MULTIMEDIA CONTENT'S METADATA MANAGEMENT FOR PERVASIVE ENVIRONMENTS

BY

FITSUM MESHESHA

A THESIS SUBMITTED TO

THE SCHOOL OF GRADUATE STUDIES OF ADDIS ABABA UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF SCIENCE

IN COMPUTER SCIENCE

JULY, 2005

ADDIS ABABA

Acknowledgements

First of all, I would like to thank my advisor Dr. Dawit Bekele for his constant support and his helpful suggestions and comments, which helped me a lot in preparing this thesis. I also like to thank Dr. Jean-Mark Pierson for his help in commenting my papers and forwarding useful suggestions that helped me shape my work. My special thanks also goes to Girma Berhe who has also helped me by commenting my papers, providing me with reading materials, and discussing with me on some of the problems that I encountered during the course of my work. I would also like to extend my gratitude to Dr. Mulugeta Libsie for his useful comments and advices, and for providing me with reading materials, including his Ph. D. dissertation.

I would also like to thank my parents who supported me with everything I needed in my education.

Finally, I would like to thank all of my friends and relatives for their moral support and encouragement during my study. Thank you all.

Table of contents

ACKNOWLEDGEMENTS I

TABLE OF CONTENTS II

LIST OF FIGURES IV

ABSTRACT V

1 INTRODUCTION 1

1.1 BACKGROUND 2
1.2 MOTIVATION 5
1.3 STATEMENT OF THE PROBLEM 6
1.4 OBJECTIVES 7
1.5 SCOPE 7
1.6 METHODOLOGY 7
1.7 ORGANIZATION OF THE THESIS 10

2 METADATA MANAGEMENT IN PERVASIVE SYSTEMS 11

2.1 INTRODUCTION 11
2.2 THE ROLE OF METADATA IN CONTENT ADAPTATION 12
2.3 METADATA STANDARDS 14
2.3.1 Dublin Core 14
2.3.2 MPEG-7 15
2.4 CROSSWALKS 17
2.5 METADATA MANAGEMENT IN EXISTING MODELS 19
2.6 ISSUES IN METADATA MANAGEMENT 20

3 THE ADAPTATION/DELIVERY INFRASTRUCTURE 22

4 PROPOSED ARCHITECTURE 25

4.1 OVERVIEW 25
4.2 ARCHITECTURE 25
4.3 THE METADATA MANAGER (MDM) 27
4.3.1 The Local Proxy Interface (LPI) 29
4.3.2 The Content Proxy Interface (CPI) 32
4.3.3 The Content Server Interface (CSI) 32
4.3.4 The Feature Extraction Interface (FEI) 34
4.3.5 The Crosswalk Interface (CWI) 36
4.4 THE METADATA REPOSITORY (MDR) 37
4.5 MODELING ADAPTATION CONSTRAINTS IN METADATA 40

5 IMPLEMENTATION AND RESULTS 48

5.1 OVERVIEW 48
5.2 IMPLEMENTATION 49
5.2.1 Setting 50
5.2.2 Metadata storage 50
5.2.3 Implemented features 53

5.3 RESULTS..56
6 CONCLUSION AND FUTURE WORK..59
REFERENCE..62

List of figures

Figure 1 An example of metadata in Dublin Core standard, adapted from [11] 15

Figure 2 A description of a video in MPEG-7 standard, adapted from [18] 17

Figure 3 Service-based distributed content adaptation infrastructure [29]........................ 23

Figure 4 The proposed metadata management architecture... 26

Figure 5 The Metadata Manager.. 28

Figure 6 A SOAP request message 30

Figure 7 A SOAP response message 31

Figure 8 A DTD for modeling adaptation constraints in metadata.................................... 45

Figure 9 An XML document containing the metadata of an image 52

Figure 10 The update metadata form of the UI.. 55

Figure 11 A simplified client device profile 57

Figure 12 The content search interface of the local proxy.. 57

Figure 13 A metadata search result.. 58

Abstract

One of the important aspects of a pervasive environment is the adaptation of content to suit a client's specific needs and choices such as the client's preferences, the characteristics of the client device, the characteristics of the network to which the client is currently connected, as well as other related factors. In order for the adaptation to be efficient while satisfying the client's requirements and maintaining the semantics and quality of the content, the adaptation system needs to have adequate information regarding the content to be adapted, the client's profile, the network profile and others. This thesis work addresses the issue of content metadata management in a pervasive environment in relation to content adaptation. For this purpose, a distributed architecture for the management of metadata of multimedia content is proposed. The proposed architecture consists of components for storage, retrieval, update, and removal of metadata in the system. It also provides interfaces to external components through which metadata can be accessed. In addition, restrictions on the adaptations that may be applied on the content are incorporated in the metadata. This enables the content creator to impose constraints on adaptations that may potentially cause the loss of critical information or a decrease in the quality of the adapted content.

1 Introduction

With the advent of the Internet, communication among people has advanced to a new era. Along with the advancement of communication and access to information, various applications, that take advantage of the communication infrastructure provided by the Internet, have come to existence.

Advancements in the area of electronics and device hardware technologies have resulted in smaller and more powerful computing devices at affordable prices. Hence, computing power can be made available not only in offices and laboratories but also in automobiles while driving, while walking in the park, in airplanes while traveling – virtually anywhere.

The networking technology has also improved considerably to provide us with high bandwidth communication that ranks in gigabits per second. In addition, wireless communication, that has invaded the whole world including Africa, is playing an important role in creating connectivity among the scattered computing power available in small, but powerful, computing devices.

All these technological advancements have led to ubiquitous access to computing power. Consequently, we have reached a new chapter in the history of computing where computation can be made from anywhere and at anytime; and information can be accessed and exchanged ubiquitously. There are several computing devices surrounding a single person, ranging from small handheld devices to powerful desktop machines. To make use of this ubiquitous access to computing power and provide useful services to the users is the subject of a new area in computer science called pervasive computing [1].

Pervasive computing has attracted the eyes of many researchers around the world. It is being studied and explored in many dimensions. However, as fascinating as the idea of pervasive computing is, it has many complications and challenges when it comes to the implementation [2, 3, 4]. One such challenge is making information available to a user regardless of the type of device he/she is using or the characteristics of the network he/she is situated in. For this, adapting the content may be required. And for the adaptation to be successful and efficient, it needs to have adequate information about the content to be adapted, the device/user the content is to be adapted, and the network. The information about the content is found in the content metadata. The information about the device/user is found from the client profile, and the information about the network can be found by monitoring the network.

This thesis work deals with the management of content metadata in pervasive environments, particularly in relation to content adaptation. For this purpose, the requirements for metadata management in an adaptation-based pervasive environment are studied and outlined. Based on these requirements, a distributed metadata management architecture is proposed. The proposed metadata management architecture incorporates various components that perform logical portions of the metadata management work. These components are distributed over the adaptation/delivery infrastructure while maintaining a well defined interface with the other components of the system, through which they provide transparent access to the metadata. The proposed architecture also includes modeling of restrictions on content adaptation that can be imposed by content creators, and incorporating these restrictions in the content metadata. The detail of the proposed architecture is presented in this thesis report.

1.1 Background

Pervasive computing applications enable users to access information from anywhere and using a variety of devices [5]. The devices vary from small portable devices like PDAs and palmtops to powerful personal computers. The communication system also varies from slower and intermittent wireless links to high speed wired links. As a result, the storage, processing, displaying and communication capabilities of these devices also vary greatly. Hence, content, especially multimedia content, cannot be equally accessed by all of these devices unless it is customized appropriately to suit each device. For example, an image of dimension 500x500pixels can be fully viewed using a desktop computer of 15inch monitor, but it cannot be (fully) viewed using a PDA of screen dimension 320x240pixels. Consequently, there are two ways by which content can be accessed by these diverse client devices. First, the content provider can prepare various versions of the same content corresponding to each type of device that can be used to access the content. This approach, however, may result in high overhead of maintaining the content variations as the type of devices and the amount of the content to be accessed increase. If we consider the above example, the content provider has to provide a version of the image with dimension 320x240pixels for the PDA to be able to display it. The second approach focuses on the customization of content to suit each user context (user's preference, device characteristics, network characteristics, and others) either offline or on the fly. Again going back to the example above, the image dimension could be reduced to 320x240pixels on its way to the PDA while the original image remains unchanged. The later approach, called content adaptation, is currently one of the major issues in pervasive systems [5, 6, 7]. In short, content adaptation is the customization of content so that it is presentable for a client of specific requirements and environmental settings (the client's preferences, the device capabilities and the network characteristics).

Depending on where the adaptation process takes place, content adaptation can be approached as [5] (i) *Server-based*: the content adaptation process will be incorporated into the traditional server (ii) *Client-based*: the client is responsible for either performing the content adaptation or for selecting the best representation of the content that is suitable for it (iii) *Proxy-based*: a proxy intercepts every communication between the client and the server and carries out the appropriate adaptation activities (iv) *Service-based*: the adaptation process is considered as a service that can be provided by a third party, possibly commercial, service provider.

The service-based approach is a relatively newer approach as compared to the other approaches. It involves the use of, specialized, third party services to carry out the adaptation. This enhances the performance of the adaptation process since the services are dedicated to content adaptation. In addition, this approach contributes to making the overall system more open and extensible. A service-based adaptation framework for pervasive environments, developed by Girma Berhe, is discussed in [5].

In service-based approach to adaptation, decision on the appropriate type of adaptation that should be carried out on a given multimedia content is made by a core component called the *decision engine*. In order for the decision engine to select the best adaptation for a given content, it has to have detailed information about the content to be adapted, the device to which the content is to be adapted and the network characteristics. The more the decision engine knows about the data and the target device/user, the better the adaptation result will be. The knowledge about the data can be obtained from its metadata - information that describes other data. And the information about the target device/user is a part of what is called client profile. The metadata associated with a given multimedia content should provide as much information to the decision engine as possible so that the best possible adaptation can be applied. The same is true for the device/user profile. In fact, it is one of the major research issues in pervasive environments [8, 9]. In this thesis work, however, we concentrate on the management of metadata of the multimedia content. A more detailed discussion about metadata is presented in Chapter 3.

1.2 Motivation

As the number and type of small, mobile client devices increases and pervasive systems become an important aspect of modern age computing, related issues also become more apparent. One of these issues is content adaptation. Particularly with the application of web services for the purpose of content adaptation, there arises a need for appropriate description of the inputs to the adaptation, i.e. the client environment and the content, so that the result of the adaptation better suits the client environment and the semantics of the content is preserved. To achieve this, the inputs to the adaptation service need to represent the best possible information about the content to be adapted and the client environment to which the content is to be adapted to. Hence, the content metadata and the client profile should supply as much information as possible about the content and the client environment.

Up to now, metadata has been mainly used for indexing, searching and retrieval of documents, especially in digital libraries. But the information contained in the metadata can be used for other purposes as well, though not fully exploited. In particular, metadata can be used to assist in the process of content adaptation by providing the necessary information about the content to be adapted. With the exception of few [5, 10], this option has not yet been explored. In these two works, metadata is used to assist the process of content adaptation. However in both works, the main focus is on the process of adaptation; and metadata management is not explored to its full potential. Hence, there is a need for the exploration of the potential of multimedia content metadata for the purpose of content adaptation and issues related to its management, which is the subject of this thesis work.

1.3 Statement of the problem

Pervasive computing entails content adaptation. Content adaptation, in turn, requires comprehensive knowledge of the content to be adapted, the client to whom the content is to be adapted to, and the network characteristics in which the client is in. The more the adaptation system knows about the content to be adapted, the more appropriate the decision regarding its adaptation will be. And content metadata contains information about the content it is associated with. This information, not only helps in the process of adaptation, but also in indexing, querying, retrieving and managing content. Hence, metadata forms an important aspect of pervasive computing. Consequently, its management becomes an important aspect as well.

The core problem of this thesis work is, therefore, the management of multimedia content metadata in adaptation based pervasive environments. The metadata in such an environment needs to be managed efficiently so that it can provide useful information about content. The management of metadata includes: acquisition into the system, proper storage, caching, updating, retrieval, removal, exchange among various components of the system, as well as querying of metadata.

The various components of multimedia content (text, image, audio or video) have different sensitivity to content adaptation, to the extent that adaptation may be forbidden completely. Hence, adaptation constraints need to be modeled for each media type and included in its metadata so that the appropriate precaution is taken when adapting the content. This modeling and inclusion of adaptation constraints in the content metadata also falls under the management of metadata.

1.4 Objectives

The main objective of this thesis work is to propose a metadata architecture that facilitates the management of metadata for multimedia content in a pervasive environment. The proposed architecture is expected to facilitate the management of the metadata in such a way that the necessary information about the multimedia content is represented in the metadata. In addition, access to the metadata should be facilitated so that the other components of the adaptation/delivery system can easily make use of the metadata.

