Luís Filipe Pinto de Almeida Teixeira

Edition and Description Framework for Video Objects
Edition and Description Framework for Video Objects

Luís Filipe Pinto de Almeida Teixeira
Edition and Description Framework for Video Objects

Luís Filipe Pinto de Almeida Teixeira

Dissertação submetida para satisfação parcial dos requisitos do grau de mestre em Redes e Serviços de Comunicação

Dissertação realizada sob a supervisão do Professor Doutor Luís António Pereira de Meneses Corte-Real, do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Porto, Julho de 2004
Supervised by:
Luís António Pereira de Meneses Corte-Real

Original Title:
Criação e edição de objectos audiovisuais suportando as normas MPEG-4 e MPEG-7

Esta tese foi financiada pelo III Programa Quadro Comunitário de Apoio, co-financiada pela Fundação para a Ciência e a Tecnologia (FCT) e Fundo Social Europeu (FSE).

Copyright © 2004 by Luís F. Teixeira

All rights reserved. No part of the material protected by this copyright notice may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage and retrieval system, without the prior permission of the author.

Typeset by the author with the IMPaX Documentation System.
Author email: luis.f.teixeira@inescporto.pt
To my mother, my father and to Anita.
Resumo

A forma como a humanidade tem registado o seu conhecimento ao longo dos tempos evoluiu de forma dramática. Actualmente, dispositivos incrivelmente pequenos podem armazenar centenas de livros ou fotografias, horas de voz, de música ou de vídeo. A informação digital alcançou todos os tipos de expressão e, acima de tudo, adquiriu importantes propriedades: portabilidade, mobilidade e ubiquidade. Apesar deste cenário promissor, alguns obstáculos têm dificultado a sua concretização plena.

A miríade de formatos e tecnologias utilizadas para codificar informação nem sempre permite uma convergência real. Para fazer face a este problema, o Moving Picture Experts Group (MPEG) especificou uma tecnologia que permite a composição e transmissão de conteúdos multimédia interactivos a débitos variáveis sobre ligações com diferentes larguras de banda – a norma MPEG-4. Por outras palavras, uma tecnologia que permite agregar todo o tipo de informação num formato de conteúdos enriquecidos, e que pode ser transmitido em todos os ambientes.

Os conteúdos audiovisuais são criados a um ritmo cada vez mais acelerado. A necessidade de processamento que lhes está associada tem também vindo a crescer. No entanto, a informação que não pode ser encontrada não é muito útil. São portanto necessárias representações alternativas que permitam algum tipo de interpretação do significado da informação. Com esse intuito, a norma MPEG-7 define a sintaxe e a semântica para a descrição de conteúdo multimédia.

Apesar das duas normas serem relativamente recentes, a sua aceitação não atingiu as expectativas iniciais. É consensual que qualquer tecnologia multimédia pode apenas ter sucesso se alcançar ambos os extremos da cadeia – os autores e os consumidores. Para atingir este objectivo são necessárias ferramentas que possam satisfazer as exigências dos utilizadores. Isso ainda não aconteceu no caso do MPEG-4 e, inerentemente, do MPEG-7.

Esta dissertação aborda esta necessidade, especificando uma plataforma de autoria modular, flexível e escalável para multimédia suportando ambas as normas. Um protótipo foi também implementado para efectuar a asserção dessa especificação. Este abrange uma ferramenta de composição, um servidor para armazenamento e distribuição, e uma aplicação cliente.

Palavras-chave: Autoria de multimédia, reutilização de conteúdo, MPEG-4, descrição de conteúdo multimédia, conteúdo semântico, MPEG-7
Abstract

The way humankind has recorded its knowledge has changed dramatically. Today, incredibly small devices can store hundreds of books or pictures, hours of speech, music or video. Digital information has reached all types of expression and, above all, it acquired important properties: *portability, mobility* and *ubiquity*. Despite this promising scenario, some obstacles have made its concretization a difficult task.

The myriad of formats and technologies used to encode information not always allows a real convergence. To face this problem, the Moving Picture Experts Group (MPEG) specified a technology that allows the composition and delivery of interactive multimedia content at variable bit rates over high and low bandwidth connections – the MPEG-4 standard. In other words, a technology that *aggregates* all types of information in a rich-content format which can be delivered everywhere.

Audiovisual content is created at an increasingly fast pace. These sources of information also have an increasing need for processing. Nevertheless, the information that can not be found is not of much use. Alternate representations that allow some kind of interpretation about the meaning of this information are therefore required. For that purpose, the MPEG-7 standard defines the syntax and semantics for the *description* of multimedia content.

Although both standards are relatively recent, its acceptance has not reached the initial expectations. It is believed that any multimedia technology can only be successful if it reaches both ends of the chain – *authors* and *consumers*. To accomplish this objective, tools that can satisfy its users demands are needed. This has not yet happened for MPEG-4 and, inherently, for MPEG-7.

