preferences and only focusing on what the user has just selected. Unlike preferences in Preferences Filter, these coming from the form have a normalized weight attribute with 1 as value. In addition, there is a search by keywords. Whatever words the user fills keywords with, the engine searches for them in each element of the list that the preferences filter has output. Descriptors whose content is compared are: Title, Abstract, Location (Name and Region) and FreeTextAnnotation.

5.3.4 Automatic User Preferences Actualization

In Section 5.3.2, preferences are defined as a static description of preferred content. As user preferences are often not static, adaptation to preferences evolution is needed. User Preferences Actualization is the tool used to adapt the preferences, taking into account the content selections of the user.

Apart from user preferences, the only input the system gets from the user is which files he/she asks for retrieval. Therefore, the analysis is solved by logging part of the content description of each retrieved file. At this step, the administrator has the opportunity to chose which MPEG-7 Descriptors from the annotations have to be logged. This storage is kept in each user preferences file in a new list called *History*. This list has an element that counts the number of retrievals (*Times*) Figure 5.10 shows the result of logging user behavior.

Reporting which content the user retrieves is useless in case this has no effect in user preferences. This is why every ten times a user preforms retrieval,

```
<History>
    <Times>5</Times>
    <Location>Arles,</Location>
    <Form>Documentary,Entertainment,Sequence,Entertainment,</Form>
    <Genre>Leisure,Information,Arts and Media,Video surveillance,Sports,</Genre>
    <Language>en,en,en,fr,</Language>
</History>
```

Figure 5.10: History list in user preferences file

user preferences are actualized using the information stored in History list. The method used is next explained.

1. Items in History are searched in list of preferences for coincidences.
2. If descriptor-value couple is same as in history, weight of preference is increased.
3. Otherwise it is added as a preference with a new weight.

Values for the quantity the weight has to increase or for the value a new weight has to take, are an opened criterium. For this application, we arbitrary define that the value to add is +0.1 per user selection of a determined content. This means that selecting 10 times one specific content is equivalent, in terms of weight, to an initial preference (`weight="1"`).

5.4 Client-Server Application Implementation

The implementation of the UMA demonstrator starts from the idea of taking several steps to achieve the retrieval of media. All these steps are of much importance. More relevant than the streaming itself is how media is organized and selected for retrieval.

5.4.1 Necessary Software

Rather than developing custom solutions, it is better to use standardized packages and profit their powerful tools to organize the information.

Server Applications

Following the principles set before in 5.1.2, this Section presents the server applications used for the implementation.

Java Server There has to be an application listening to the requests of the client's browser. For this purpose a servlets container implemented by Apache called Apache Tomcat Server (version 4.0) has been selected. This application supports JavaServlet and Java Server Pages technologies. The reason to select this applications is the fact that it is a widely extended software, used in many Internet web servers.

Apache Xindice There has to be an XML native database to manage the collection of XML files of the system. Apache Xindice is the selected database manager. It has all its core API in Java and implements all the tools that the UMA System needs. These are: treatment of DOMs[1], possibility of querying XML files with XPath [2] and possibility to actualize content of XML files with XUpdate (Xindice's own API).

Real Server To control the streaming stage of the system a specific server is needed: from Real Networks, the Real Server (version 8.0). This application uses RTSP (Real Time Streaming Protocol) which is an Internet Engineering Task Force (IETF) Proposed Standard for control of streaming media on the Internet.

Client Applications

There are some tools that the environment needs to work in the client device.

Java Plug-In Some standard browsers are not originally prepared to run Java applets. That is why the user has to download the Java Plug-In from Sun Microsystems.

Java Security Policy File In our client-server application, the Client Device Capabilities file has to be read and its information sent to the server. To access this data, the server asks the Java Security Manager of the user. This Manager reads a file stored in the device that sets the permissions for any running Java application. Our UMA Demonstrator let's the user download a *java.policy* file that enables this access.

Real One Player As the actual server used for the streaming stage is from Real Networks, their own client application is needed to play media.

[1]The Document Object Model is an API that allows programs and scripts to dynamically access and update the content, structure and style of documents [3].

Chapter 6

Experiments

In this chapter, the whole UMA environment, annotation tool and client-server application is tested. Following the logical path set by the environment itself, we first annotate a collection of media files using the annotation tool. Some of these media files have the same content but with different media properties (resolution, audioRating, ...). We group these *variations* of the same content using the MPEG-21 DID [7]. We use a scenario with three different user preferences files (simulating 3 different users) and four client device capabilities files (simulating the browsing from 4 different devices) for the tests.

We conclude with the limitations of the environment.

6.1 Content Annotation

As seen in Chapter 4, the Annotation Tool is used to generate the description of media files. In this Section, we explain how this tool is used for the annotation of video content. The annotated video is part of a set of variations that will be used for the tests of Section 6.2.

The media file chosen is *FamilyAnimalsHL.avi*. It is a segment extracted from the *animals* sequence, taken from the MPEG-7 Content Set [6].

We open the media file with the annotation tool. Next we start with a new description. We annotate some features and properties manually: Title, Abstract, Form, Genre and Language, and colorDomain, Frame rate, colorImportance and audioRating. Some other media properties are annotated automatically using the metadata of the media file: MediaIdentification, FileFormat, FileSize, Frame height and width, and MediaTime. For this video, we don't describe any subsequences. Figure 6.1 shows the application while

we annotate the description. Finally, we save the description in a file that we call *FamilyAnimalsHL.xml* (Figure 6.2).