The second objective of this thesis work is to develop a prototype metadata management system that will demonstrate the validity of the proposed architecture for the management of multimedia content metadata.

1.5 Scope

This thesis work targets the management of metadata for multimedia content in pervasive environments. Specifically, the management scheme is concerned in managing the metadata with the intention of supplying the necessary information about the multimedia content with respect to content adaptation. In this thesis, multimedia refers to information represented as text, image, audio or video. Hence, this thesis is primarily involved with the management of the metadata of each component of multimedia content for the purpose of content adaptation. Modeling of restrictions on adaptation in the content metadata also falls within the scope of this thesis work.

1.6 Methodology

Different methodologies are used for the various phases of this thesis work. The first phase of this work involved the study of issues and areas closely related to the thesis work in order to gain a deeper understanding of the problem and its setting. This was accomplished mainly through reading journal papers, articles, books and other reading materials that could enrich the understanding of the subject area. Major activities performed in this phase were:

- Study pervasive computing and the various issues and challenges that are raised along with it. Since the area is relatively new, it needed to be studied in greater depth so as to address the various areas of research currently underway.
- One of the major issues in pervasive environments is content adaptation and since the work in this thesis is closely related to the adaptation of multimedia content in pervasive environments, content adaptation was also one of the studied areas.
- The main concern of this thesis work is with the management of metadata for multimedia content. Hence, a comprehensive study of metadata including its representation, its nature and characteristics, its uses in various areas, the various standards, its creation, its classification and other related issues were performed.
- Study of the characteristics of each media type (text, image, audio and video) towards content adaptation was also studied so that adaptation relevant information could be included in their metadata.
- Study of other important and relevant aspects such as representation and storage possibilities for metadata.

The completion of this first phase resulted in a clear and deep understanding of the problem and its surroundings as well as the requirements for the solution to be proposed.

The second phase of this thesis work involved the development of a metadata management scheme to provide a solution to the problem. The work done in the first phase was the major input to this phase. Discussion with advisors and with colleagues in the department who work in related areas was the other important input. The major activities in the second phase involved:

- Clearly specifying the requirements for the solution to be developed. Based on the study performed in the first phase, the issues that need to be dealt with in order to come up with a reasonable solution were clearly understood and stated.
- Develop an architecture for the management of metadata that satisfies the requirements set in the previous stage. In this stage, the various components of the architecture were identified, their internal details were designed, and the means of communication among the various components of the architecture were also designed.
- As part of the proposed architecture, a modeling of adaptation constraints in the metadata of the content was developed.

The last phase of this thesis work deals with the design and development of a prototype metadata management system. The prototype of this thesis work is related to the prototype of Distributed Content Adaptation Framework (DCAF) developed at INSA, Lyon France by Girma Berhe [5] in that the prototype of DCAF uses metadata in the process of adaptation. Hence, a study of this prototype helps in outlining the requirements for the prototype of this thesis work. Consequently, a considerable amount of the work in this phase involved studying this prototype in order to understand it and identify where and how it uses metadata as well as what it lacks with respect to metadata management. The activities in this phase include:

- Study of the prototype developed for DCAF and the various related issues required to understand the prototype.
- Study of tools, technologies and protocols required for implementation of the prototype.
- Outline the requirements for the prototype.
- Develop a design to meet the requirements
- Implement the design and test it appropriately.

1.7 Organization of the thesis

This thesis document is organized as follows: Chapter 2 presents a detailed discussion on metadata, metadata representation standards, existing metadata models and other issues related to the management of metadata. Chapter 3 discusses the adaptation/delivery infrastructure on which the proposed metadata management architecture is based. Chapter 4 presents the metadata management architecture that is proposed in this thesis work. This includes the various components of the architecture and their functions as well as the communication among the various components of the metadata manager distributed at different sites. Adaptation constraints and their modeling in metadata are also discussed in this chapter. Chapter 5 discusses the implementation of the proposed metadata management architecture. Finally, a conclusion of the thesis work and suggestions for future work are outlined in Chapter 6.

2 Metadata management in pervasive systems

2.1 Introduction

Metadata is structured information that describes, explains, locates, or otherwise makes it easier to retrieve, use, or manage an information resource [11]. In short, metadata is often referred to as data about data or information about information. Such descriptive information enables us to have a good idea of the object before actually accessing it. This proves useful in situations where accessing the object is costly. For example, let the object that we want to access be a video of several megabytes, and after downloading it from the Internet we learn that it is not the video that we wanted. Here, we wasted a considerable amount of bandwidth for downloading the wrong content. But, if we first consulted the metadata of the video (either from its original location or by downloading it to our machine), we could have learnt that it is not what we wanted, with a much lesser wastage of bandwidth.

Furthermore, due to its nature, exact-match querying of audio-visual content is difficult, if not impossible. Hence, the descriptive information in the metadata can prove helpful in querying audio-visual content. For instance, textual descriptions of digital images can be included in the metadata for query/retrieval purposes so that the images can be searched based on the description found in their metadata.

Metadata has been in use in traditional libraries for a long time as a means of indexing books and other documents in the form of catalogs. The information in the catalogs is used to easily search and locate items in the library, which are properly placed in shelves based on some order. In digital libraries as well, metadata is used to facilitate indexing, searching, retrieval, and management of digital documents. The use of metadata can also be extended to museums for description of the various items found in the museum.

In general, the uses of metadata can fall in one of the following categories: resource discovery, organizing electronic resources, interoperability, digital identification, and archiving and preservation [11].

The information contained in the metadata of a multimedia content can contain features of the content ranging from low-level features such as color information of an image, to high-level features such as title and textual description. Some of these features, particularly the low-level features, could be extracted automatically from the content by using extraction software tools. However, the high-level features are difficult to automatically extract since they carry semantic meaning. Hence, human involvement is necessary in order to incorporate such features in metadata [11, 12].

Metadata may be classified from various angles. For instance, it can be classified as *content-dependent* and *content-independent* based on the dependency of the metadata on the content [13]. An example of content-dependent metadata can be the *size* of a document, while *location* can be an example of content-independent metadata. Another type of classification can be based on the purpose the metadata is used for, such as for document indexing, for resource discovery, and so on. A detailed discussion of classification of metadata for multimedia documents is given in [14].

2.2 The role of metadata in content adaptation

In this thesis work, we explore another dimension of the use of metadata, that is, its use in content adaptation. The use of metadata to support the process of content adaptation is relatively newer than the other uses of metadata mentioned above. Content adaptation relies on in-depth knowledge of the content to be adapted and the context to which the content is to be adapted to. Hence, it is up to the content metadata to provide this 'knowledge' of the content to the appropriate component which performs the adaptation.

For instance, in the ADMITS project [10], the use of metadata for content adaptation is exploited. In this project, metadata is used to support the adaptation of video content. Metadata is used to describe variations of a video content, which are results of applying adaptation operations on the video content, and the relationship among the variations and the original content. The metadata also holds adaptation hints that help in the process of adaptation.

In [5], metadata is used to facilitate the communication between clients and the local proxies (refer to Chapter 3 for details of the adaptation/delivery infrastructure). The request of the client is formulated into a metadata, which is then matched against the metadata of existing content. The local proxies may also negotiate with the client, based on metadata, to further refine the request of the client. Such use of metadata for communication avoids the unnecessary exchange of content, which is usually of a larger size than the metadata. In addition, since the metadata is rich with descriptions of the content, it can easily be used to formulate the client request because what the client provides is also a description of the content.

Metadata is also used to decide on the process of adaptation. In the service-based adaptation framework of Girma Berhe, one important component of the local proxy is the CNAM - Content Negotiation and Adaptation Module [5]. This module creates an optimal adaptation plan, i.e. it decides on the appropriate adaptation operators and their composition to be applied on a multimedia content, by making use of the information about the client (*client profile*), the network, the content (*metadata*) and others. Hence, the metadata has to provide the CNAM with as much information as possible so that the best possible adaptation, that meets the requirements of the client, can be applied on the content. The CNAM builds a *context profile* from the metadata and the client profile, and sends it to the adaptation service along with the content to be adapted so that it can be transformed based on the client's preferences.

The metadata may also include constraints on the adaptations that could be performed on the content. The constraints help preserve the semantics of the content after the adaptation. For example, reducing the dimension of a medical image below 50% may result in the loss of important information. Hence, the content provider may include a restriction in the metadata of the content specifying that the image size should not be reduced below 50%. Therefore, the adaptation service should be aware of these constraints and adapt the content accordingly. Adaptation constraints and their modeling in content metadata are discussed further in section 4.5.

2.3 Metadata standards

Standards are crucial for interoperability and having a common understanding about a certain phenomenon. This is especially true for pervasive systems where it is impossible to avoid having heterogeneous systems. Various national and international standardization bodies are established to set standards for various aspects in various fields of study, including computer science [15].

Metadata is created and used by a variety of people from various domains and different levels with in a domain. And, most importantly, in pervasive systems the metadata will be inevitably used by various types of devices and systems. Therefore, to create a common understanding, to achieve interoperability and meaningful communication among professionals and among computing systems, metadata standards are needed. The following subsections briefly discuss some of these standards.

2.3.1 Dublin Core

Dublin Core is an international metadata standard. It is developed by ISO to define a simple and concise element set that enables professionals from various domains to describe their content in an interoperable manner. The standard defines 15 core elements for describing content: Title, Creator, Subject, Description, Publisher, Contributor, Date, Type, Format, Identifier, Source, Language, Relation, Coverage, and Rights. The element set and the rules have been kept simple in order to allow people, including those who are not professionals in cataloging, to use them with little or no difficulty. All Dublin Core elements are optional, all are repeatable, and they can be presented in any order [11, 16]. An example of a Dublin Core description is sown in Figure1 below.

Figure 1 An example of metadata in Dublin Core standard, adapted from [11]

```
Title="Metadata Demystified"
Creator="Brand, Amy"
Creator="Meyers, Barbara"
Subject="metadata"
Description="Presents an overview of metadata conventions in publishing."
Publisher="NISO Press"
Publisher="The Sheridan Press"
Date="2003-07"
Type="Text"
Format="application/pdf"
Identifier="http://www.niso.org/standards/resources/Metadata_Demystified.pdf"
Language="en"
```

The Dublin Core standard is very general and domain independent. Hence, it lacks elements for describing characteristics unique to multimedia content. However, there are works underway to extend the elements of the Dublin Core so that it can be used for description of multimedia content. For instance, an extension of Dublin Core elements, through the use of sub-elements, for video content is discussed in [17].

2.3.2 MPEG-7

MPEG-7 is also an ISO standard for metadata of multimedia content. It is formally called *ISO/IEC 15938 - Multimedia Content Description Interface* [12, 18]. It defines the metadata elements, structure, and relationships that are used to describe audiovisual objects including still pictures, graphics, 3D models, music, audio, speech, video, or multimedia collections [17]. The standard specifies:

- *Description Tools* including [17, 18]:
    - *Descriptors* that define the syntax and semantics of each metadata element. They describe features, attributes, or groups of attributes of multimedia content. Descriptors are specifically defined for audiovisual content to describe features like *Structure, Color, Texture, Shape*, and *Motion*, for visual objects and features like *Musical Instrument Timbre*, *Melody Description*, and *Spoken Content Description* for audio content.
    - *Description Schemes* that specify the structure of the relationships between the elements. They are based on XML.
- A *Description Definition Language* to define the syntax of the Description Tools, to allow the creation of new Description Schemes, and to allow the extension and modification of existing ones.
- *System tools* to support the storage and delivery of descriptions, the multiplexing of descriptors with multimedia content, synchronization, file format, protection of intellectual property and others.

An example of an MPEG-7 description of a video, through the use of keywords, about presidential election in the United States of America is shown below in Figure 2.

Figure 2 A description of a video in MPEG-7 standard, adapted from [18]

```
<Mpeg7>
        <Description xsi:type="ContentEntityType">
                <MultimediaContent xsi:type="VideoType">
                        <Video>
                                <MediaLocator>
                                        <MediaUri>video.mpg</MediaUri>
                                </MediaLocator>
                                <TextAnnotation>
                                        <KeywordAnnotation xml:lang="en">
                                                <Keyword>United States of
America</Keyword>
                                                <Keyword>Presidential election</Keyword>
                                        </KeywordAnnotation>
                                </TextAnnotation>
                        </Video>
                </MultimediaContent>
        </Description>
</Mpeg7>
```

The last two sub-sections discussed two of the popular metadata standards. There are, however, a number of other standards for metadata developed for various purposes and applications. These include: Text Encoding Initiative (TEI), Metadata Encoding and Transmission Standard (METS), Metadata Object Description Schema (MODS), and Learning Object Metadata (LOM) [17]. There are also some domain specific metadata standards. For example, in the GIS world, there are several metadata standards developed by different national and international bodies. One such standard is the Content Standard for Digital Geospatial Metadata (CSDGM), which is an American initiative developed by the Federal Geographic Data Committee (FGDC) [19].