This dissertation addresses this need, proposing a specification for a modular, flexible and scalable *authoring framework* for multimedia, supporting both standards. A prototype has also been implemented to assert this specification. It consists of a multimedia composition tool, a storage and delivery server and a client application.

**Keywords:** Multimedia authoring, content reuse, content repurpose, MPEG-4, multimedia content description, semantic content, MPEG-7
Résumé

La forme comme l’humanité a enregistré sa connaissance au long des temps a évolué de forme dramatique. Actuellement, les dispositifs incroyablement petits peuvent stocker des centaines de livres ou photographies, heures de voix, de musique ou de vidéo. L’information digital a atteint tous les types d’expression et surtout elle a acquis les propriétés importantes : portabilité, mobilité et ubiquité. Malgré de ce scénario prometteur, quelques obstacles ont rendu difficile leur concrétisation complète.

La myriade de formats et de technologies utilisées pour coder l’information pas toujours permet une vraie convergence. Pour faire face à ce problème, Moving Picture Experts Group (MPEG) a spécifié une technologie qui permet la composition et la transmission de contenus multimédia interactifs à des débits variables sur des liaisons avec de différentes largeurs de bande. Par autres mots, une technologie qui permette agréger tout le type d’informations dans un format de contenus enrichis, et que peut être transmis dans tous les environnements.

Les contenus audiovisuels sont créés à un rythme de plus en plus accéléré. La nécessité de traitement qui leur est associé est aussi venue à augmenter. Néanmoins, les informations qui ne peuvent pas être trouvées ne sont pas très utiles. Ce sont donc nécessaires représentations alternatives qui permettent quelque type d’interprétation de la signification des informations. Avec cette intention, la norme MPEG-7 définit la syntaxe et la sémantique pour la description de contenu multimédia.

Malgré des deux normes de trente relativement récente, son acceptation n’a pas atteint les attentes initiales. C’est consensuel que une technologie multimédia peut seulement avoir succès s’atteindre les deux extrémités de la chaîne – des auteurs et des consommateurs. Pour atteindre cet objectif sont nécessaires outils qui puissent satisfaire les exigences des utilisateurs. Cela encore n’est pas arrivé dans le cas de la MPEG-4 et, inhérentement, de la MPEG-7.


Mots-clés: Création de multimédia, réutilisation de contenu, MPEG-4, description de contenu multimédia, contenu sémantique, MPEG-7
Preface

"Facing today's scenario in the audiovisual communications area, it is possible to predict its evolution in a near future in four distinct, yet complementing, scenarios. The first is related with mobility and consequently with multimedia content access in mobile terminals. The second refers to the growth of domestic networks, where several equipments that usually populate our houses will be connected to the Internet and communicate with each other, offering access to contents. In the third scenario the personal computer will still be the privileged access to multimedia contents. Finally, in a fourth scenario, television will see an increase on its protagonism, becoming interactive, and emerging as the main source of multimedia content." [1]

The year was 2001 and the previous analysis introduced my undergraduate final project's report. Three years later, all scenarios remain plausible, although the recent boom on mobile access to multimedia content suggests that the first scenario has gained some edge over the others. Now, as back then, the MPEG-4 standard seems as a quiescent panacea. To be reasonable, probably less now than then. The industry seems to have adopted part of the technologies of MPEG-4, especially the audiovisual encoders. For example, AAC (MPEG-4 Part 2) is the underlying technology of the digital music store, pioneered by Apple. Another example is H.264/AVC (MPEG-4 Part 10) that has recently been adopted by the DVD Forum to be included in the next generation of High Definition DVD format, together with MPEG-2 and WMV9. Nevertheless, the more interesting features promised by MPEG-4 remain somewhat forgotten and the industry seems to be sceptical about the MPEG-4 as a whole. Maybe the concepts behind the standard are too complex and only time will allow consumers and manufacturers to absorb all its potentialities. Maybe the odd licensing is keeping interest away. Maybe a killer app is yet to come. Or maybe none is true and a new technology, yet to be developed, will definitely
make a difference. Probably three years from now we will have more assurances.

Meanwhile, the MPEG has finished the MPEG-7 standard for multimedia description and began specifying a new standard for content management – the MPEG-21 — and, interestingly, work among the MPEG-4 team has been increasingly more active. Streaming of text, compression and streaming of fonts and synthesized textures and the Animation Framework eXtension (AFX) are some of the important new features making MPEG-4 even more complete. As for the MPEG-7 standard, its main virtue when compared to the preceding metadata standards, its generality, seems to be also its critical point. And, once more, the main development of MPEG-7 is almost confined to academic research projects. All hopes are now on MPEG-21 as the unifying standard. Only time will say if this is reality or a larger-than-life fiction.

Despite this grim perspective, motivation was high at the beginning of this project. Multimedia content will always be created and consumed, whether using MPEG-4 and MPEG-7 or not. Yet, this dissertation tries to demonstrate that MPEG-4 and MPEG-7 can be used together in a real scenario for the authoring of multimedia content and that its potentials are yet to be fully explored. Hope I can accomplish that.