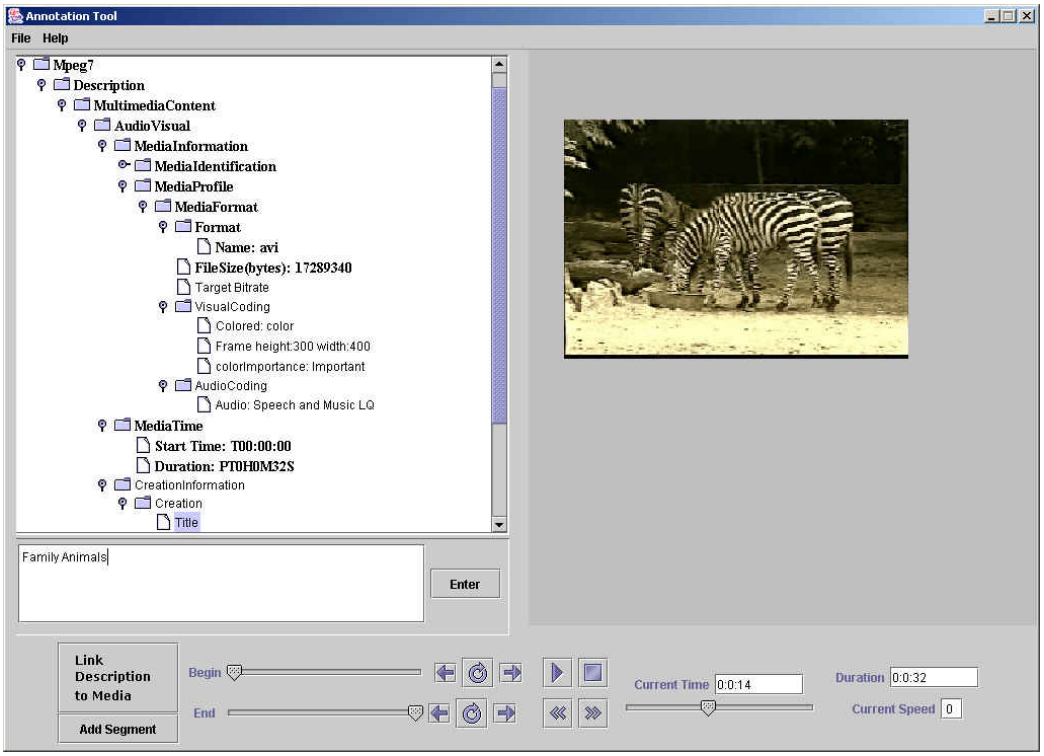

Figure 6.1: Annotation of *FamilyAnimalsHL.avi* using the Annotation Tool

```
<?xml version="1.0" encoding="UTF-8"?>
<Mpeg7>
  <Description xsi:type="ContentEntityType">
    <MultimediaContent xsi:type="AudioVisualType">
      <AudioVisual>
        <MediaInformation>
            <MediaIdentification>
                <EntityIdentifier organization="MPEG"
                type="MPEG7ContentSetId">FamilyAnimalsHL</EntityIdentifier>
            </MediaIdentification>
            <MediaProfile>
                <MediaFormat>
                    <FileFormat href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:">
                        <Name>avi</Name>
                    </FileFormat>
                    <FileSize>17289340</FileSize>
                    <BitRate/>
```

```
            <VisualCoding>
                <Format colorDomain="color"/>
                <Frame height="300" rate="15" width="400">
                </Frame>
                <colorImportance>Important</colorImportance>
            </VisualCoding>
            <AudioCoding>
                <audioRating>Speech and Music LQ</audioRating>
            </AudioCoding>
        </MediaFormat>
    </MediaProfile>
</MediaInformation>
<MediaTime>
    <MediaTimePoint>T00:00:00</MediaTimePoint>
    <MediaDuration>PT0H0M32S</MediaDuration>
</MediaTime>
<CreationInformation>
    <Creation>
        <Title>Family Animals</Title>
        <Abstract>
            <FreeTextAnnotation>Sequences with families of
            different type of animals
            </FreeTextAnnotation>
            <StructuredAnnotation>
                <Who>
                    <Name/>
                </Who>
            </StructuredAnnotation>
        </Abstract>
        <Creator>
            <Role href="urn:mpeg:mpeg7:cs:RoleCS:2001:">
                <Name/>
            </Role>
            <Agent xsi:type="PersonType">
                <Name>
                    <GivenName/>
                    <FamilyName/>
                </Name>
                <Icon>
                    <MediaUri/>
                </Icon>
            </Agent>
        </Creator>
        <CreationCoordinates>
            <Location>
                <Name/>
                <Region/>
                <AdministrativeUnit/>
            </Location>
```

```
                    <Date>
                        <TimePoint/>
                    </Date>
                </CreationCoordinates>
                <CopyrightString/>
            </Creation>
            <Classification>
                <Form href="urn:mpeg:mpeg7:cs:FormatCS:2001:">
                    <Name>Documentary</Name>
                </Form>
                <Genre href="urn:mpeg:mpeg7:cs:ContentCS:2001:">
                    <Name>Information</Name>
                </Genre>
                <Language>en</Language>
            </Classification>
        </CreationInformation>
        <TemporalDecomposition>
            <AudioVisualSegment></AudioVisualSegment>
        </TemporalDecomposition>
        <FreeTextAnnotation/>
      </AudioVisual>
    </MultimediaContent>
  </Description>
</Mpeg7>
```

Figure 6.2: MPEG-7 description of the *FamilyAnimalsHL.avi* sequence

6.1.1 Content set

Using the Annotation Tool we obtain the MPEG-7 descriptions that the test needs. Moreover, we group the variations in MPEG-21 files (Section 5.1.1). Table 6.1 is a summary of the stored descriptions. Note that we limit the content features annotation to Title, Form, Genre and Language, for testing purposes. We have skipped the Bitrate property because we use a streamer server to control the bandwidth, so the Bitrate descriptor does not affect the system performance.