2.4 Crosswalks

The existence of various metadata standards raises a problem in presenting metadata for different applications that use different standards. Applications need to support several metadata standards in order to communicate and operate with other related applications.

However, as the number of standards increases, it will be difficult to cope up with all the possible standards. Hence, an application uses only one of the standards. But if, for various reasons, it needs to access metadata of a different, but related, standard than the one it is using, it could have the new metadata translated to the standard it supports by a software system. This software system should be able to 'understand' various standards and provide *crosswalks* that bridge the various, related, metadata standards [20].

The construction of crosswalks between standards is a difficult task and requires a detailed knowledge and experience of both standards that need to be bridged. A mapping of elements between the two related standards needs to be developed and this mapping should be appropriately implemented to facilitate automated transformation of metadata documents from one standard to a related, but different, one. For instance, the construction of crosswalks between metadata standards in the GIS domain is discussed in [19]. The paper has outlined a series of steps to be followed in developing a crosswalk between two geographic metadata standards:

- Harmonization: aims at obtaining a formal and homogeneous specification of both standards.
- Semantic mapping: aims at establishing a semantic correspondence of elements between the standards.
- Additional rules for metadata conversion to handle exceptional situations like data type conversions.
- Mapping implementation: aims at developing an automated crosswalk by applying appropriate tools.

A more general discussion of crosswalks between metadata standards and the steps involved in their development is given in [20].

2.5 Metadata management in existing models

Metadata being an important aspect of many applications, a number of efforts have been made, and are being made, to address the various issues that revolve around it, such as standardization, automatic and semi-automatic generation, representation, storage, and others. Even though metadata has been widely used for the purpose of document search and retrieval, there are some initiatives towards its use in content adaptation [5, 10]. However, in these works, the metadata management is performed as part of the adaptation process. There is no clearly designated component of the system that handles the metadata management issues. This may have some disadvantages, one of which is the absence of clear separation of role. Metadata management is one logical part of the whole content adaptation/delivery process and should be dealt with separately by a component fully dedicated to it. If the metadata management is not handled by a dedicated component, metadata will not be explored to its full potential. However, if the metadata is managed by a dedicated component of the system, the full potential of metadata in supporting the adaptation/delivery process will be explored.

Making the metadata management independent of the rest of the system is important because the various components of the adaptation/delivery system can be improved and enhanced independently without affecting one another. For instance, one metadata management scheme may be replaced with a more efficient one (keeping the interface the same) without necessitating a change on the other components. In the same manner, the component that performs the adaptation process may be improved or replaced with a better one without affecting the management of metadata. This clear separation of responsibilities also facilitates the development and maintenance of the system as a whole. Hence, in this thesis we propose a metadata management scheme which primarily targets content adaptation in pervasive environments.

2.6 Issues in metadata management

The management of metadata in a service-based adaptation based infrastructure (refer to Chapter 3 for details of the infrastructure) involves a number of issues. These include:

- Acquisition/removal of metadata to/from the system. The metadata manager should handle the insertion of metadata into the system when the content is first introduced into the system at the content servers. It should also handle removal of metadata from the system whenever the content is removed from the system in order to avoid the inconsistency of having a description (metadata) of a content that does not exist.
- The metadata management should provide appropriate access to the metadata for the other components of the system that make use of it, for example the CNAM – the component that makes adaptation decisions on a certain multimedia content. For this, the metadata manager should provide a well-defined interface through which users of the metadata can access the metadata in a transparent manner, i.e. the interface should hide the distribution and replication of the metadata as well as its underlying management.
- Updating of the metadata when there is a change on the content. If the content changes and this change affects the characteristics of the content recorded in the metadata, the metadata also have to be updated to reflect this change. This also falls under the responsibilities of the metadata management system.
- Generation of metadata for newly created content. After an adaptation has been performed on a given content, we have a new content, which is a version of the original content. This new content should also have a metadata so that it can be used to serve similar requests in the future. The metadata manager should generate the metadata for this new content by applying the necessary changes to the metadata of the original content, since for the most part the two contents have

the same information in their metadata except for the ones affected by the adaptation.

- Caching of metadata. In order to provide fast access to metadata, caching of metadata at the proxies is important. The metadata manager should handle the caching of metadata at the proxies to increase the efficiency of access to the metadata.
- Completing the metadata by extracting missing features. The metadata may not include all the necessary information required to serve client requests or to decide on the process of adaptation. In this case, the metadata manager should further enrich the metadata by employing the appropriate feature extraction module to extract the missing information.
- Metadata standard conversion. If the metadata of a newly inserted content is represented in a format that is different from the one that is used in the system, then the appropriate transformation (crosswalk) needs to be applied on the metadata in order to make it usable by the system. Hence, the metadata management should employ the appropriate crosswalk module to perform the transformation.
- Incorporating adaptation constraints in the content metadata so that the semantics of the content is preserved after adaptation.

The issues discussed above are those that are closely related to the adaptation/delivery infrastructure found in [5], and which serves as the base for this thesis work (refer to Chapter 3 for details). There are a number of other issues that are raised along with metadata management as the metadata passes through various stages in its life cycle from creation to consumption [21]. Since metadata is used in different application domains and for various purposes, the issues also vary accordingly [22].

3 The Adaptation/delivery infrastructure

This chapter discusses the adaptation/delivery infrastructure that is used to develop the proposed metadata management architecture. The infrastructure is developed by Girma Berhe at LIRIS (Laboratoire d'Informatique en Images et Systèmes d'Information) – INSA, Lyon. The overall service-based adaptation/delivery system is known as Distributed Content Adaptation Framework (DCAF) [5].

In a service-based distributed content adaptation/delivery system, which provides access to content for clients through various pervasive devices, there are a number of components in the background that carry out different activities. DCAF is such a framework with a set of components, each performing a logical portion of the whole adaptation/delivery activity [5].

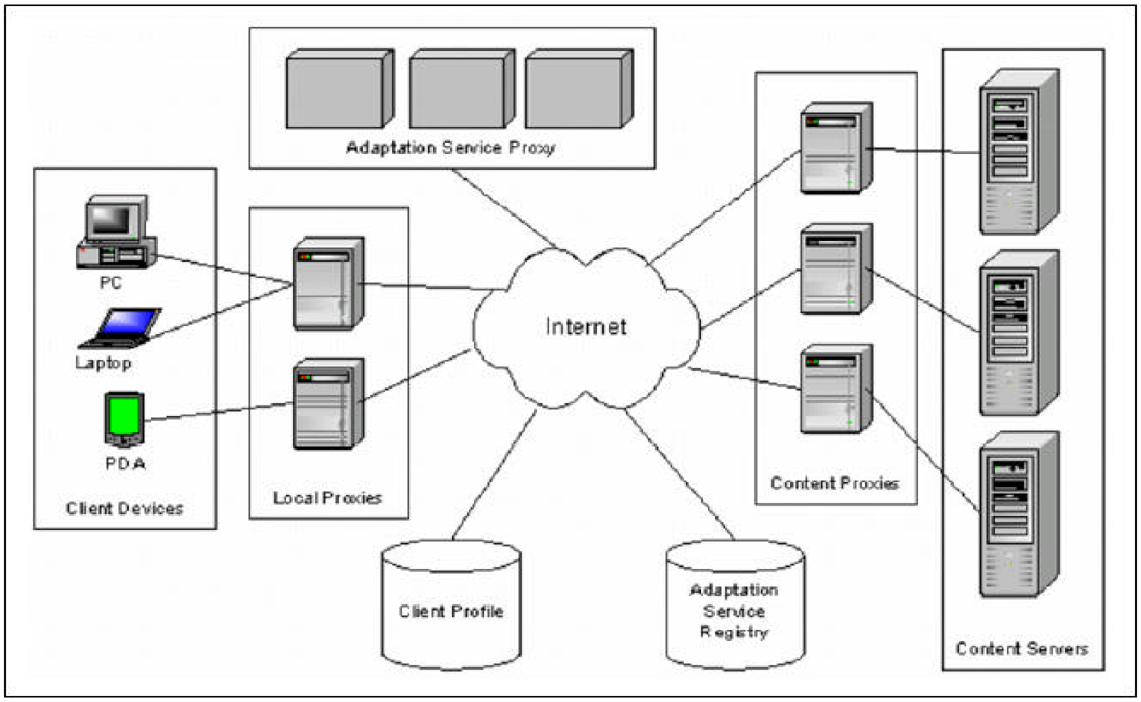
In DCAF, there are designated components to perform each of the following major activities:

- Communicating with the client to accept requests for content and providing the requested content (possibly carrying out adaptation whenever necessary and possible).
- Storing and providing access to profile information for:
    - Client
    - Content (metadata)
    - Network
    - Adaptation services
- Storing and providing access to multimedia content.

- Gathering and analyzing client profile, content profile (metadata), and network profile to determine the appropriate type of adaptation to be performed on a certain multimedia content.
- Carrying out the necessary adaptation based on the results of the analysis of the various profile information.

Since in our project the objective is to manage multimedia content metadata in such a distributed pervasive environment, we use this framework, DCAF, as a reference model to develop the proposed metadata architecture. This framework and its various components are illustrated in Figure 3 below.

Figure 3 Service-based distributed content adaptation infrastructure [29]



The various components of this infrastructure are discussed below briefly. The full description of the system and the components as well as their functions can be found from [5].

- **Local Proxy:** responsible for retrieving client requests and profiles, analyzing client profile and content metadata, planning adaptation strategy, and integrating and forwarding adaptation results to the user. In addition, caching of adapted content also falls under the responsibility of the local proxy. This helps to increase the efficiency of similar requests from users in the future. One of the major components of the local proxy is the Content Negotiation and Adaptation Module (CNAM). The CNAM is responsible for creating an optimal adaptation plan according to the *delivery context* which is generated by analyzing client, content, and network profiles [29].
- **Content Proxy:** provides access to the content server, i.e. it provides access to multimedia content and content metadata. It also performs caching of content for efficient access.
- **Content Server:** stores multimedia content and the associated content profile or content metadata.
- **Client Profile:** stores profiles of clients. These include user preferences as well as device characteristics.
- **Adaptation Service Registry:** allows the lookup of adaptation services by storing service profile such as service type, location, supported media formats etc.
- **Adaptation Service Proxy:** perform content adaptation on behalf of the user or content provider and are implemented as Web Services.

Our proposed architecture for metadata management, which is presented in the next chapter, is based on the infrastructure discussed above, making appropriate modifications and customizations as necessary and permissible.

4 Proposed architecture

To manage the metadata in a service-based adaptation/delivery framework in such a way as to answer to the issues raised in section 2.6, we propose a metadata architecture based on the service-based adaptation driven infrastructure discussed in the previous chapter. The metadata management is basically composed of components and interfaces that are distributed at the various sites and working cooperatively to efficiently manage the metadata and provide appropriate access to it.