Porto, Luís F. Teixeira
July 2004

Reader Notes

This document is composed of six chapters. All chapters begin with a small introduction to its contents. In the end, bibliographic references are listed. These references are identified in the text using a number within square brackets.

Not surprisingly, many acronyms are used throughout the text. To avoid cluttering the text with definitions, a list of acronyms at the beginning of the document is provided, allowing easy referencing.

A final note regarding the fonts used to typeset this document. The font used in body text is Computer Modern Roman and the blocks containing program code and class or module names use the Adobe Courier font. Chapter and section titles are in various sizes of Adobe Helvetica-Narrow Bold.
Acknowledgments

What an incredible journey this was.

First I would like to gratefully acknowledge the wise supervision of Prof. Luis Cortereal during this project. Many thanks also to Luis Teixeira (not me!) for his keen suggestions. I am also forever indebted to my friend and roommate at INESC Porto, Jaime Cardoso for his immense patience in reading and revising the text.

This dissertation would not be possible without the kind support provided by FCT - Fundação para a Ciência e a Tecnologia (Foundation for the Science and the Technology). Many thanks to them for supporting and promoting scientific research in Portugal.

I am also deeply thankful to INESC Porto for giving me the opportunity to work for the past three years in very compelling research projects. Because of that, to all, with whom I had the privilege to work with, I owe a lot of gratitude. However, as unfair as it may be I would like to mention some persons. Pedro Cardoso, Pedro Ferreira and Ricardo Morla for their warm welcome and teaching when I was the newbie around (now I am beginning to feel the oldie around). Filipe Sousa for being such a supportive colleague and friend. And Mário, Barral, Orlando, Vítor, João, and all the former MOG team at INESC Porto for embracing me in such a great group.

A final word of thanks goes to my family and friends. To my parents, for being my parents and their immense support. To my sister Teresa, for her kindness. To Ana, for her endless love. To Nês, Zé Pedro, Jorge, Pedro Q., Nuno and all my true friends for their support.
Table of Contents

Resumo ........................................ i
Abstract ....................................... iii
Résumé ......................................... v
Preface .......................................... vii
Table of Contents ................................ x
List of Figures .................................. xiv
List of Tables ................................... xvii
Acronyms ....................................... xix

1 Introduction .................................. 1
   1.1 Overview .................................. 1
   1.2 Scope ...................................... 3
   1.3 Structure .................................. 5

2 Multimedia Representation and Description 7
   2.1 Overview .................................. 7
   2.2 Representing Multimedia Content .......... 9
       2.2.1 MPEG-4 ............................... 9
       2.2.2 SMIL ................................. 20
       2.2.3 SVG ................................. 21
TABLE OF CONTENTS

2.2.4 X3D ................................................. 22
2.2.5 Flash ............................................. 23
2.2.6 Comparison ...................................... 23
2.3 Describing Multimedia Content .................................. 28
  2.3.1 MPEG-7 ........................................... 28
  2.3.2 DC ............................................... 32
  2.3.3 RDF ............................................. 33
  2.3.4 AAF ............................................. 34
  2.3.5 Comparison ...................................... 35

3 Multimedia Authoring ........................................... 37
  3.1 Overview ........................................... 37
  3.2 Multimedia Authoring Paradigms .............................. 38
    3.2.1 Structure ........................................ 39
    3.2.2 Flowchart ....................................... 39
    3.2.3 Timeline ........................................ 40
    3.2.4 Script .......................................... 41
    3.2.5 Comparison ...................................... 41
  3.3 Requirements ......................................... 43
    3.3.1 Expressive Power ................................ 43
    3.3.2 Authoring Capabilities ............................ 45
  3.4 Analysis of Available Authoring Tools ....................... 46
    3.4.1 MPEG-4 Authoring Tools ........................... 46
    3.4.2 Other Authoring Tools ............................. 48

4 Design and Specification ......................................... 51
  4.1 System Requirements .................................... 51
  4.2 Use Cases ........................................... 52
    4.2.1 Authentication Manager ................................ 55
    4.2.2 Storage Manager .................................... 57
    4.2.3 Authoring Tool .................................... 58
    4.2.4 Profile Manager .................................... 60
    4.2.5 Multimedia Service Provider ......................... 60
  4.3 A Distributed Architecture ................................ 62
  4.4 Data Model ........................................... 64
    4.4.1 Media-Data ....................................... 65
### TABLE OF CONTENTS

4.4.2 Metadata ......................................................... 65  
4.5 User Interfaces .................................................... 70  

5 A Multimedia Authoring Framework: edVO 73  
5.1 Overview .......................................................... 73  
5.2 Integration of External Tools ..................................... 76  
  5.2.1 GPAC ......................................................... 76  
  5.2.2 FOX Toolkit .................................................. 79  
  5.2.3 Mozilla ...................................................... 80  
  5.2.4 Zope .......................................................... 82  
  5.2.5 Darwin Streaming Server .................................... 83  
5.3 Design and Implementation Analysis .............................. 83  
  5.3.1 Composer ..................................................... 83  
  5.3.2 Server ......................................................... 88  
5.4 Graphical User Interfaces .......................................... 91  