Content Features				MPEG-7	audio	color	color	Frame	
Title	Form	Genre	Lan.	Variation	Rating	Domain	Importance	WxH	rate
Wonders Around Us	Documentary	Information	en	animals	S&MLQ	color	Important	320x240	25
Anton i David amb Webcam	Entertainment	Leisure		AntoniDavid AntoniDavidBW		color graylevel	Useful	320x240 320x240	
Family Animals	Documentary	Information	en	FamilyAnimalsHH FamilyAnimalsHL FamilyAnimalsHBW	S&MHQ S&MLQ S&MHQ	color color graylevel	Important Important	320x240 400x300 320x240	15 15 15
Nike Football Ad.	Entertainment	Sports	fr	nikefootballHH nikefootballHL nikefootballLBW	S&MHQ S&MLQ S&MLQ	color color graylevel	Important Important	320x240 400x300 320x240	15 15 15
Angie	Entertainment	Music	en	RollingStones-AngieH RollingStones-AngieL	S&MHQ S&MLQ				
Vincent's Bedroom in Arles	Entertainment	Arts&Media		vgbedroom vgbedroomBW		color graylevel	Essential	513x387 513x387	
Video Surveillance sequence		Information		VideoSurv VideoSurvBW		color graylevel	Useful	320x146 320x146	15 15

Table 6.1: Annotated variations for tests

6.2 Browsing and retrieval

After a presentation of the test framework, browsing and retrieval is tested using a particular user-device combination. The results obtained with other combinations are then summarized.

6.2.1 Test Framework

We use 7 different contents stored in 15 variations. The tests are done with 4 different devices and 3 users.

Client devices

Three devices have been selected for the tests: a standard PC, a generic Personal Digital Assistant (PDA), and the Sony Ericsson T68 Mobile phone. In all cases, we have supposed access to the server over the World Wide Web. We suppose that the PDA can be used either with headphones or internal speakers; both cases are here simulated. This results in 4 CDC Descriptions (Table 6.2).

	Bitrate	FrequencyRange		ColorDepth	Screen		Max
		minValue	maxValue		width	height	FrameRate
PC	1000000	20	20000	24 bits	1024	768	30
PDA (H.P.)	11000	20	20000	16 bits	320	240	15
PDA (I.S.)	11000	100	12000	16 bits	320	240	15
Sony Ericsson T68	2000	150	3700	256 colors	80	101	15

Table 6.2: CDCapabilities for test

Users

To simulate diverse users, we register 3 different User IDs in the system and select different preferences for each one of them. To compare a limited range of content features only those annotated in the variations (Form, Genre and Language) are chosen for the users preferences. Table 6.3 shows the preferences for each user. In parenthesis, the weight of each preference is given.

Additionally, we chose several preferences for the administrator (Table 6.4) None of them overrides the corresponding user preference, and they all have weight 1.

User ID	Form(s)	Genre(s)	Language(s)
user1	Documentary(1)		en(1)
user2	Entertainment(1)	Music(1)	
user3		Information(1) Arts and Media(1)	en(1)

Table 6.3: User preferences for test (in parenthesis the weight of the preference)

User ID	Form(s)	Genre(s)	Language(s)
Admin	Interview(1)	Daily News(1)	en(1)

Table 6.4: Server preferences for test (in parenthesis the weight of the preference)

6.2.2 Step-by-step browsing and retrieval (particular case)

The full browsing and retrieval procedure is shown step-by-step for a particular user-device combination. We arbitrarily chose user2 (Table 6.3) and the PDA with internal speaker (Table 6.2) as retrieval device. Next Sections present the steps taken to achieve the browsing and retrieval for a new user.

Home Page

With a standard Internet browser (e.g., Microsoft Internet Explorer) we access the home page of the UMA system (Figure 6.3).

Registration of a new user

We open the link to the "New Registration". First, we enter the desired user ID (in this case *user2*) and a password. Then, we select the preferences (Figure 6.4).

Finally, we submit the form to the server. If there is no other user with the same name, a new file called *user2.xml* is stored in the server's database (Figure 6.5).

Then, the server redirects to another Web page (Figure 6.6). There, the system gives the option to re-edit preferences ("Your Preferences") or start the browsing and retrieval of media ("Start UMA").