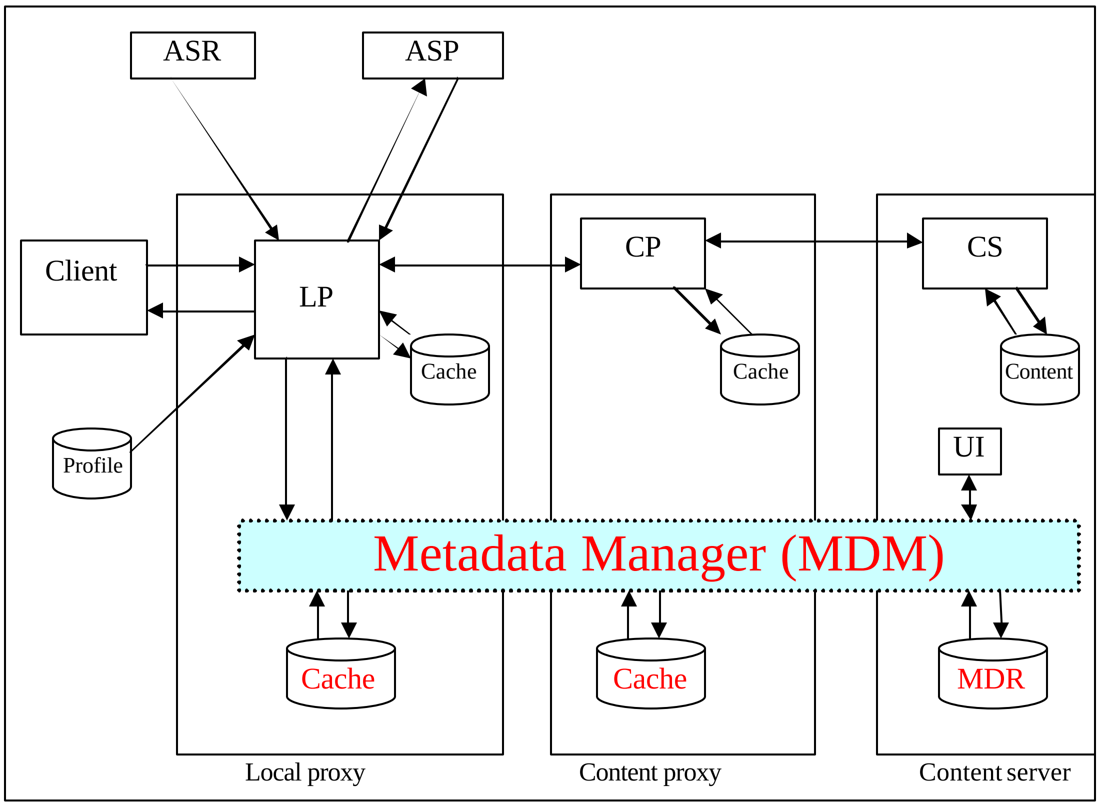
4.1 Overview

The proposed metadata management architecture is going to be distributed over the various components of the adaptation/delivery system – local proxies, content proxies, and content servers. The metadata is inserted into the system at the content servers. It is then consumed at the content proxies and local proxies; it is appropriately cached, updated, deleted or transformed at the content proxies and/or local proxies. Hence, the metadata architectur e proposed for the management of metadata should be aware of these phenomena that occur in the life time of the metadata and perform the necessary functions to properly manage and provide access to it. The architecture achieves communication among the various components of the metadata management at the different sites by means of message passing. The messages can be for exchange of request/response messages, for exchange of update messages, or for exchange of metadata.

4.2 Architecture

In this section we provide the logical architecture of the metadata management system that we propose. In the proposed system, the main work of management of metadata is performed by the *Metadata Manager (MDM).* The Metadata Manager is a set of cooperating modules that execute operations in order to manage the metadata and to provide appropriate access for whoever needs it through well-defined interfaces. The Metadata Manager is distributed at the various places where metadata is located and provides a transparent access to the metadata. These are the *content server* where content and metadata are originally inserted and stored permanently, the *content proxy* where content and metadata are cached to facilitate access to the content servers and to better serve similar requests efficiently, and the *local proxy* where adapted content and metadata may also be cached to increase the efficiency of similar future requests. Hence, the Metadata Manager consists of various modules with different functionalities as well as different interfaces, distributed over the system, while providing a transparent access to the metadata for the components of the system that need to access and make use of the metadata, such as the Content Negotiation and Adaptation Module (CNAM) [5]. The proposed architecture is shown in the Figure 4 below.

Figure 4 The proposed metadata management architecture



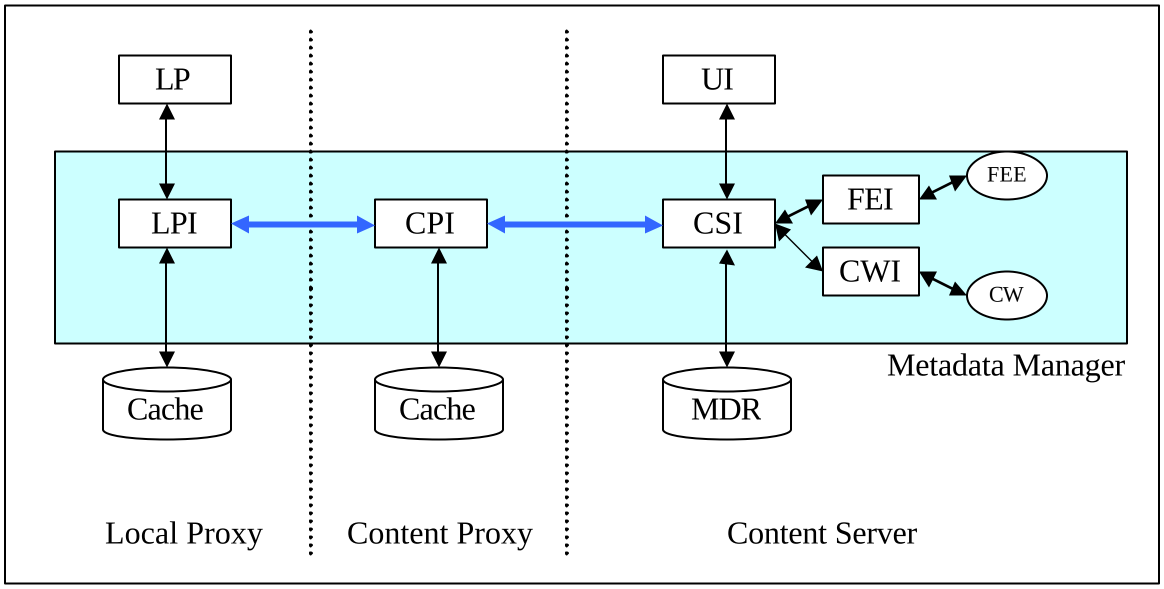
In this system, the metadata is stored in the Metadata Repository (MDR) located at the content server. In addition, for efficiency purposes, metadata is kept in caches at the content proxy and at the local proxy. On top of these storage components lies the Metadata Manager (MDM). The MDM is responsible for managing the metadata stored througho ut the system (i.e. at the caches and at the MDR) and for providing an interface for accessing the metadata so that the other components of the system can access and use it. To achieve this, the MDM has different interfaces to external components depending on the location and nature of the components. Furthermore, the MDM has other internal modules that perform additional functions for the efficient management of metadata.

Among the requirements for metadata management are the issues of inserting, deleting and updating metadata. For this purposes, we have introduced a *User Interface (UI)* component in the architecture. The UI component is located at the content server and allows users (potentially metadata experts) to insert new metadata into the system, remove unwanted metadata from the system, or update existing metadata in the system. The UI communicates with the portion of the MDM at the content server to achieve the insertion, deletion, and update of metadata.

4.3 The Metadata Manager (MDM)

The Metadata Mana ger has a number of modules inside it that perform a certain logical portion of the overall metadata management task, as shown in Figure 5 below. These are the *Local Proxy Interface (LPI)* which is responsible for the portion of the metadata management at the local proxy and providing access to the metadata for the local proxy, the *Content Proxy Interface (CPI)* which is responsible for management of metadata at the content proxy and serves as a bridge between the local proxy and content proxy, the *Content Server Interface (CSI)* which is responsible for the management of metadata at the content server and providing an interface for insertion, deletion, update of metadata, the *Feature Extraction Interface (FEI)* which is responsible for completing the metadata by extracting features from the content, and the *Crosswalk Interface (CWI)* which is responsible for transforming metadata from one standard to another so that it can be used in the system.

Figure 5 The Metadata Manager



In this architecture, the communication between the MDM and the other components of the system will be conducted through the well-defined interfaces that are provided by the MDM itself. The communication among the three fragments of the MDM, located at the local proxy, content proxy and content server, is done by means of message passing using the Simple Object Access Protocol (SOAP) [26, 27]. SOAP is an XML-based lightweight protocol designed to work with existing Internet and XML open standards [28]. The fact that SOAP is XML based makes it an ideal choice in our case because the metadata that is going to be exchanged is also in XML. Hence, the metadata can be easily transferred among the three fragments of the MDM as a SOAP response message.

4.3.1 The Local Proxy Interface (LPI)

This component of the MDM is responsible for management aspects of the metadata at the local proxy. This involves providing an interface for the responsible component of the local proxy that wants to access the metadata, for instance the CNAM [5]. In addition, the LPI is responsible for caching metadata at the local proxy to increase the efficiency of metadata access. For requests of metadata that cannot be satisfied from the cache at the local proxy, the MDM is responsible for retrieving the required metadata from the neighboring cache, i.e. from the cache at the content proxy. This is achieved by having the LPI forward the request to the Content Proxy Interface (CPI) which is located at the content proxy.

Consequently, the LPI has to provide an interface to external components of the system for retrieval of metadata. This is basically presented in the form of a query against the metadata. The result of this query can be one or more metadata document(s) that satisfy the query or NULL if no metadata that matches the query is found. Hence the interface will have a general form of:

getMetadata (*query_parameters*) : metadata[]

where

- ***query_parameters*** is a list of values that can be found in the metadata. For instance, a query can be made using *title* and *keywords* as query parameters. The actual number and type of query parameters, however, depends on the level of precision required and other factors that are implementation specific. For example, to retrieve metadata of **text** documents written in **English** language, we can invoke getMetadata() as follows**: getMetadata(***query_parameters***)** where *query_parameters* = {“English”, “text”}.
- **metadata[]** is a, possibly empty, set of metadata documents that are returned as a result of the query.

To reply to a ***getMetadata()*** query, the LPI performs a query against the cache at the local proxy. If this query returns NULL, the LPI has to forward the query to the CPI through the underlying network. This is achieved by composing a SOAP request message and sending it to the CPI. The request is simply the ***getMetadata()*** query, received from an external component of the system, wrapped in a SOAP envelope. Figure 6 below shows a sample SOAP request message that sends a query based on a *keyword* parameter.

Figure 6 A SOAP request message

```
<SOAP-ENV:Envelope  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
          xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
          xmlns:xsd="http://www.w3.org/1999/XMLSchema">
   <SOAP-ENV:Body>
          <mdm:getMetadata xmlns:mdm="MetadataManager"
                    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
             <keyword xsi:type="xsd:string">lung</keyword>
          </mdm:getMetadata>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The component of the MDM that receives this message, in this case the CPI, extracts the parameter from the message and performs a local ***getMetadata()*** query against its cache. If this local query returns metadata, it wraps it with a SOAP envelope and sends it to the LPI as a SOAP response message. If, however, the CPI cannot find a matching metadata in its cache, it in turn forwards the request to the SCI as a SOAP request message. Then it forwards the response it receives to the LPI as a SOAP response. Figure 7 below shows such a response message, the body of which is a metadata document.

Figure 7 A SOAP response message

```
<SOAP-ENV:Envelope  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
            xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
            xmlns:xsd="http://www.w3.org/1999/XMLSchema">
    <SOAP-ENV:Body>
            <mdm:getMetadataResponse xmlns:mdm="MetadataManager"
                        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
                <mediaProfile>
                        <title>Lung Cancer X-ray</title>
                        <creator>ABC Hospital</creator>
                        <description>An X-ray image showing a lung cancer.</description>
                        <keywords>lung, cancer, x-ray</keywords>
                        <catagory>medical</catagory>
                        <date>10/10/2005</date>
                        <type>image</type>
                        <language>en</language>
                        <uri>http://contentserver/data/lungXRay.gif</uri>
                        <details>
                                    <format>gif</format>
                                    <size>2048</size>
                                    <colorDepth>16</colorDepth>
                                    <dimension>
                                                <height>10</height>
                                                <width>5</width>
                                    </dimension>
                        </details>
                        <constraints>
                                    <reduce_color>
                                                 min_color="8"/>
                                    </reduce_color>
                        </constraints>
                  </mediaProfile>
            </mdm:getMetadataResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

After receiving the SOAP response message, the LPI extracts the body of the message (i.e. the metadata) and returns it to the component that first initiated the ***getMetadata()*** call as a set of metadata documents.

4.3.2 The Content Proxy Interface (CPI)

This component is responsible for caching of metadata that are previously retrieved from the MDR so that requests for metadata from the LPI are satisfied without going to the MDR. The CPI accepts requests for metadata from the LPI. If it finds the requested metadata in its cache, then it returns it to the LPI. If the required metadata is not found in its cache, the request is forwarded to the MDR at the content server. When the CPI gets the metadata from the MDR, it in turn forwards it back to the LPI. The CPI does not interact with other modules that are not part of the metadata manager, i.e. it only communicates with the portion of MDM at the local proxy (LPI) and the portion of the MDM at the content server (CSI). And both communications are made through the underlying network infrastructure (refer to Figure 5 above).

The LPI – CPI and the CPI – CSI communications are all done using SOAP. The request and response messages take the same form as the ones discussed for the LPI in section 4.3.2 above and illustrated in Figure 6 and Figure 7, respectively.