6 Summary and Conclusions 97  
6.1 Summary .......................................................... 97  
6.2 Application of the Work Done ................................... 99  
6.3 Further Developments ............................................ 100  
  6.3.1 Implementation-wise ......................................... 100  
  6.3.2 Integration-wise ............................................. 101  
6.4 Conclusion ........................................................ 102  

List of References ............................ 105  

Index .............................................................. 111
# List of Figures

1. Multimedia content delivered to different devices ........................................ 2  
3. Typical example of different sources being blended - weather forecast program. 8  
4. MPEG-4 object-based representation architecture .......................................... 10  
5. Typical MPEG-4 system .................................................................................. 11  
6. Hierarchical representation of a scene .............................................................. 13  
7. Object descriptors linking scene descriptors to elementary streams [19] ............ 18  
8. MPEG-4 systems layer model [29] .................................................................. 19  
9. Relation between MPEG-4, SMIL and X3D (based on [29]) .......................... 24  
10. MPEG-7 main elements [42] .......................................................................... 30  
11. Overview of MDS tools [41] .......................................................................... 32  
12. Relationship between content description standards ........................................ 36  
13. Structure-based paradigm .............................................................................. 39  
14. Flowchart paradigm ....................................................................................... 40  
15. Timeline paradigm ......................................................................................... 40  
16. Script-based paradigm .................................................................................... 41  
17. MPEG-4 authoring tools ................................................................................ 47  
18. Other multimedia authoring tools .................................................................. 49  
19. Subsystems diagram ....................................................................................... 54  
20. Authentication subsystem use cases diagram .................................................. 55  
21. Sequence diagram of a typical resource access .............................................. 56  
22. Storage Manager subsystem use cases diagrams ............................................ 57  
23. Authoring subsystem use case diagram .......................................................... 58  
24. Typical authoring sequence diagram .............................................................. 59
<table>
<thead>
<tr>
<th>Fig No.</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>Profile Manager subsystem use case diagram</td>
<td>60</td>
</tr>
<tr>
<td>26</td>
<td>Multimedia Service Provider use case diagram</td>
<td>61</td>
</tr>
<tr>
<td>27</td>
<td>Request of multimedia service typical sequence</td>
<td>62</td>
</tr>
<tr>
<td>28</td>
<td>System architecture</td>
<td>63</td>
</tr>
<tr>
<td>29</td>
<td>Hierarchy of top-level types</td>
<td>67</td>
</tr>
<tr>
<td>30</td>
<td>Hierarchy task analysis of Media Object search</td>
<td>72</td>
</tr>
<tr>
<td>31</td>
<td>edVO prototype architecture</td>
<td>74</td>
</tr>
<tr>
<td>32</td>
<td>GPAC modules</td>
<td>77</td>
</tr>
<tr>
<td>33</td>
<td>Embedding Mozilla</td>
<td>81</td>
</tr>
<tr>
<td>34</td>
<td>Creating an object in Composer</td>
<td>84</td>
</tr>
<tr>
<td>35</td>
<td>Relationship between classes in Composer</td>
<td>85</td>
</tr>
<tr>
<td>36</td>
<td>Server interface with composer using a proxy</td>
<td>86</td>
</tr>
<tr>
<td>37</td>
<td>Server storage structure</td>
<td>89</td>
</tr>
<tr>
<td>38</td>
<td>Server storage structure</td>
<td>90</td>
</tr>
<tr>
<td>39</td>
<td>Composer – main window</td>
<td>93</td>
</tr>
<tr>
<td>40</td>
<td>Composer – changing the gradient colors</td>
<td>93</td>
</tr>
<tr>
<td>41</td>
<td>Composer – changing the text</td>
<td>94</td>
</tr>
<tr>
<td>42</td>
<td>Composer – linking a route</td>
<td>94</td>
</tr>
<tr>
<td>43</td>
<td>Composer – browsing the repository</td>
<td>95</td>
</tr>
<tr>
<td>44</td>
<td>Server – browsing a multimedia presentation</td>
<td>95</td>
</tr>
<tr>
<td>45</td>
<td>Software evolution model</td>
<td>100</td>
</tr>
</tbody>
</table>
List of Tables

1  Comparison between multimedia description languages (based on [35]) ... 27
2  Properties of authoring paradigms ........................................... 42
3  Brief description of the Authentication Manager use cases .............. 56
4  Brief description of the Storage Manager use cases ...................... 58
5  Brief description of the Authoring Tool use cases ........................ 59
6  Brief description of the Profile Manager use cases ....................... 60
7  Brief description of the Multimedia Service Provider use cases ......... 61
8  MPEG-7 root elements used ..................................................... 66
9  MPEG-7 top-level types used ................................................... 68
Acronyms