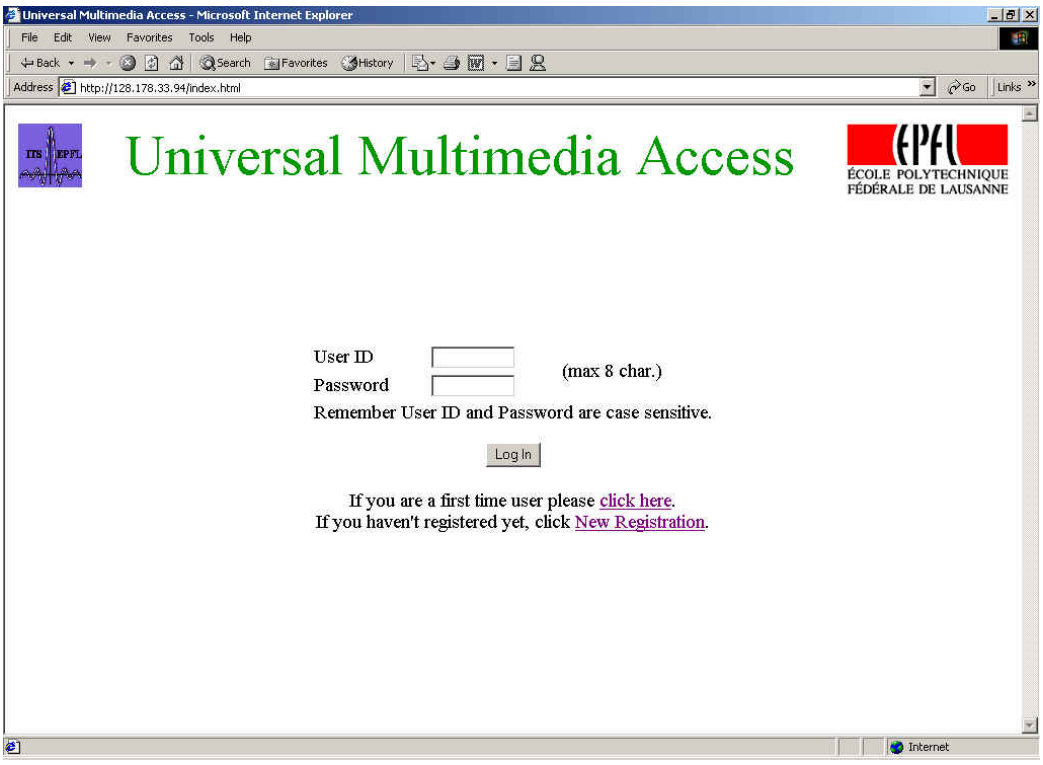

Figure 6.3: *Home Page* of the UMA Demonstrator

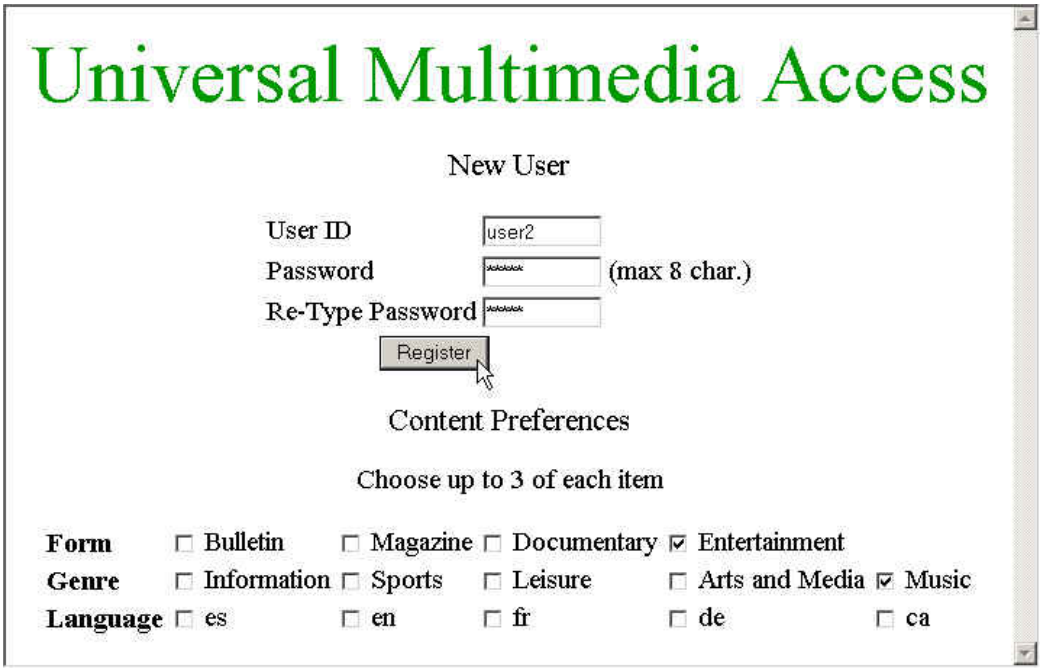

Figure 6.4: *New Registration Page* of the UMA Demonstrator

```
<?xml version="1.0" encoding="utf-8"?>
<UserPrefs>
  <User>
      <ID>user2</ID>
      <PW>hello</PW>
  </User>
  <List>
      <Form weight="1">Entertainment</Form>
      <Genre weight="1">Music</Genre>
  </List>
  <History>
      <Times></Times>
      <Location></Location>
      <Form></Form>
      <Genre></Genre>
      <Language></Language>
  </History>
</UserPrefs>
```

Figure 6.5: The *user2.xml* user preferences file

Figure 6.6: *User Logged In* Page of the UMA Demonstrator: lets one chose between re-editing preferences or sending the CDC description to start the browsing.

Send CDC File

On selection of "Start UMA", a Java applet reads the CDC file stored in the client device and sends it to the server. The server application performs the CDC Filtering with this information. The list of selected valid variations that the filter outputs is not visible to the user. For each variation, there can be one or more incapabilities that make the variation invalid for the specific device. For example, the *animals* variation has greater Frame rate (25 frames per second) than the one supported by the PDA (MaxFrameRate = 15). The list of suppressed variations due to incapabilities of the device is shown in Table 6.5, which gives a reason for the suppression of each variation. Note that in the case of *Anton i David amb Webcam* and *Video Surveillance sequence*, none of the variations are suppressed because of device incapabilities. Table 6.6 gives the list of variations after the CDCFilter. These are the variations that match the device capabilities with the media properties best.