4.3.3 The Content Server Interface (CSI)

This component is responsible for managing the metadata in the MDR located at the content sever, where the metadata is permanently stored. The CSI is responsible for providing an interface to the portion of the MDM at the content proxy, i.e. the CPI. In addition, it communicates with the User Interface (UI) component so that users can insert, delete, and/or update metadata. Moreover, the CSI is responsible for providing an interface to two other components of the MDM: the Feature Extraction Interface (FEI) and the Crosswalk Interface (CWI). These are responsible for invoking Feature

Extraction Engines (FEE) and Crosswalks (CW), respectively. The FEEs extract features from multimedia content that are missing in the corresponding content metadata, while the CWs transform metadata from one standard to another. FEI and CWI are discussed further in sections 4.3.4 and 4.3.5.

The CSI provides different interfaces to different components of the MDM as well as external components of the architecture. First, it has to communicate with the CPI through a SOAP interface in order to accept SOAP request messages and to send metadata documents as SOAP reply messages. The message structure and processing is the same with that of the LPI discussed in section 4.3.2.

Second, it has to provide interfaces for insertion, deletion, and update of metadata as per the request of the user through the UI. For this purpose the CSI provides the following interfaces:

- For insertion of metadata into the system:

  insertMetadata (*metadata, error_detail*) : *status*

  where,

  metadata is the metadata document, in XML, to be inserted to the system and ***status*** is a boolean flag returned to indicate the status of the operation. If successful, it will have a value of *True*; otherwise it will have a value of *False.* The operation may be unsuccessful for various reasons. For example, the metadata may not be a well-formed XML document. In this case, the operation will fail while trying to parse the document; hence the metadata will not be inserted to the MDR. This is indicated by returning *False* in the **status** flag. Whenever the **status** flag returns *False*, the caller can get details of the error that has occurred from the ***error_detail*** parameter, which gives a description of the error.
- For removal of metadata from the system:

removeMetadata(*metadata_identifier, error_detail*) : *status*

where,

metadata_identifier is a unique identifier of the metadata document to be removed from the MDR and ***status*** and ***error_detail*** have the same meaning and function as for **insertMetadata()** discussed above.

- For update of metadata:

  updateMetadata(*metadata_identifier, attribute_list, value_list, error_detail*) : *status*

  where,

  metadata_identifier is a unique identifier of the metadata document to be updated, ***attribute_list*** is an array of attributes (such as *title*, *keyword*, etc) of the multimedia content that are to be updated and ***value_list*** holds the corresponding values of the attributes in the ***attribute_list***; and ***status*** and ***error_detail*** have the same meaning and function as for **insertMetadata()** discussed above.

Third, the CSI has to communicate with the Feature Extraction Interface (FEI) and Crosswalk Interface (CWI) of the MDM. Consequently, it needs to appropriately invoke the interfaces provided by these components. These two components and their interfaces are discussed in sections 4.3.4 and 4.3.5 below.

4.3.4 The Feature Extraction Interface (FEI)

This module is responsible for calling the appropriate *Feature Extraction Engine (FEE)* depending on the needs of the adaptation process. The feature extraction is required to enrich the metadata by including features that are not originally present but are required to fulfill some task. At first, the metadata can be composed of basic and available features of the content, and it will be augmented later with additional features, in the course of operation of the system, by extracting them from the content. The actual implementation of the various feature extraction engines, however, is outside the scope of this thesis work. In our architecture, we only provide an interface to invoke the appropriate extraction engine depending on the requirements of the system and consume the features that are extracted by incorporating them to the appropriate metadata.

Currently, there are several feature extraction tools that extract features from a multimedia content automatically [23, 24, 25]. The extraction tools may extract low-level (physical) features as well as features that are of higher semantics [24]. For example, low-level features of image content include width, height, color depth, etc while high-level features include keywords, textual description, title, etc. The extraction process can be either fully automated for low-level features that can be directly extracted from the content based on some algorithm. But it can also be semi-automatic [25] for those features that need a higher level of understanding and semantics. In both cases, the number and type of feature extractors keeps on augmenting due to the increase in the level of expertise, the availability of more powerful tools and techniques, the requirements of the applications that use the features, and other reasons. Hence, our architecture cannot possibly incorporate all the possible feature extraction engines. It should not restrict itself to a few of the feature extraction engines currently available either. The architecture should be open to incorporate new feature extraction tools by providing a well-defined interface through which they can be invoked and their results used.

The FEI has to provide an interface through which the CSI can request for feature extraction. For this purpose, the FEI internally maintains a list of FEEs, the corresponding features that are extracted by these FEEs, and the type of multimedia content they extract features from. The list takes the following form: (FEE, content_type, feature_list). Whenever a new FEE is introduced into the system, the FEI adds the FEE along with the features it extracts to the internal list. The interface provided by the FEI for the CSI takes the following form:

extractFeature(*content, content_type, feature_name_list, feature_value_list, error_detail*) : *status*

where,

content is the multimedia content from which features are to be extracted, ***content_type*** is the type of the multimedia content (text, image, audio or video), ***feature_name_list*** is the list of features to be extracted from the content, ***feature_value_list*** is the value of the features that are extracted from the content (it is empty when the call is made; and may be populated or empty after the call returns), and ***status*** and ***error_detail*** have the same meaning and function as for **insertMetadata()** discussed above.

The FEI consults its internal list of FEE – feature_list pairs to determine which FEE(s) to invoke in order to reply to an ***extractFeature()*** call. There may not be an FEE that can extract all the requested features in the ***feaure_name_list***. Hence, the FEI may need to invoke several FEEs to carry out one ***extractFeature()*** call. It is also possible that some or all of the features in the ***feaure_name_list*** may not have corresponding FEEs that can carry out the extraction. In this case, the ***extractFeature()*** call fails, and the details of the failure are returned in ***error_detail***.

4.3.5 The Crosswalk Interface (CWI)

This component of the MDM is responsible for transforming metadata from one standard to another standard. This is important because the system employs a certain metadata standard (MPEG-7, for example) but the metadata of a content can be inserted into the system in a format different from that used by the system (for example Dublin Core). This can be because of various reasons. For instance, the content provider that supplies the metadata may use a different metadata standard than that used by the system. In such cases, the metadata manager should convert the metadata to the standard that is used by the system. For this purpose, the CWI invokes the appropriate *crosswalk (CW)*, that transforms the new metadata from its standard to the standard used by the system, and inserts the resulting metadata (which is in a format usable by our system) into the Metadata Repository.

As there are several standards for metadata, there are (and will be) several crosswalks that implement the transformation of metadata from on standard to another. As in the case of the feature extraction engines, the implementation of the various crosswalks is outside the scope of this work. It should be performed by professionals who have a deep knowledge and experience in the various metadata standards [20]. Hence, the architecture is kept open in this regard by providing well-defined interfaces to invoke the appropriate crosswalk and make use of the result.

The CWI maintains a list of all the Crosswalks (CW) that it has under it. The list takes the form: (CW, from_standard, to_standard) and it keeps on growing as new CWs are introduced into the system. This list is consulted when a request for a standard conversion is received from the CSI. For the CSI to be able to forward such requests, the CWI provides the following interface:

convert_standard(*metadata, from_standard, to_standard, error_detail*) : *status*

where,

metadata is the metadata document that is to be converted to a new standard, ***from_standard*** is the current metadata standard of ***metadata***, ***to_standard*** is the target metadata standard to which ***metadata*** is to be transformed, and ***status*** and ***error_detail*** have the same meaning and function as for **insertMetadata()** discussed above

4.4 The Metadata Repository (MDR)

The Metadata Repository is where the metadata is stored. The metadata is going to be represented in XML. The metadata document may contain any relevant information about the multimedia content. In addition, the metadata may also include adaptation constraints that need to be enforced when the content is adapted (see section 4.5 below). Hence, the MDR is responsible for storage of metadata encoded in XML. Each metadata document is stored in the MDR and properly indexed so that it can be searched for and retrieved efficiently. This is achieved by using a database to store the metadata documents along with a unique identifier assigned to each.

In our architecture, the content of the metadata is categorized into three parts. The first part contains general information that can be used for querying purposes. These include properties such as *title, content type, keywords, description, date of creation, creator,* etc. The second part contains technical information that is used in the process of adaptation. Such information is different for the different media types. For example file format and file size can be sited as common attributes that fall in this group. Color information of images and videos, bit rates of audios, frame rates of videos are also examples of content characteristics that fall in this second category. The third part contains constraints on adaptations that may be applied on the content. These are information that need to be taken into consideration when adapting the content (see section 4.5).

This classification of the content in the metadata facilitates the storage of metadata documents in databases tables. With this approach, each of the attributes in the first category can be considered as columns in the table while the attributes in the second category are stored under one column. The attributes in the third column too are stored under a single column. This simplifies the storage of the metadata in existing relational tables and facilitates querying for metadata documents using the SQL language through existing database programming interfaces. When a query for a metadata document is performed, the result of the query is a metadata document (in XML format) composed of the attributes from the three categories. Hence, the classification of the contents of the metadata content is hidden from the other components of the system. As far as the other components are concerned, the metadata is inserted and retrieved as a complete XML document. This technique is closely related with technical implementation details and is discussed further in Chapter 5.

Hence, when a new metadata is inserted into the system, the CSI parses the XML document and identifies the three categories of the content of the metadata discussed above and inserts them to the database.

In the proposed architecture, metadata is first introduced into the system at the content server when new content is added to the system. Then, it is sent to (and cached at) the content proxy which then sends it to the local proxy where it is also cached. There it is used in the process of adaptation. After the adaptation, the metadata for the new content is generated by the MDM at the local proxy and the new metadata is cached at the local proxy metadata cache for future requests.

When metadata is no more needed, for example when content is removed from the content server or a variant of a multimedia content is removed from the caches at the local proxy, the MDM will remove the metadata from the MDR and/or the caches. For instance, if a multimedia content is removed from the content server, then the Content Server Interface (CSI) will remove the associated metadata from the MDR and it will inform the Content Proxy Interface (CPI) to remove any cached metadata of the removed content from its cache. The CPI in turn informs the Local Proxy Interface (LPI) to remove cached metadata of the removed content from its cache. This avoids the presence of metadata that describe a content that does not exist in the system.

When content is updated at the content server through the User Interface (UI), the CSI updates the corresponding metadata at the MDR and informs the CPI to remove cached metadata of the updated content from its cache. The CPI, in turn, informs the LPI to remove any cached metadata from the local proxy cache. This avoids potential inconsistencies that may occur due to the presence of different metadata for a single content. Since multimedia content is not likely to be updated very frequently, this does not impose a serious drawback on the system performance. As another alternative, the updates may be transmitted to the caches so that any cached metadata is updated accordingly. This ensures that caches are still available and are up-to-date. However, if the update messages are large, the cost in terms of bandwidth may be high. Furthermore, additional processing of metadata is required at the proxies to update the document in the cache. Hence, the former approach is employed in the implementation of our architecture for experimental purposes.

4.5 Modeling adaptation constraints in metadata

Metadata for multimedia content contains a wide variety of information about the characteristics of the document. Such information is useful when making decision on the adaptation of the document based on a user's criteria. In this thesis work, we consider the following common characteristics of each media type (text, image, audio, and video) to be included in the associated metadata, as shown in Table 1. But it should be understood that the metadata of multimedia content can contain far more characteristics than these depending on its applications [17]. The number and type of features also increases from time to time depending on various factors such as requirements of applications, availability of extraction tools, etc. Furthermore, depending on the domain of the application that is using the metadata, far more domain specific features of the content may be included in the metadata.

Our proposed architecture, however, is independent of the features included in the metadata. It should manage the metadata regardless of the type of features contained in it, or the type of standard the metadata is represented in. It is capable of incorporating new features into the metadata by using feature extraction tools as discussed in Chapter 4. The list in Table 1 is used for experimental purposes in such a way that the most common features of the different media types are considered.

Table 1 Properties of multimedia content included in metadata

<u>Text</u> • Title • Creator • Date • Format (doc, pdf …) • Size • Language (English, French …) • URI • Description • Abstract • Keywords • Category	<u>Image</u> • Title • Creator • Date • Format (gif, jpeg …) • Size • Dimension (h X w) • Color depth • Description • Keywords • URI • Category
<u>Audio</u> • Title • Creator • Date • Format (mp3, wav …) • Duration • Size • Language • Genre • Bit rate • Category • Description • Keywords • URI	<u>Video</u> • Title • Creator • Date • Format (mpg …) • Duration • Size • Language • Color • Frame rate • Dimension (h X w) • Category (movie, news clip …) • Description • Keywords • URI

As can be observed from the Table 1, the metadata can hold vital information about the content from low-level features as bit rate and color depth to high-level features as keywords and description.

The main concern of this thesis is in the usage of metadata for content adaptation. That is, the metadata should be managed in such a way that the appropriate decision, regarding the adaptation of the content, can be made. In this aspect, the metadata provides the necessary information about the content. In addition, it can also contain information regarding the types of adaptation that could and could not be applied on the content. Such information can be incorporated by the owner/creator of the content into the metadata so that the necessary information is not lost during adaptation, i.e. the semantics of the content is preserved after the adaptation.