AAC  Advanced Audio Coding
AAF  Advanced Authoring Format
AAL  ATM Adaptation Layer
API  Application Programmer Interface
ATM  Asynchronous Transfer Mode
AU   Access Unit
AVC  Advanced Video Coding
BIFS Binary Format for Scenes
D    Descriptor
DC   Dublin Core
DCMI Dublin Core Metadata Initiative
DDL  Description Definition Language
DMIF Delivery Multimedia Integration Framework
DOM  Document Object Model
DS   Description Scheme
ES   Elementary Stream
GPAC GPAC Project on Advanced Content
ACRONYMS

GUI  Graphical User Interface
HTTP  HyperText Transfer Protocol
IEC  International Electrotechnical Commission
IETF  Internet Engineering Task Force
IP  Internet Protocol
IPMP  Intellectual Property Management and Protection
IS  International Standard
ISO  International Organization for Standardization
JPEG  Joint Photographic Experts Group
MDS  Multimedia Description Scheme
MO  Media Object
MP  Multimedia Presentation
MP3  MPEG-2 layer 3 audio coding
MPEG  Moving Picture Experts Group
MXF  Material eXchange Format
OCI  Object Content Information
OD  Object Descriptor
OMG  Object Management Group
PNG  Portable Network Graphics
QoS  Quality of Service
RBAC  Role-Based Access Control
RDF  Resource Description Framework
RTP  Real-Time Transport Protocol
RTSP  Real-Time Streaming Protocol
SL  Synchronisation Layer
SMIL  Synchronized Multimedia Integration Language
SMPTE  Society of Motion Picture and Television Engineers
SVG  Scalable Vector Graphics
UDP  User Datagram Protocol
UI  User Interface
UML  Unified Modeling Language
URI  Uniform Resource Identifier
URL  Uniform Resource Locator
UUID  Universal Unique IDentifier
VRML  Virtual Reality Modeling Language
W3C  World Wide Web Consortium
WYSIWYG  What You See Is What You Get
X3D  eXtensible 3D
XML  eXtended Markup Language
XMT  eXtensible MPEG-4 Textual format
Chapter 1

Introduction

Starting point. From the pieces of information to the composing tools that weave interactive and compelling multimedia presentations. This chapter gives a concise look over the current state of multimedia content creation and what led to it. It also presents the main objectives of the work developed toward this dissertation. In the end, a brief description of what follows is also presented.

1.1 Overview

Since the dawn of human knowledge, humankind has collected and recorded information. From the first known cave paintings, to the Gutenberg bible, and to the current multi-terabyte multimedia databases, giant leaps of knowledge have occurred. But, in essence, humankind always intended to represent knowledge itself and preserve it. The introduction of computer technologies in the mid XXth century, emerged as a new storage medium. However, the trigger that set the diffusion of information ubiquitously was the introduction of the World Wide Web.

Typical conveyance methods, such as paper, film or magnetic tape restrict the use of information. Current technologies provide the ability to unify in a digital environment the media used to represent different types of information. Different media can thus become part of a larger whole. The creation of on-line, time-based presentations is one important example of this [2]. Time-based presentations allow the composition of different pieces of information, defining when and where they appear on the screen. Current presentations allow the assembly of text, image and time-based media elements and the synchronisation and referencing of elements. In order to create and view these multimedia presentations, tools for building and displaying them are required.
The definition of multimedia is not, however, very consensual. A possible definition for multimedia is that it is a computer-based interactive experience that incorporates text, graphics, sound, animation, video, and virtual reality. We will be using this definition throughout the text.

Even though multimedia is becoming widely available in the World Wide Web, it does not yet reach all environments. The MPEG-4 standard tries to overcome this [3, 4]. It is designed for a wide range of environments, as illustrated in Figure 1, from high capacity devices (e.g. PCs with multimedia capabilities) to low bit-rate communications devices (e.g. mobile devices). To handle this diversity, MPEG-4 allows the creation of scalable content, which can be encoded once and automatically reproduced in devices having distinct capabilities with acceptable quality. An important feature that enables scalability in MPEG-4 is the object-based composition of different types of media objects — video, images or audio of synthetic or natural origin, speech and text. Each object is encoded and decoded separately and its delivery to the client device may depend on the transmission’s conditions.

![Figure 1: Multimedia content delivered to different devices](image)

Another problem that grows with the diffusion of multimedia information is the ability of effectively cataloguing and searching this content using familiar concepts to the users. The MPEG-7 standard addresses the description of multimedia content [5], providing the means to navigate through an increasingly large pool of knowledge and information. Moreover, the association of additional information to multimedia content as it is created provides valuable information to content adaptation engines throughout the delivery chain.
depicted in Figure 1. These adaptation engines aim at providing the most suitable content, considering device capabilities and user preferences.

The joint use of MPEG-4 and MPEG-7 may contribute to overcome some of these problems. This dissertation specifies an modular, authoring system for multimedia content that is based on the combined of both standards. Its scope is defined next.