MPEG-7 Variation	audio Rating	ColorDom	colorImp	Frame		CDC Reason for suppression
				WxH	rate	
animals	S&MLQ	color	Important	320x240	25	Frame rate
FamilyAnimalsHH	S&MHQ	color	Important	320x240	15	Sound
FamilyAnimalsHL	S&MLQ	color	Important	400x300	15	Frame W&H
FamilyAnimalsHBW	S&MHQ	graylevel		320x240	15	Sound
nikefootballHH	S&MHQ	color	Important	320x240	15	Sound
nikefootballHL	S&MLQ	color	Important	400x300	15	Frame W&H
RollingStones-AngieH	S&MHQ					Sound
vgbedroom		color	Essential	513x387		Frame W&H
vgbedroomBW		graylevel		513x387		Frame W&H

Table 6.5: Variations supressed by CDC Filter

MPEG-7 Variation	audio Rating	ColorDom	colorImp	Frame	
				WxH	rate
AntoniDavid		color	Useful	320x240	
nikefootballLBW	S&MLQ	graylevel		320x240	15
RollingStones-AngieL	S&MLQ				
VideoSurv		color	Useful	320x146	15

Table 6.6: Variations selected by CDC Filter

Ordered variations according to preferences

With the list that the CDCFilter outputs, the server application measures the rating for each variation according to the user and server preferences files.

Table 6.7 gives the total preferences built by the User/Server Preferences Filter (in parenthesis the weight: as there are no coincidences, all the weights stay with value 1).

Forms	Interview(1), Entertainment(1)
Genres	Daily News(1), Music(1)
Language	en(1)

Table 6.7: Total preferences obtained from user and server preferences files (in parenthesis: weight of the preference)

Finally, the server redirects to a web page with the resulting list of files (Figure 6.7). In this list (Table 6.8), we can see that media whose content matches the total preferences best, is on top: *RollingStones-AngieL* media file has Form="Entertainment", Genre="Music" and Language="es" as content features which are preferences of the total preferences). On the other hand, *VideoSurv* has none of the content features that the user nor the administrator, prefer.

Figure 6.7: *Results Page* of the UMA Demonstrator

	PDA (w. internal speaker)
1	RollingStones-AngieL
2	AntoniDavid
3	nikefootballLBW
4	VideoSurv

Table 6.8: Ordered variations for user2 with the PDA with internal speaker

Search by form

Now, we reorder the list searching for specific content. We enter the word *football* in Keywords and select Form *Bulletin*, in the form presented beneath the list of files (Figure 6.8).

The Form Filter (Section 5.3.3) takes the last list presented by the server, and reorders it taking as preferences the newly selected elements of the form (Figure 6.9). In the list obtained (Table 6.9), we can see that the reordering makes *nikefootballLBW* appear the first as it has *football* as one of the words of the title. As there is no media with the content feature Form with value *Bulletin*, this feature is not taken into account when ordering.

Figure 6.8: *Results Page* of the UMA Demonstrator: user fills the form situated beneath the list to search for specific content

Figure 6.9: *Results Page* of the UMA Demonstrator after submitting the form (Keywords: *football*, Form(s): *Bulletin*)

	PDA (w. internal speaker)
1	nikefootballLBW
2	RollingStones-AngieL
3	AntoniDavid
4	VideoSurv

Table 6.9: Ordered variations for user2 with the PDA with internal speaker after submitting the form (Keywords: *football*, Form(s): *Bulletin*)

Content Retrieval

We select the *nikefootballLBW* variation for streaming. Then, the server actualizes the *user2.xml* file with the content features specified in the *History* list (Figure 6.10). Finally, streams the media file described by the corresponding variation (Figure 6.11).

```
<?xml version="1.0" encoding="utf-8"?>
<UserPrefs>
  <User>
      <ID>user2</ID>
      <PW>hello</PW>
  </User>
  <List>
      <Form weight="1">Entertainment</Form>
      <Genre weight="1">Music</Genre>
  </List>
  <History>
      <Times>1</Times>
      <Location></Location>
      <Form>Entertainment,</Form>
      <Genre>Sports,</Genre>
      <Language>fr,</Language>
  </History>
</UserPrefs>
```

Figure 6.10: The *user2.xml* user preferences file after the actualization of History

6.2.3 Results for all cases

This Section presents a collection of lists showing the variations ordered by User Preferences Filter after all the previous steps of the procedure. To create these lists, we have accessed the system with all the combinations of users and devices (Tables 6.10, 6.11 and 6.12).

	PC	PDA (w. headphones)	PDA (w. internal speaker)	T68
1	animals	FamilyAnimalsHH	RollingStones-AngieL	—
2	FamilyAnimalsHH	RollingStones-AngieH	AntoniDavid	
3	RollingStones-AngieH	AntoniDavid	nikefootballLBW	
4	AntoniDavid	nikefootballHH	VideoSurv	
5	nikefootballHH	VideoSurv		
6	vgbedroom			
7	VideoSurv			

Table 6.10: Ordered variations for user1 with the PC, PDA with headphones, PDA with internal speaker and Sony Ericsson T68

In the case of the Sony Ericsson T68, the results are empty lists. The main reasons are the frame resolution and the sound capabilities of this device. As there is no media with lower resolution than 80x101, or only speech as audio media, none of the variations stored for test can be played with this

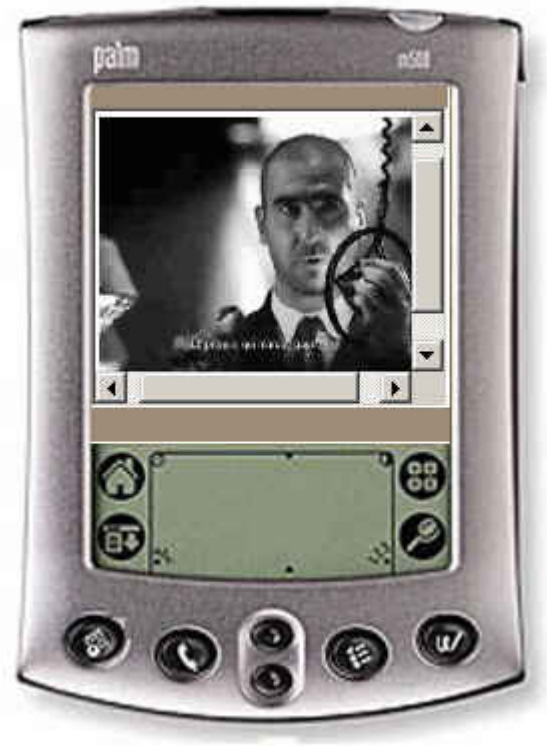

Figure 6.11: Retrieval of the *nikefootballLBW* variation with the UMA Demonstrator