In particular, such information is critical in sensitive domains, such as the medical domain, since the multimedia content can be used for diagnosis purposes. For example, a description about a patient's medical history may be written in English. In order to be used by a physician who does not speak English, it needs to be translated to a language he/she can understand. But since the automatic language translation services currently available are not very reliable, the creator of the content may include a "don't translate" restriction in the metadata of the document. Hence, the physician who doesn't speak English needs to have the document translated by an appropriate human translator. Hence, all information the multimedia content contains should have an acceptable quality, even after adaptation

The example above is an extreme case in which a certain type of adaptation (language translation) is completely forbidden on a multimedia content. But there can be less restrictive constraints that can be included in the metadata by the creator. For example let's consider a medical image with color depth of 24 bits. The image may contain information that is visible in high color depth values only. Hence, the creator may include a restriction such as "don't reduce color depth below 8 bits" in the image metadata. In this case, a color-reduction adaptation is possible but it should not reduce the color depth of the image below 8 bits, otherwise it may result in an improper diagnosis.

Another example of such a restriction can be seen regarding the dimension of the image. A "don't resize below 40% of the original size" constraint may be included in the metadata to prevent the reduction of the image dimension below a reasonable size which makes the image unusable due to loss of critical information. For instance, in an image indicating a cancer, the part of the image that actually shows a sign of cancer may be a few pixels wide. Hence reducing the size of the image below a certain threshold may result in the destruction of these few, but important, pixels.

In addition to information loss, quality decrease is also another reason for including constraints on adaptation. For example, reducing the bit-rate of an audio content may reduce its size so that it can be transmitted over a weak link or be played on a device with smaller capability. But this reduction of bit-rate may damage the quality of the audio. Hence, the creator of the audio content may set restrictions on this type of adaptation as well.

In addition, the creator of the content may want the content to be adapted by a specific adaptation service. For this, he/she may include a constraint that indicates the use of a specific adaptation service for a certain type of adaptation. For instance, if service X is popular for language translation adaptation, the creator of a text content may include a constraint in the text metadata saying that the text should be translated by service X. The reason for selecting a specific service may not only be popularity. It could be due to security, financial, legal or other reasons. However, regardless of the reasons, the creator of the content should be able to include such type of constraints in the content metadata.

The types of constraints that can be imposed on each media type are different as there are different types of adaptations that can be performed on each type of media. Some of the adaptations may result in loss of information or in decrease of quality while others may not have negative effects. Examples of content adaptations that can possibly result in loss of information or in quality decrease include: *summarization* and *language translation* for text content; *scaling* and *color reduction* for image content; *bit-rate reduction* and *stereo-to-mono reduction* for audio content; and *color reduction* and *spatial size reduction* for video content.

The modeling of adaptation constraints in the metadata can be precisely described using XML syntax so that it can be used by humans as well as machines. The constraint is incorporated into the metadata by humans who created the content and it is used by software that decides on the process of adaptation of the content. Hence, they need to be expressed in a format understandable by both, XML. Since the content metadata is also expressed in XML, incorporating the constraints into the metadata will not require much effort.

For this purpose, the following scheme is used. Figure 8 below shows a part of the metadata DTD for a multimedia content that concerns adaptation constraints. The DTD is defined based on the restrictions regarding a limited number of adaptation types on the four media. Namely, for text media: Text summary and Language translation; for image: Resizing and Color depth reduction; for audio: Bit rate reduction; for video: Resizing and Color reduction. This model does not exhaustively cover all the possible adaptation types, instead it considers a few, but common adaptation types on the four media types. However, the technique can be used to model constraints for a number of adaptation types that exist currently or that may come to existence in the future.

Figure 8 A DTD for modeling adaptation constraints in metadata

```
<!ELEMENT constraint (prefered_service*, ((reduce_bitrate?, stereo-to-mono?) |
(resize?, reduce_color?) | (translate?, summarize?)))>
<!ELEMENT prefered_service (service)>
<!ATTLIST prefered_service adaptation_type CDATA #REQUIRED>
<!ELEMENT service EMPTY>
<!ATTLIST service uri CDATA #REQUIRED>
<!ELEMENT resize EMPTY>
<!ATTLIST resize min_size CDATA #REQUIRED>
<!ELEMENT reduce_color EMPTY>
<!ATTLIST reduce_color min_color CDATA #REQUIRED>
<!ELEMENT reduce_bitrate EMPTY>
<!ATTLIST reduce_bitrate min_bitrate CDATA #REQUIRED>
<!ELEMENT stereo-to-mono EMPTY>
<!ATTLIST stereo-to-mono allow (yes|no) #REQUIRED>
<!ELEMENT translate EMPTY>
<!ATTLIST translate allow (yes|no) #REQUIRED>
<!ELEMENT summarize EMPTY>
<!ATTLIST summarize allow (yes|no) #REQUIRED>
```

As can be seen from Figure 8, most of the elements in the DTD are empty elements with one attribute. For most of the constraints, this attribute is a threshold, which indicates the minimum value that specific property of the multimedia content can have. The other types of attributes have a *yes* or *no* value. For example, for a *language translation* adaptation on a text document, we can either forbid or allow the adaptation. Due to the nature of the adaptation, we cannot specify a minimum value below which the adaptation is not allowed, at least currently. However, if sometime in the future, it is possible to reliably measure the accuracy of the translation service, we may specify a minimum level of accuracy above which the text can be translated.

Based on this DTD, we can formally express constraints on adaptations in the metadata of the content. For example, for an image that should not be resized below 60% of its current size, we can have the following in the image metadata:

```
<constraint>
        <resize min_size = “60”/>
</constraint>
```

As another example, for a video that could be resized to 50% and its color cannot be reduced below 24bits; we can have the following in the video metadata:

```
<constraint>
        <resize min_size = “50”/>
        <reduce_color min_color = “24”/>
</constraint>
```

For a text document that should not be translated, we can have:

```
<constraint>
        <translate allow = “no”/>
</constraint>
```

For an audio content whose bit rate should not be reduced below 64kbps, we can have the following restriction in the audio metadata:

```
<constraint>
        <reduce_bitrate min_bitrate = “64”/>
</constraint>
```

To specify an adaptation service for certain types of adaptations for a text content, we can have the following constraint:

```
<constraint>
        <prefered_service adaptation_type="translation">
                <service uri="http://www.services.com.et/translator"/>
        </prefered_service>
        <prefered_service adaptation_type="summary">
                <service uri="http://www.services.com.et/summarizer"/>
        </prefered_service>
</constraint>
```

Once, the necessary constraints are included in the content metadata, it is the responsibility of the local proxy to make sure that they are obeyed. It is the local proxy which is responsible for analyzing the client profile and content metadata in order to decide on the appropriate type of adaptation to be carried out. If in this process the local proxy finds out that a certain type of adaptation is not allowed because of restrictions imposed by the creator, it should prevent the adaptation from being carried out. But instead it could suggest other possibilities of accessing the requested content. As an alternative, it can suggest other variations of the content in different modalities, for example if it is an image that is originally requested and cannot be viewed due to constraints on the adaptation, the local proxy may suggest a textual description of the image as an alternative.

5 Implementation and results

In this thesis work, we have proposed an architecture for the management of content metadata. The detail of this architecture was discussed in the previous chapter. This chapter discusses the implementation of the proposed architecture.

5.1 Overview

The prototype system developed in this work particularly aims at implementing the proposed metadata architecture in order to demonstrate its role in supporting the adaptation and delivery of content to clients. In addition, other components of the adaptation/delivery framework were also implemented to create an environment in which the metadata management can work.

The prototype system allows users to search for content based on various criteria and then view the content(s) returned from the search. For this purpose, it provides an interface for the user through which he/she can login and select his/her device profile. After the user is logged in, he/she will be presented with a search interface that enables searching for content. The result of the search is a list of metadata that match the search criteria. The user can then select to view one of the contents based on the information in the metadata. When the user selects one of the search results, the system analyzes the characteristics of the user's device (from the device profile the user selected at login) and the metadata of the selected content to see whether the device can display the content or not. If the device can display the content in its current state, the content will be retrieved, based on the location identifier found in the metadata, and displayed for the user. If the device cannot display the content, the appropriate content adapter is located and the content is adapted before being displayed for the user. However, if an appropriate adapter is not available or restrictions in the metadata do not allow the required adaptation, an appropriate message is displayed to the user.

5.2 Implementation

The implementation involved developing the various components of the Metadata Manager (MDM) and other supporting components which are not part of the MDM but are required for the metadata management to operate properly.

The MDM is implemented as a web-based application that is invoked using standard web-based interfaces. The components of the MDM are implemented as servers that bind to a certain port address and listen for incoming messages/requests. The servers communicate with each other using SOAP messages; and they communicate with external components using standard interfaces. For instance, they accept GET and POST requests from browsers and send back appropriate responses to the requests.

All major components of the Metadata Manager (MDM) are implemented, with the exception of the Crosswalk Interface (CWI). The components of the architecture that are implemented include the Local Proxy Interface (LPI), Content Proxy Interface (CPI), Content Server Interface (CSI), Feature Extraction Interface (FEI), Metadata Repository (MDR), the caches at the proxies, the User Interface (UI), as well as two Feature Extraction Engines (FEEs). The Crosswalk Interface (CWI) is not implemented due to time constraints, since it requires familiarity with various metadata standards.

To demonstrate the usage of the metadata management, it was necessary to partially implement some of the components of the adaptation/delivery framework (refer to Chapter 3 for details of the framework) that are not part of the metadata management system. These include: (i) the local proxy that decides whether adaptation is necessary or not by evaluating the client profile and the content metadata, (ii) content adaptors (iii) client information (profile) and tracking user session. These components are implemented for experimental purpose and do not provide full functionality. They, however, serve the purpose of establishing the environment in which the metadata management system can operate.

5.2.1 Setting

The implementation is carried out on Microsoft Windows platform (Windows XP and Windows 2000). The Java programming language (version 1.4.2) is used for the coding. MySQL database (version 4.1.1) is used to implement the MDR and the caches at the proxies; and for storing user login information. The three interfaces of the MDM, i.e. LPI, CPI, and CSI, are implemented as Java Servlets using the Java Servlet technology. Apache Tomcat (version 5.0), which provides an implementation for the Java Servlet specification, is used as web-server. Message based communication among the different components of the MDM is implemented using the SOAP protocol (version 1.1), over HTTP, through the APIs provided by the Java programming language for SOAP and XML processing.

5.2.2 Metadata storage

Metadata is stored in the MDR, which is implemented using a relational database table. The database server that is used in the impleme ntation of the MDR is MySQL database server.

As discussed in section 4.4, the content of the metadata are classified into three groups: (i) those that can be used for query and are common to all media types, (ii) details that are media type specific and are used for adaptation , and (iii) constraints on adaptation. For experimental purposes, we consider the following attributes to be included in the first group: *title, creator, date, description, keywords, category, media type, language, and* *identifier*. These attributes can be used by clients for querying metadata, and hence content. Therefore, each of them is represented as a separate column in a relational table to facilitate querying through the SQL language.

The attributes in the second group vary for each media type. And even the same media type, the list is unpredictable. It depends on the type of feature extractors available and/or the level of detail required by the system. For instance, for image media, we can have *width*, *height*, *format*, *color depth*, etc.; for audio we can have *bit rate*, *format*, etc. Therefore, all the attributes in this group are stored in a single column (of type 'text') in the database as a fragment of an XML document. For example, for image content, the following is stored in the database table as a single string:

```
<format>jpeg</format>
<dimension>
        <width>100</width>
        <height>100</height>
</dimension>
<colordepth>24</colordepth>
```

When the metadata is requested, this column will be read from the table and incorporated into an XML document along with the attributes from the other groups. The resulting document is a well-formatted XML document that contains all the information about that content.

The attributes of the third group (i.e. constraints) are implemented similarly. For example, for image content, the following constraint on size reduction is stored in the database table as a single column:

```
<reduce_size>
        <min_width>50</min_width>
        <min_height>50</min_height>
```

</reduce_size>

When the metadata is requested, this column will be incorporated into the XML document that constitutes the metadata of the image content.

An example of a complete metadata document for an image content, including attributes from the three groups, is shown in Figure 9 below.