1.2 Scope

The creation of content typically follows a well-known process. Figure 2 shows what is called the content life-cycle. The stages of this cycle include:

- **Commission** – the commissioning of the content by a producer begins the content creation cycle.
- **Elaboration** – iterative process that defines the concept of the content, and is performed during pre-production.
- **Capture** – during this stage the content is brought into the database; it includes shooting, graphic origination, archive research and acquisition; content may exist prior to either Commission or Elaboration.
- **Analysis** – includes, for example, the processes of feature extraction and motion analysis.
- **Synthesis** – iterative process in which extracted descriptions are merged to synthesize useful descriptions.
- **Composition** – first stage of editing to form the intended content.
- **Packaging** – building of the final version of the content to be distributed.
- **Delivery** – includes the transfer of content from the production form to the delivery conveyor: content server, web server, packaged media, or broadcast.
- **Consumption** – stage where the content is viewed or received.
- **Interaction** – stage where the consumer interacts with the content.
- **Archiving** – library functions which include archive or long term storage, retrieval and access.

This dissertation focus especially on the *Composition* stage of content creation as well as on *Archiving*. Some parts of *Synthesis* and the stages that follow *Composition* are also covered to some extent. We do not cover, however, the first stages: *Commission, Elaboration, Capture and Analysis*. It is assumed that raw content is already created and available for composition.
The main objective put forward when the project was undertaken was to design a flexible and generic framework to allow the creation, manipulation and publishing of multimedia content.

To accomplish that, and before attempting a specification, a thorough study was firstly ensued. Although the use of MPEG standards were considered right from the beginning, other technologies that allow the representation and description of multimedia content were also studied. After directly comparing all available options, the first choice remained the most suitable for the system in hands.

Another study considering multimedia authoring was also performed. The objectives were to know which elements comprehend the foundation basis of a multimedia authoring system as well as its requirements. Several projects addressing multimedia authoring were also studied to endeavour a specification that improves on already developed projects. One issue that received less interest from previous projects, that focused mainly on authoring,
was the association of multimedia content with metadata. However, in this dissertation we attempt to include in the authoring process in order to complement and enhance content creation.

Finally, a multimedia authoring framework was specified and partially implemented in a prototype which tries to validate the design options. The specification included the definition of requirements, typical use cases, the data model used to store content and metadata and user's interfaces.

1.3 Structure

Concluding the introduction to this dissertation, some clues for the remaining chapters are now given.

The next Chapter presents an overview of the technologies available nowadays to represent and describe multimedia content. The MPEG-4 and MPEG-7 have the most insightful overviews but other competing technologies are also analysed and compared.

Chapter 3 defines the concepts of multimedia authoring. The different approaches followed by multimedia authoring system, called multimedia paradigms are studied as well as the system's requirements. An overview of available authoring tools using different technologies and paradigms is also presented.

In Chapter 4 a generic authoring framework for multimedia content is presented. The system requirements and use cases are specified and a distributed architecture is proposed. The data model used for the content is also specified as well as the user interfaces.

Chapter 5 describes the implementation of a prototype used to assert the specification presented in Chapter 4. It gives some implementation details and describes the strategy followed to implement the system.

Finally, Chapter 6 summarizes the dissertation's content and an evaluation of possible applications of its work. Short-term improvements but also long-term improvements, like the integration with other projects are studied. The chapter ends with some concluding remarks.
Chapter 2

Multimedia Representation and Description

In this chapter we discuss the technologies used to represent and describe multimedia with special emphasis on MPEG-4 and MPEG-7, the two key technologies used on this project. We also analyse other technologies that to a certain degree compete with the Moving Picture Experts Group (MPEG) standards [7]. The comparison between these technologies hopefully will make clear the reason why the choice leaned to MPEG-4 and MPEG-7.

2.1 Overview

Nowadays, the ever growing proliferation of consumer electronic products, such as digital photo cameras and digital video cameras, associated with the ease of plugging these devices to a computer, fast editing with inexpensive content editing software solutions, and sharing with others, using the many available conveyance means, makes each and every individual a potential content creator. In addition, the use of multimedia information is being generalised in every types of networks, with special emphasis on mobility [8]. Although the means for content creation are in the market, some problems arise when one wants to (re)use, create or share content: there are many media formats but usually with poor integration of multimedia content, in the sense that multimedia content is not restricted to framed video and audio; experiencing the content is not as universal as the end-user would probably expect, sometimes due to rigid content adaptation which can be time-consuming and inefficient. On the other hand, the user's expectations are high: enhanced content, rich-media, interoperability and content ubiquity, are some common buzzwords. At certain point, it was clear that a framework that could handle these problems was needed [9].
When the coding/compression algorithms began to sprout, the first video encoders were developed, based on a representation model of the vision consisting of a bidimensional and static (frame-based) representation of a sequence of images – the television paradigm. In fact, video objects recorded from different sources are sometimes blended together to produce what appears to be a single video scene. A typical example of this is the weather forecast, where the weather forecaster in Figure 3(a) and the background map in Figure 3(b) are separate objects. One is a natural video object recorded as shaped video, the other is a computer generated graphic. However, the resulting video stream is encoded with both elements together (Figure 3(c)) and the client has no possibility to separate one object from the other.