	PC	PDA (w. headphones)	PDA (w. internal speaker)	T68
1	RollingStones-AngieH	RollingStones-AngieH	RollingStones-AngieL	—
2	animals	AntoniDavid	AntoniDavid	
3	AntoniDavid	FamilyAnimalsHH	nikefootballLBW	
4	FamilyAnimalsHH	nikefootballHH	VideoSurv	
5	nikefootballHH	VideoSurv		
6	vgbedroom			
7	VideoSurv			

Table 6.11: Ordered variations for user2 with the PC, PDA with headphones, PDA with internal speaker and Sony Ericsson T68

	PC	PDA (w. headphones)	PDA (w. internal speaker)	T68
1	animals	FamilyAnimalsHH	RollingStones-AngieL	—
2	FamilyAnimalsHH	RollingStones-AngieH	VideoSurv	
3	RollingStones-AngieH	VideoSurv	AntoniDavid	
4	vgbedroom	AntoniDavid	nikefootballLBW	
5	VideoSurv	nikefootballHH		
6	AntoniDavid			
7	nikefootballHH			

Table 6.12: Ordered variations for user3 with the PC, PDA with headphones, PDA with internal speaker and Sony Ericsson T68

device.

After testing the browsing and retrieval, we can observe that the UMA system works as expected.

The system effectively sorts out these variations considered invalid due to device capabilities. Observing the tables of variations, one can see that the collection of variations is the same for each device, no matter who is the user.

Also, the system manages user and server preferences, and orders lists of variations according to preferred content. Observing again the tables of results, one can see that depending on which user accesses the system, the list of files is ordered in a different way.

Finally, the server is able to stream the variation selected by the user.

Additionally, from the results we can draw another conclusion: performance of the system has a strong dependency on which variations are stored. From the limited collection stored in our system, variations like FamilyAnimalsHL or nikefootballHL have not been selected in any of the cases. Therefore, rather than storing any variation it is interesting to prevent useless cases. On the other hand, some devices like PDA cannot display, for example, any of the variations of the Van Gogh's picture "Vincent's Bedroom in Arles". In this case, instead of storing a graylevel variation of this picture, it could be more interesting to store a variation with a lower resolution. These limitations are overcome by using the automatic approach.

6.3 Limitations

After testing our UMA environment, we have encountered diverse limitations.

The subjectiveness of some of the properties selected to describe media, affects annotation and matching of these properties with device capabilities. This is the case of descriptors like colorImportance or audioRating. The annotation of media depends on each user: when audio is of High or Low Quality, or when color is Essential or Useful, depend on users likes. Moreover, the value that these descriptors take when annotated, affects the matching with the device capabilities.

The way the implemented CDC Filter works is also subjective. The procedure sets the media variations as unplayable when any of the media properties does not fit the device capabilities. Instead of suppressing invalid variations, the CDC Filter could adapt the variation (e.g., scale the resolution to make it fit the display of a device).

Chapter 7

Conclusions and Future Work

The design and implementation of a UMA environment has offered us the opportunity to view the complexity of a UMA system. First, it gave us an interesting insight into the MPEG standards for multimedia description. These standards brought us to the implementation of an application to annotate media descriptions. Then, we built an application capable of delivering media under the principals of UMA: treat user preferences and device capabilities. Finally, the development of the demonstrator gave us an idea of the distance between theoretical background and software implementation.

This Chapter proposes extensions and improvements of the theoretical design and of the performance of our implementation. We conclude this work with a summary of the achieved results.

7.1 Possible extensions and improvements

This Section shows how the environment could be extended and improved, through the application of lately defined standards and through the application of better software solutions.

From the point of view of theoretical improvements, a possible extension of the system would be the usage of the MPEG-21 Digital Item Identification and Description (DIID), and the MPEG-21 Digital Item Adaptation (DIA).

For our demonstrator, we used a relatively small media database. Increasing this database, makes it more difficult to identify corresponding MPEG-7 and MPEG-21 Descriptions. For this reason we propose to use DIID standard for a better identification of files.

Due to timing of MPEG, DIA could not be applied to our system. Usage of DIA could benefit the system as descriptions for adaptation of content

have been specified in this standard. Moreover, applying this standard would only change the descriptions but not the proposed features, so changes could be easily implemented.

Diverse means to improve the implementation of the UMA environment are also available.

First of all, to let the demonstrator work on any device, all devices should be able to run an HTML browser, Java applets and store an XML CDC File. So far, most pocket devices (PDA, mobile phones) do not offer these functionalities.

To approach the environment to the UMA "scalable summary" solution, "on-the-fly" adaptation could be performed by the system. Special attention is given to the MPEG-7 Camera[11] and Scene Cut Detection[12] techniques. Also, it would be interesting to permit the retrieval of audio or video clip segments, since the Annotation Tool is already able to describe segments.