Figure 9 An XML document containing the metadata of an image

```
<?xml version="1.0" encoding="UTF-8"?>
<mediaProfile>
	<title>Heart Surgery</title>
	<creator>ABC Hospital</creator>
	<description>A medical image of a human heart during a surgery.</description>
	<keywords>heart, surgery</keywords>
	<catagory>medical</catagory>
	<date>10/10/2005</date>
	<type>image</type>
	<language>en</language>
	<uri>http://someserver/somepath/heart.jpg</uri>
	<details>
		<size>3457</size>
		<format>jpeg</format>
		<colorDepth>24</colorDepth>
		<dimension>
			<height>320</height>
			<width>240</width>
		</dimension>
	</details>
	<constraints>
		<reduce_color>
			<min_color>8</min_color>
		</reduce_color>
	</constraints>
</mediaProfile>
```

In addition, an auto-increment field is used as an identifier that uniquely identifies each metadata record in the table.

The caches at the proxies (local proxy and content proxy) are implemented in the same manner so that a metadata based query from the caches is possible. If the caches were implemented as regular caches in which we store the whole XML document, we cannot query the content of the document for specific attributes of the content. Instead, we have to process each document individually for the attributes, which greatly affects the response time of metadata requests.

5.2.3 Implemented features

Most of the components of the system are implemented as classes in the Java programming language. The main components of the MDM are implemented as Java Servlets which are invoked through a standard web interface. Some of the major classes are discussed below.

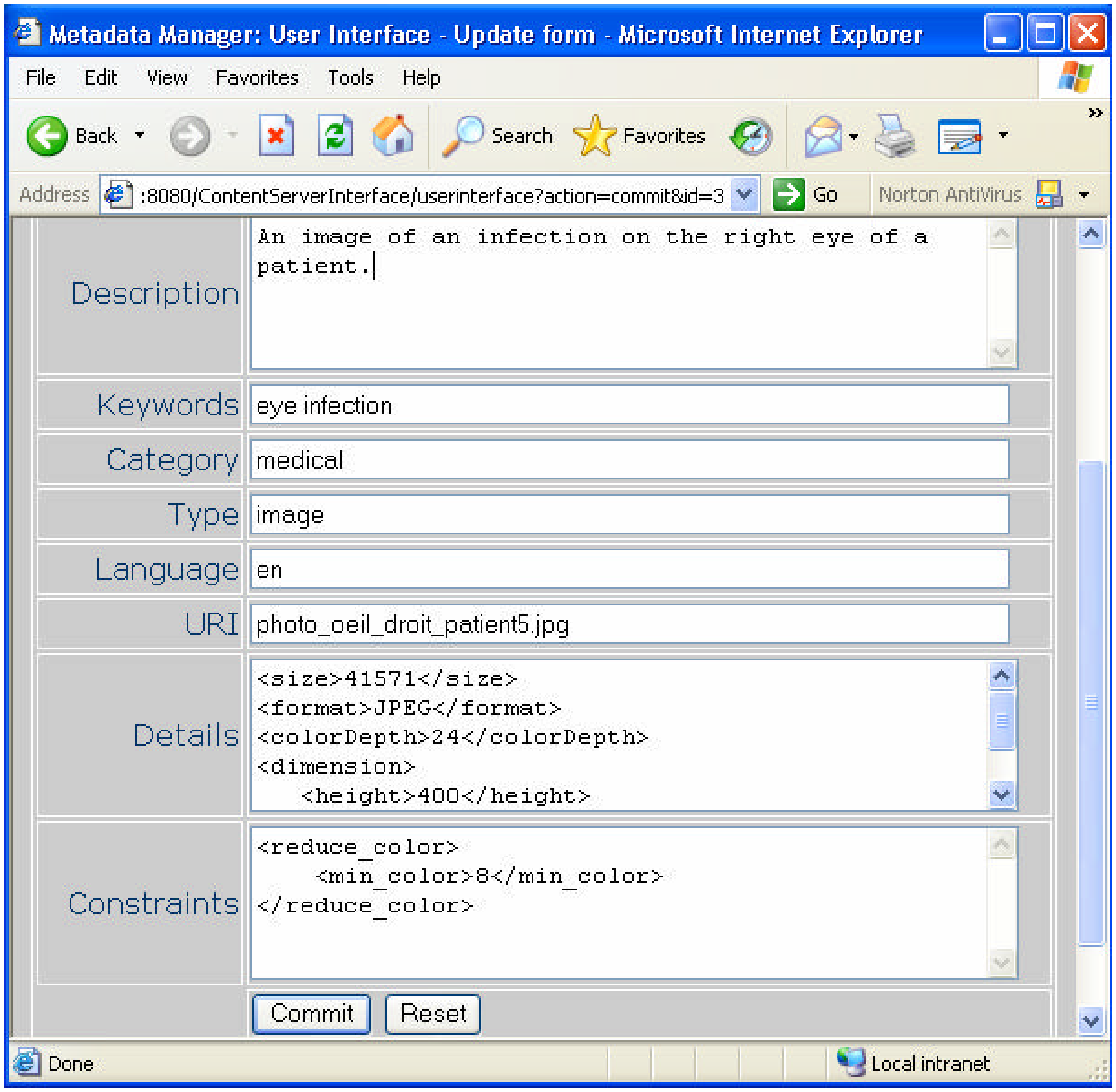
Class LPI: this class implements the Local Proxy Interface component of the MDM. It is a Java Servlet that listens to incoming request through a standard web interface. This class provides a method that implements the *getMetadata()* interface of the Local Proxy Interface which allows the retrieval of metadata based on certain criteria which are passed to it as parameters. The *getMetadata()* interface is implemented in such a way that if it does not find matching metadata documents from the local cache, it gets it from the cache at the content proxy or the MDR at the content server. However, this process is transparent to the caller. This class also provides a method for generating metadata for newly adapted content. In addition, it has methods for listening for incoming requests as well as for inserting/deleting/retrieving metadata to/from the cache.

Class CPI: this class implements the Content Proxy Interface component of the MDM. It is also a Java Servlet that binds to a certain port and accepts requests through a web interface. This class serves as a bridge between the Local Proxy Interface and the Content Server Interface. It listens for incoming metadata requests from the LPI. When a request arrives, it searches its cache for matching metadata, if it does not find any; it sends a request to the CSI. Ultimately, it returns the metadata it finds to the LPI, and caches it in the local cache if necessary. This class also has additional methods for listening for incoming requests as well as for insertion/deletion/retrieval of metadata to/from the local cache.

Class CSI: this class implements the Content Server Interface component of the MDM. This class is also implemented as a Java Servlet. It has methods for insertion/deletion/update/retrieval of metadata to/from the MDR. In addition, it instantiates and invokes the FEI class, which implements the Feature Extraction Interface of the MDM. At metadata insertion, the CSI class checks to see if there are missing features in the metadata. If there are missing features, it checks if there are Feature Extraction Engines (FEE) that can extract the features from the content. If the appropriate FEE is present, the missing features are extracted from the content and incorporated to the metadata, which is then inserted to the MDR. For experimental purposes, two Feature Extraction Engines (FEEs) are implemented. The first FEE extracts *file size*, *file name*, and *media type* from the content. It is a general FEE that can be applied to any of the media types. The second FEE extracts *width*, *height*, *format*, *file size*, and *color depth* from image content.

Class UI: this class implements the User Interface component of the MDM. It provides a web-based user interface for insertion/deletion/update/search of content and metadata to/from the MDR. It invokes the CSI with the appropriate action that it received from the user, and parameters gathered from the user interface. Figure 10 below shows one of the interfaces (update interface) provided by this class to the user for updating metadata documents.

Figure 10 The update metadata form of the UI



Metadata Manager: User Interface - Update form - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
:8080/ContentServerInterface/userinterface?action=commit&id=3
Go
Norton AntiVirus
Commit
Reset
Done
Local intranet

Description	An image of an infection on the right eye of a patient.
Keywords	eye infection
Category	medical
Type	image
Language	en
URI	photo_oeil_droit_patient5.jpg
Details	`<size>41571</size>` `<format>JPEG</format>` `<colorDepth>24</colorDepth>` `<dimension>` `<height>400</height>`
Constraints	`<reduce_color>` `<min_color>8</min_color>` `</reduce_color>`

Class <u>LocalProxy</u>: this class provides an interface to the user for query of content using metadata. To retrieve the metadata, it invokes the *getMetadata()* interface of the LPI with the query parameters accepted from the user. This class also analyzes client profile (which is basically a simplified profile of the client device) and the content metadata to determine whether content adaptation is necessary or not. If content adaptation is necessary and the constraints in the metadata do not prohibit the adaptation required, it performs the adaptation on the content and sends the adapted content to the client. It then caches the new content locally and invokes the *getnerateMetadata()* interface of the LPI to generate metadata for the new content (and cache it locally). The content adaptation is implemented for image content only, and it performs two types of adaptation: *scaling* and *color reduction*. The adaptation is not implemented as a web service due to time constraints. Instead it is implemented as a Java Servlet, called ImageAdapter, which can be invoked through a standard web interface. Neither the LocalProxy class nor the ImageAdapter class is part of the metadata management. However, they are required in order to gear the system into operation so that the metadata management can be operational.

There are a number of additional classes that perform various important activities. For example, there are utility classes that enable composing and sending SOAP messages, processing and manipulating XML documents, displaying HTML interfaces, etc.

5.3 Results

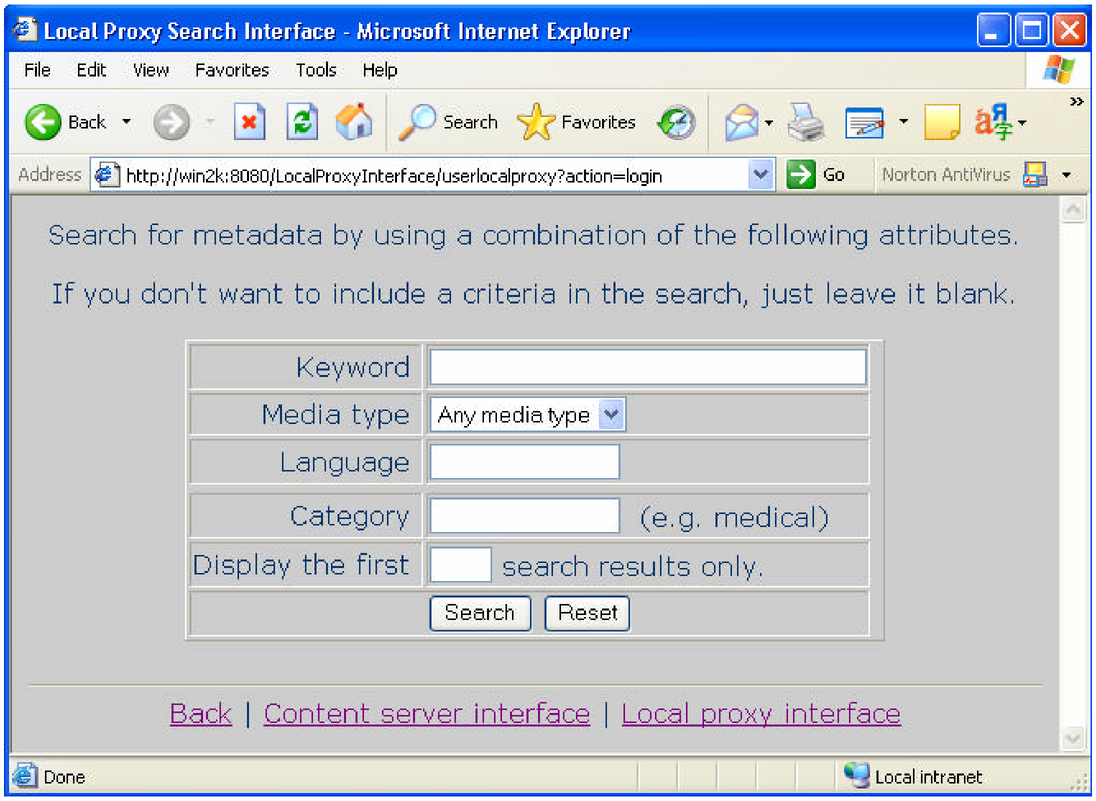
For testing purposes, a number of simplified client device profiles were created and searches for metadata were performed using these profiles. An example of a device profile used in testing the system is shown in Figure 11 below.

Figure 11 A simplified client device profile

```
<DeviceProfile>
        <screen_dimension>
                <width>240</width>
                <height>320</height>
        </screen_dimension>
        <colordepth>8</colordepth>
        <imagecapable>yes</imagecapable>
        <audiocapable>yes</audiocapable>
        <videocapable>no</vid eocapable>
</DeviceProfile>
```

The user selects a device profile at login. After login the user is presented with a search page through which he/she can search for content based on metadata. This search form is shown in Figure 12 below.

Figure 12 The content search interface of the local proxy


Local Proxy Search Interface - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address Go Norton AntiVirus
Done
Local intranet

http://win2k:8080/LocalProxyInterface/userlocalproxy?action=login

Search for metadata by using a combination of the following attributes.