![Weather forecaster and background map](image)

(a) Weather forecaster  (b) Background map

![Resulting stream](image)

(c) Resulting stream

**Figure 3:** Typical example of different sources being blended - weather forecast program.

Moving forward from this approach, equally followed by the previous standards produced by the MPEG working group [10], the MPEG-4 standard adopts a new representation model based in the explicit representation of media objects in one scene. This new paradigm is close to the human relational mechanism with the surrounding environment: object assimilation and identification, allowing an interaction with that environment. This kind of representation has also improved flexibility in content creation allowing, besides the re-utilization of existing multimedia applications, new paths for multimedia applications. The MPEG-4 standard is the first audiovisual codification standard that is, at the same time, close to the human experience with real objects, and to the way the audiovisual content is created [11, 12].
These sources of information have an increasing demand for processing, which brings an additional need for alternate representations that go beyond the audiovisual information, even if this information is coded in a well organized object representation, which is the case of MPEG-4. Forms of representation that allow some kind of interpretation of the information meaning are therefore needed. To address the problem of quick and efficient searching, processing and filtering of various types of multimedia content, MPEG initiated another standard project – MPEG-7, or *Multimedia Content Description Interface*. The MPEG-7 standard defines the syntax and semantics of the description of audiovisual content [13, 14]. In complement to the MPEG-4 scene description, MPEG-7 defines to describe each object either visual (as for its shape, colour, texture, etc) or aural (as for its timbre, tone, melody, etc). It is also defined the association of context information allowing the definition of a description closer to the semantic level.

More recently, the MPEG-21 standard project, officially called *Multimedia Framework*, aims at understanding how the building blocks of a multimedia application infrastructure integrate, simultaneously discussing which new standards are needed to fill eventual gaps [15].

Even though multimedia representation and description are closely related, the next two sections give a separate overview of the technologies available for each purpose. Despite only mentioning, until now, MPEG standards, there are other ongoing activities that may to some extent overlap with these and that will also be analysed.

### 2.2 Representing Multimedia Content

In this section an overview of MPEG-4 as a technology for representing multimedia content is presented. Later in the chapter some competing activities, namely the specification efforts from World Wide Web Consortium (W3C) [16], SMIL and SVG, as well as the Web3D [17] specification X3D, are also analysed. These technologies are closely related to MPEG-4, specifically in providing a standardised format for the integration of media of different nature.

#### 2.2.1 MPEG-4

MPEG-4 is a standard (ISO/IEC 14496) for the composition of audiovisual scenes consisting of both natural and synthetic media. It allows the creation of interactive multimedia content, which can be streamed within a wide range of bit rates (from under 64kb/s to over 4Mb/s) in environments characterised by a high degree of variability and flexibility.
In short, the standard fulfills the following objectives [18]: efficient coding of audiovisual content in objects, either of natural or synthetic origin (Figure 4); composition of audiovisual scenes, through the description of groups of audiovisual objects; universal access through multiplexing and synchronization of the data associated with the objects, to transport over network channels with the appropriated Quality of Service (QoS); user interaction with the audiovisual scene. MPEG-4 achieves these objectives by providing standardised tools to support:

- **Coding** - represent units of audio, visual, or audiovisual content, called media objects; media objects can be of natural or synthetic origin.
- **Composition** - describe the composition of media objects to create compound media objects that form multimedia scenes.
- **Multiplex** - multiplex and synchronise the data associated with media objects to transport over network channels providing a QoS appropriate for the nature of the specific media objects.
- **Interaction** - interact with the scene locally or remotely, using a back channel.

![Figure 4: MPEG-4 object-based representation architecture](image)

The structure of the MPEG-4 standard consists of several parts (nineteen at the time of writing). Not all of these parts are already considered International Standard (IS). The ISs parts comprise: Systems [19], Visual [20], Audio [21], Conformance Testing [22], Reference Software [23], and Delivery Multimedia Integration Framework (DMIF) [24]. In the next subsections some features of MPEG-4 are analysed. However, only features relevant to multimedia authoring will be referred. Other parts, including coding of objects.
is not analysed.

**Typical System**

Although MPEG-4 was designed to be used in a wide range of environments [25], usually the content production and distribution follow a common path. A typical process, using MPEG-4, is depicted in Figure 5. The diagram identifies where the elements are positioned and how they interconnect in the workflow. Note, however, that this example does not cover all the possible MPEG-4 configurations. It is, as such, just a typical system.

![Diagram of Typical MPEG-4 System](image)

**Figure 5: Typical MPEG-4 system**

On one end of the system we find the authors that produce multimedia content, using the authoring tools of their preference. One part of this content can be captured in real-time. The other part consists of previously created and is called off-line content.
The proposed architecture by the standard allows the production of contents, organized in MPEG-4 programs, in a flexible way and with great re-usability. When a program is created, it is composed by one or more scenes. Each scene can then group information about the objects that compose it and information that allows the characterization of each object in time and in space. The next subsection explains content representation in more detail.