Another drawback of the system is that not all the power of database management is used. Actually, our server has a rather slow response compared to other server applications. Also, the environment has a limitation using XML schema and namespaces: they cannot be validated by the annotation tool, nor by the Apache Xindice database, due to Java XML API limitations. Therefore, attention to state-of-the-art of XML tools should be applied.

Deepening on system's performance, we have realized that the "weight" attribute of user and server preferences can generate unexpected results. While the weight of the user preferences grows more and more as the client retrieves media, the weight of the server preferences stays the same. In addition, the server preferences are unique for all the users, so they don't make the same effect on all users. To solve this problem, we propose a normalization of the weights of the users preferences according to server preferences, every time the first ones are actualized.

Security management is another issue for future work.

7.2 Conclusions

In this work, we designed and implemented a UMA environment using open standards. The result of this development has brought us the Annotation Tool, to describe media using the MPEG-7 standard, and a Client-Server Application, to enable client browsing and retrieval of content. Even though the system showed some limitations, Universal Multimedia Access (description of media, and treatment of user preferences and client device capabilities)

has been achieved.

Development of a UMA environment has brought us an insight into the application of open standards to a real system. Their interoperability with upcoming extensions and applications opens many new paths.

David Marimón Sanjuán

Lausanne, 11th September 2002

Appendix A

How To Use The Annotation Tool

The MPEG-7 Annotation Tool is an application for the manual content description of Media (images, audio, video) using the MPEG-7 standard. I takes a media file as an input and permits its description using user-friendly annotation. The final output is an MPEG-7 description file.

A.1 Functionality

A.1.1 MPEG-7 Description

All description files are formed using MPEG-7. As this is a XML based standard the application shows the content of these (.mp7 or .xml) files in a tree where each node is a MPEG7 tag. For user-friendlyness, MP7 tags can be translated to more comprehensive screen names.

1. New Description
   To start a new description, a template must be opened using this command.

2. Open Description
   To open any kind of MPEG-7 file the user has to select this option.

3. Save Description

4. Preview Descrition
   To visualize the output file while editing, it can be previewed on a default browser.

A.1.2 Media

Media files (e.g., mpeg, mov, jpeg, gif, mp3,...) can be played with this application.

1. Open Media
   To open a media file the application opens a browser. Different extensions lead us to different types of media (movies, images, sounds).

2. Play Media
   Once the media is opened depending on its type different environments are loaded: the treatment of time is only useful for audio and video, while images still need to be linkable to the description.
   When Playing Sound and Movies, the environment loaded is the same. In the control panel, right in the lowest part of the window, the user can control speed and time. Begin and End controls are for Segmentation which is explained in next Section.

A.2 How to edit elements

Once the description is open the user can edit the text elements. This is achieved by selecting an element in the tree, entering the text in the Edit panel under the tree and just clicking Enter to update the content (to visualize changes in tree the user has to select another Element).

A.2.1 Link Description to Media

For convenience, the application can add the content of some elements such as FileSize or Resolution directly from the opened media. The "Link Description to Media" button actualizes some Elements depending on the type of media opened:

Movies Media Identification, FileSize, FileFormat, StartTime, Duration, and Frame (width and height).

Sounds Media Identification, FileSize, FileFormat, StartTime and Duration.

Images Media Identification, FileSize, FileFormat and Frame (width and height).

A.2.2 Add Segment

Another capability enabled for movies and sounds is to Add Segments. Using the buttons and sliders related to Begin and End of segment, the user can chose a specific segment in the time line. Once the period is chosen, the user must select an AudioVisualSegment Element in the Tree where the segment will be added, then the Add Segment Button pops up a window with information on Parent's identification (relation with the other elements in the tree), Start and Duration Times and also let's the user identify the segment.

Appendix B

How to Install and Use the UMA Demonstrator

The UMA Demonstrator is a client-server application that permits browsing and retrieval of media taking into account user preferences and client device capabilities.

B.1 The Server

The server application is prepared to run on a Microsoft Windows platform. Although, some of the software is available for different environments, the streamer server used is only available for the Windows environment.

B.1.1 Installation of the necessary server applications

The server applications needed for the system to run are: the Apache Tomcat Server (v 4.0), the Apache Xindice (v 1.0), and the Real Server (v 8.0).

Apache Tomcat Server

This software is used to manage the Java servlets. One can download the Tomcat Server from http://jakarta.apache.org/tomcat/ Once this package is installed (with its own installation), some modifications are needed. In the *server.xml* file, stored in *ApacheTomcatServer_directory/conf/*:

1. Change port="8080" to port="80" in the HttpConnector. By doing so, the server will automatically be accessed using a URL without port specification.

   ```
   <Connector
   ```

```
className="org.apache.catalina.connector.http.HttpConnector"
          port="8080" minProcessors="5" maxProcessors="75"
          enableLookups="true" redirectPort="8443"
          acceptCount="10" debug="0" connectionTimeout="60000"/>
```

2. After the lines set out before, we have to define the DefaultContext by adding this line:

```
<DefaultContext reloadable ="true"/>
```

Apache Xindice

This software is used to manage the XML databases. One can download the Apache Xindice package from http://xml.apache.org/xindice/ Once the package is uncompressed, we can add a BAT file to automatize the start of the Xindice server. This file has to be stored in the root directory of this installation (where the *startup.bat* file of Xindice is stored) and must do the instructions set out below. You have to do a few changes on the paths to match your installations directories.

```
set PATH = %PATH%; xindice_directory\bin
set JAVA_HOME = java_home_directory\bin
set CLASSPATH = %CLASSPATH%; xindice_directory\java\lib\xindice.jar; xindice_directory\jacorb.
set XINDICE_HOME = xindice_directory\ startup
```

Also, we have to ensure that the HTTPServer of Xindice listens to default port 4080. This port is set in the *server.xml* file stored in *Xindice_directory/config/*:

```
<service class="org.apache.xindice.server.services.HTTPServer"
docroot="./docs/" name="HTTPServer">
    <bind chunksize="32768" host="" keepalive="16" linger="1000"
    port="4080" reverse="no" timeout="300" />
```

Real Server

This software is used to manage the streaming of media content. One can download the installation application from http://www.realnetworks.com On the Real Server Administrator 8.0 control page (Windows Start → Programs → RealServer), we have to change the port of the RTSP protocol: Configure → General Setup → Ports, and then change the default port to 558.