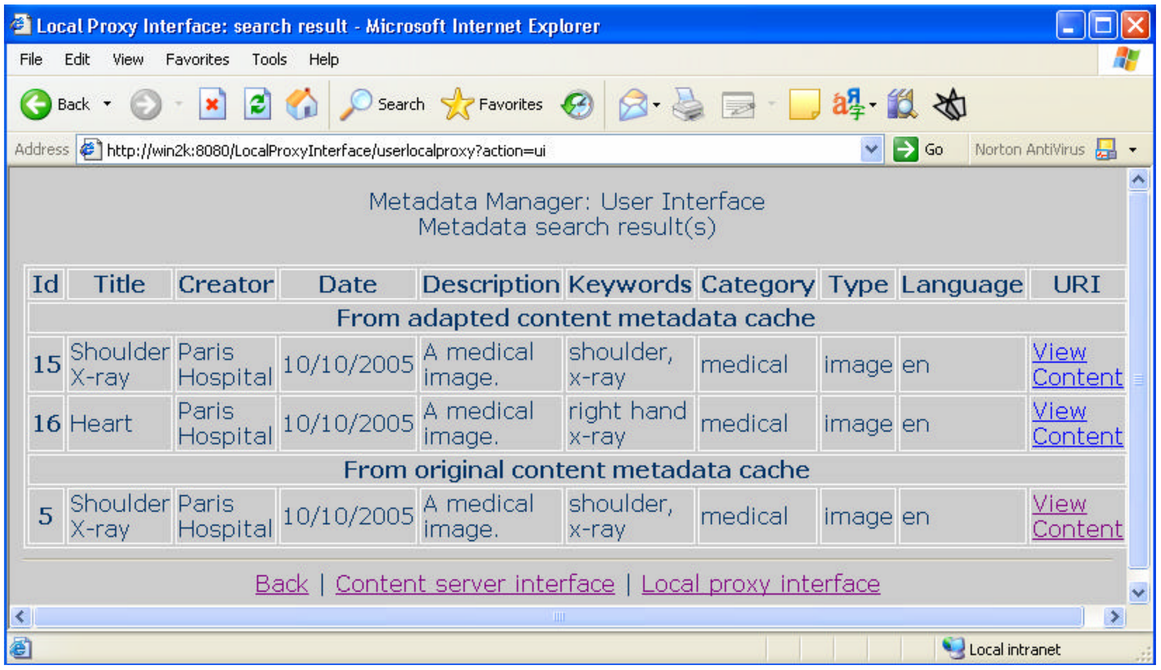
If you don't want to include a criteria in the search, just leave it blank.

Keyword	
Media type	Any media type
Language	
Category	(e.g. medical)
Display the first	search results only.
	Search Reset

Back | Content server interface | Local proxy interface

After searching, the user is presented with content metadata that match the query. The metadata may be that of adapted content, or original content. A search result for a query using this interface is shown in Figure 13 below.

Figure 13 A metadata search result



Local Proxy Interface: search result - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://win2k:8080/LocalProxyInterface/userlocalproxy?action=ui
Go
Norton AntiVirus
Local intranet

Metadata Manager: User Interface
Metadata search result(s)

Id	Title	Creator	Date	Description	Keywords	Category	Type	Language	URI
From adapted content metadata cache									
15	Shoulder X-ray	Paris Hospital	10/10/2005	A medical image.	shoulder, x-ray	medical	image	en	View Content
16	Heart	Paris Hospital	10/10/2005	A medical image.	right hand x-ray	medical	image	en	View Content
From original content metadata cache									
5	Shoulder X-ray	Paris Hospital	10/10/2005	A medical image.	shoulder, x-ray	medical	image	en	View Content

Back | Content server interface | Local proxy interface

The user may then select to view one of the results by clicking on the link "View Content". At this point the LocalProxy determines whether adaptation is required or not, and whether the constraints in the metadata allow the required adaptation or not. If adaptation is required and is allowed, the content is adapted to the client device profile and is displayed. The adapted content, together with its metadata which is generated by the LPI, may be cached at the local proxy to serve similar request in the future. If adaptation is not required, the content is displayed as it is. However, if adaptation is not allowed, and/or if a content adapter is not available, an appropriate message is displayed to the user.

6 Conclusion and future work

In this thesis work we have tried to address the issue of content metadata management in a pervasive environment by developing an architecture that facilitates the management of metadata. The architecture consists of a distributed Metadata Manager (MDM), a Metadata Repository (MDR), metadata caches, and a User Interface (UI). The MDM manages the metadata in such a way that as much information about the content as possible is supplied to the components of the adaptation/delivery framework that make use of the metadata. In addition to supplying information about the content, the metadata also provides information about the restrictions on the adaptation that may be applied on the content so that its semantics and quality are preserved after the adaptation.

The MDM consists of various components that are distributed over the system and perform logical portions of the management task. This distribution, however, is hidden from the rest of the system by providing a transparent interface through which the metadata can be accessed. The components that constitute the MDM are located at the local proxy, content proxy, and content server; and they communicate with each other through message passing.

Our architecture also incorporates additional components that further increase the usability of the metadata. The Feature Extraction Interface (FEI) of the MDM enriches the metadata by extracting missing features from the content by employing the appropriate Feature Extraction Engine (FEE). The Crosswalk Interface (CWI) of the MDM transforms metadata from one standard to another related one by using a suitable Crosswalk (CW).

Metadata is permanently stored in the Metadata Repository (MDR) at the content server, where the content is stored. However, to increase the efficiency of access, metadata is cached at the local proxy and content proxy by the appropriate component of the MDM.

The proposed metadata management architecture was implemented by developing a metadata management system as well as other supporting components of the adaptation/delivery framework (refer to Chapter 3 for details of the framework). For the metadata management system to operate properly, it was necessary to partially implement components of the framework that are not part of the MDM such as content adapters, client profiles, etc. Nevertheless, the MDM was successfully implemented and its operations in managing the metadata were experimentally observed.

Even though most of the issues regarding metadata management in a pervasive environment were addressed in this thesis work, there are areas which were not covered by the proposed architecture. One of these areas involves the implementation of feature extraction engines and metadata crosswalks as web services. In our architecture, Feature Extraction Engines (FEEs) and Crosswalks (CWs) are modeled as modules that can be integrated into the architecture. However, due to their nature, both of them are open, i.e. there can be new FEEs and CWs in the future that provide improved quality and functionality. Hence, if they are implemented as web services, the interface to invoke and use them will be more standardized. In addition, more possibilities can be considered and the one which is more appropriate can be used at any given instant. Consequently, the metadata architecture could be improved in the future to provide standard interfaces through which the FEEs and CWs can be invoked.

The other issue to be addressed in a future work is regarding the implementation of the system. Due to the fact that most of the supporting components of the architecture were simulated (decision on adaptation, adaptation strategy optimization, client profile management, content adaptors, and others), performance measurements could not be performed on the adaptation/delivery system as a whole. Therefore, in a future work, these components of the framework could be implemented in a more realistic manner and performance measurements could be conducted to further fine tune the system so that it provides the best possible service to the client.

Reference

[1] Mark Weiser, *The Computer for the 21st Century*, Scientific American, September 1991.

[2] M. Satyanarayanan, *Pervasive Computing: Vision and Challenges*, IEEE Personal Communications, August 2001.

[3] Debashis Saha, Amitava Mukherjee, *Pervasive Computing: A Paradigm for the 21st Century*, IEEE Computer Society, March 2003.

[4] Karen Henricksen, Jadwiga Indulska, Andry Rakotonirainy, *Infrastructure for Pervasive Computing: Challenges*, Workshop on Pervasive Computing INFORMATIK 01, 2001.

[5] Girma Berhe, Lionel Brunie, Jean-Marc Pierson, *Modeling Service-Based Multimedia Content Adaptation in Pervasive Computing*, Proceedings of the 1st conference on Computing frontiers, pages 60 – 69, Ischia, Italy, 2004.

[6] Zhijun Lei, Nicolas D. Georganas, *Context-based Media Adaptation in Pervasive Computing*, Proc. of Canadian Conference on Electrical and Computer Engineering (CCECE), Toronto, May 2001.

[7] John R. Smith, Rakesh Mohan, Chung-Sheng Li, *Scalable Multimedia Delivery for Pervasive Computing*, ACM Multimedia, Orlando, FL, November 1999.

[8] Karen Henricksen, Jadwiga Indulska, Andry Rakotonirainy, *Modeling Context Information in Pervasive Computing Systems*, Proceedings of the First International Conference on Pervasive Computing, pages 167 – 180, 2002.

[9] Albert Held, Sven Buchholz, Alexander Schill, *Modeling of Context Information for Pervasive Computing Applications*, Proc. of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI2002), Orlando, FL, Jul 2002.

[10] László Böszörményi, Hermann Hellwagner, Harald Kosch, Mulugeta Libsie, Stefan Podlipnig, *Metadata driven adaptation in the ADMITS project*, Signal

Processing: Image Communication, Volume 18, Issue 8, pages 749-766, September 2003.

[11] NISO Press, *Understanding Metadata*, http://www.niso.org.

[12] Mulugeta Libsie, *Metadata Supported Content Adaptation in Distributed Systems*, Ph. D. Thesis, University Klagenfurt, Austria, June 2004.

[13] Susanne Boll, Wolfgang Klas, Amit Sheth, *Multimedia Data Management: Using Metadata to Integrate and Apply Digital Media*, pages 14-16, McGraw-Hill, March 1998.

[14] Bohms, K and T. Rakow, *Metadata for Multimedia Documents*. ACM SIGMOD RECORD, Vol. 23, No. 4, December 1994, pages 21-26.

[15] Susanne Boll, Wolfgang Klas, Amit Sheth, *Multimedia Data Management: Using Metadata to Integrate and Apply Digital Media*, pages 16-19, McGraw-Hill, March 1998.

[16] *www.dubincore.org* - The Dublin Core metadata standard website.

[17] Jane Hunter, Renato Iannella, *The Application of Metadata Standards to Video Indexing*, Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries, pages 135 – 156, 1998.

[18] ISO/IEC FDIS 15938-5, *Information Technology — Multimedia Content Description Interface — Part 5: Multimedia Description Schemes.*

[19] J. Lacasta, J. Nogueras-Iso, M. P. Torres, F. J. Zarazaga-Soria, *Towards the Geographic Metadata Standard Interoperability*, Proceedings of the 6th AGILE, Lyon, France, April 24-25, 2003.

[20] Margaret St. Pierre, William P. LaPlant, Jr., *Issues in Crosswalking Content Metadata Standards,* http://www.niso.org/press/whitepapers/crosswalking.html, visited on May 16, 2005.

[21] Harald Kosch, László Böszörményi, Mario Döller, Mulugeta Libsie, Peter Schojer, Andrea Kofler, *The Lifecycle of Multimedia Metadata*, IEEE MultiMedia, vol. 12, no. 1, pp. 80-86, January/March 2005.

[22] Brigitte Kerhervé, Oliver Gerbé, *Models for Metadata or Metamodels for Data?*, Proc. 2nd IEEE Metadata Conference, 1997.

[23] James Griffioen, Raj Yavatkar, Robert Adams, *Automatic and Dynamic Identification of Metadata in Multimedia*, 1st IEEE Metadata Conference. 1996.

[24] Ulrika Hedström, *Automatic Generation of Metadata for Indexing, Searching and Navigating in an Intranet*, Masters Thesis, Uppsala University, July 1999.

[25] Ediz Saykoly Ugur Güdükbay, Özgür Ulusoy, *A Semi_Automatic Object Extraction Tool for Querying in Multimedia Databases*, 7th Workshop on Multimedia Information Systems MIS'01, 2001.

[26] Tom Clements, *Overview of SOAP*, http://java.sun.com/developer/technicalArticles/xml/webservices/, visited on March 3, 2005.

[27] The SOAP version 1.1 specification, http://www.w3.org/TR/SOAP.

[28] R. Allen Wyke, Sultan Rehman, Brad Leupen, *XML Programming*, Microsoft press, 2002, pages 277-279.

[29] Girma Berhe, Lionel Brunie, Jean-Marc Pierson, *Realization of distributed content adaptation with service-based approach for pervasive systems.*

Declaration

I, the undersigned, declare that this thesis is my original work and has not been presented for a degree in any other university, and that all source of materials used for the thesis have been duly acknowledged.

Declared by:

Name:	______________________________
Signature:	______________________________
Date:	______________________________

Confirmed by advisor:

Name:	______________________________
Signature:	______________________________
Date:	______________________________

Place and date of submission: Addis Ababa, July 2005.