B.1.2 Installation of Java applets and servlets

To install the applets and servlets, we only need to extract:

umaJava.zip to the ApacheTomcatServer_directory/webapps/ROOT directory.

umaJAR.zip to the ApacheTomcatServer_directory/lib directory.

B.1.3 Organization of files

Inside the ROOT directory cited in Section B.1.2, we can find:

- The *ServerPrefs.xml* file, where we define specific behavior of the system.
- The directory of the MPEG-7 descriptions.
- The directory of the MPEG-21 descriptions.
- The directory of the user preferences files.

B.1.4 Start the server

To start the server we have to start the three server applications:

Apache Tomcat Server Windows Start → Programs → Apache Tomcat 4.0 → Start Tomcat.

Apache Xindice Execute the BAT file created in Section B.1.1.

Real Server Windows Start → Programs → RealServer → RealServer 8.0

B.1.5 Solving problems

The Tomcat server sometimes is unrestartable. To solve this, in some cases it is enough to change port="8080" to port="8108" of the WarpConnector in the *server.xml* file described in Section B.1.1, and then start the server as usual.

```
<Connector
className="org.apache.catalina.connector.warp.WarpConnector"
     port="8080" minProcessors="5" maxProcessors="75"
     enableLookups="true"
     acceptCount="10" debug="0"/>
```

B.2 The Client

The Client software is prepared to run on a Microsoft Windows platform due to the same reason as the server: the media player software.

B.2.1 Installation of necessary client applications

For the client, two applications are needed: The Java Plug-In and the RealOne Player. Both are downloadable from the Internet:

Java Plug-In http://java.sun.com/getjava/installer.html

RealOne Player http://www.real.com/

B.2.2 The Client Device Capabilities and the *java.policy* file

Both the CDC file and the *java.policy* file can be downloaded and installed from the Home Page of the UMA system. For the CDC file, the user should reedit it according to his/her own device capabilities. The *java.policy* file permits the access of the server to the client device capabilities file stored in the user's device.

Bibliography

[1] World Wide Web Consortium (W3C), Extensible Markup Language 1.0 (Second Edition, www.w3.org/XML/), October 2000.

[2] World Wide Web Consortium (W3C), XML Path Language (XPath) 1.0, www.w3.org/TR/xpath, November 1999

[3] World Wide Web Consortium (W3C), Document Object Model (DOM) Level 1 Specification (Second Edition - W3C Working Draft), www.w3.org/DOM/, September 2000.

[4] P. van Beek, A. B. Benitez, J. Heuer, J. Martinez, P. Salembier, Y.Shibata, J.R. Smith and T. Walker, "Text of 15938-5/FDIS Information Technology - Multimedia Content Description Interface - Part 5 Multimedia Description Schemes" Tech.Rep. N4242, ISO/IEC JTC 1/SC 29/WG 11, Sydney, AU, July 2001.

[5] A. Yamada, M. Pickerings, S. Jeannin, L. Cieplinski, J.-R. Ohm, M. Kim,"Text of 15938-3/FDIS Information Technology - Multimedia Content Description Interface - Part 3 Visual", Tech.Rep. N4358, ISO/IEC/JTC1/SC29/WG11, Sydney, AU, July 2001.

[6] S. Paek, "Description of MPEG-7 Content Set", Tech.Rep. N2467, ISO/IEC JTC1/SC29/WG11, Atlantic City, USA, October 1998.

[7] T. Schwartz, V. Iverson, Y-W. Song, R. Van de Walle, D. Chang and E. Santos, "Text of 21000-2/FCD Information Technology - Multimedia Framework - Part 2 Digital Item Declaration" Tech.Rep. N4530, ISO/IEC JTC 1/SC 29/WG 11, Pattaya, TH, December 2001.

[8] N. Rump, Y-W. Song and H. Sakamoto, "Text of 21000-3/CD Information Technology - Multimedia Framework - Part 3 Digital Item Identification and Description" Tech.Rep. N4532, ISO/IEC JTC 1/SC 29/WG 11, Pattaya, TH, December 2001.

[9] A. Vetro, A. Perkis and S. Devillers, "Text of 21000-7/WD Information Technology - Multimedia Framework - Part 7 Digital Item Adaptation" Tech.Rep. N4820, ISO/IEC JTC 1/SC 29/WG 11, Fairfax, US, May 2002.

[10] O. Steiger, A. Cavallaro and T. Ebrahimi,"MPEG-7 Description of Generic Video Objects for Scene Reconstruction", In Visual Communications and Image Processing (VCIP) 2002, Proc. of SPIE, volume 4671, pages 947-958, San Jose, CA, USA, January 21-23, 2002

[11] O. Steiger,"Smart Camera for MPEG-7", Tech. Rep., Swiss Federal Institute of Technology, Lausanne, 2001.

[12] N. Minh-Thanh, "Démonstrateur d'Accès Multimédia Universel", Tech. Rep., Swiss Federal Institute of Technology, Lausanne, 2001.

[13] T. Ludwig, "Voice vs. Music", Internet Sound Institute (www.soundinstitute.com), 1996.