The process of content creation can be split in two stages: production and publishing. In the production stage, multimedia information is created, including multimedia content and control information. On the other hand, the publishing stage includes content adaptation to the imposed restrictions by, for example, the network or the end devices. The later subsection Adaptive Content how MPEG-4 implements this.

The content stored in servers is eventually made available by content providers. The way the content is exchanged between creators and providers, and how it is stored, depends on the purpose and the required flexibility for reusing it. From the servers to the end devices, streams of information with MPEG-4 content are transmitted, called Elementary Streams (ESs). Each MPEG-4 session can use several ESs, which can in turn re-assemble the information being transmitted in a synchronized and coherent way. When the content is produced in real-time, it can be delivered directly to the servers in inter-operable ESs.

As already mentioned, one of the main objectives of the MPEG-4 architecture is to promote the possibility of using a wide range of access conditions so that the content can be created only once and used in several environments. This objective is attained through the abstraction of the content from the delivery layer using a common, well-defined, interface. When the content is delivered, the points of inter-operability are the specific formats of individual ESs, and the steps defined by the interface that should be followed to access the ESs. In general, the implementation under this interface is not normalised by MPEG-4. This way the multimedia representation can remain completely independent of the way it is distributed.

Media Objects

Scenes are composed by many types of media objects organised hierarchically in a tree-like structure where its leaves are the primary media objects. Examples of primary media objects are: still images, video and audio. One possible scenario, where some of these objects are used, is a scene composed by a background image, by a person talking without any background and with an associated voice, and a background music. Its hierarchical representation will be something like Figure 6. A similar object hierarchy could have been
represented with the weather forecaster, shown previously in Figure 3.

![Scene Diagram]

**Figure 6: Hierarchical representation of a scene**

MPEG-4 defines a set of primary media objects, capable of representing natural and synthetic content, which can be either placed in a 2D or a 3D coordinate system. Besides the types referred previously, the MPEG-4 standard specifies also other types of media objects, such as:

- Text and graphics;
- Synthetic animated heads and associated text, used to synthesise voice and animate the head;
- Synthetic animated bodies, animated in a similar way; and
- Synthetic sounds.

A media object, in its coded form, consists of a set of descriptive elements that allow its manipulation in a multimedia scene, as well as streaming information, if needed. An important feature of the coded form is that, each object, can be represented independently of what surrounds it and the background. The coded representation is as efficient as possible, taking into account the desired features. Examples of these features are robustness in error-prone environments, easiness of object extraction and edition, or even the possibility of scalable representation of the object.

Along with the primary media objects, a scene also includes composite objects that group primary objects. Whereas the primary objects define the leaves in the scene hierarchical structure as depicted in Figure 6, the composite objects comprehend sub-trees. An example: the visual object associated with the person talking and the corresponding
MULTIMEDIA REPRESENTATION AND DESCRIPTION

voice are grouped, forming a new composed object containing visual and aural components of the person. Such grouping allows authors to assemble complex scenes and allows consumers to manipulate a coherent set of objects.

Composition and Interaction of Media Objects

The aggregation of media objects into a multimedia presentation needs a way to describe the spatial and temporal relations between each object. Hence, some form of scene description is required to, for example:

- Layout media objects in any coordinate system;
- Apply a set of transforms to change the geometric or acoustic form of an object;
- Group primary media objects into composite objects;
- Apply streaming data to media objects, in order to change its attributes (e.g. a sound, an object texture with movement, a synthetic face with animation parameters);
- Change, interactively, the user’s positions of viewing and hearing anywhere in the scene.

Interaction with the scene is not restricted to changing the viewing and hearing positions. In general, the user observes the scene composed according to the author’s definitions, but depending on the degree of freedom allowed by the author, the user may interact with the scene. Some of the expected interactions with the scene include:

- Changing the position of observation/hearing, for example, through scene navigation;
- Dragging objects to different positions;
- Triggering events through a click in an object, for example, starting or stopping a video; and
- Selecting the desired language when multiple languages are available (audio or subtitles).

MPEG-4 provides a standardised way to describe a scene composed by media objects - **BInary Format for Scenes (BIFS)**. BIFS also provides the functionality that allows the interaction with the composing objects in a scene [26]. This format for scene description is based in several concepts of the standards for 3D graphics, Virtual Reality Modeling Language (VRML) [27], such as its structure and object composition functionality. To overcome some limitations of VRML, the scene description structure in MPEG-4 extends it to fulfill all features described above.
In MPEG-4, objects are represented as nodes and their interactive behaviors described using routing mechanism that associates an event source with an event sink. **Object Descriptors (ODs)** are used to associate scene description components to the actual elementary streams that contain the corresponding coded data. This will be further discussed later in this chapter. A mechanism that allows a scene to be remotely manipulated is also provided. BIIFS update commands serve as remote scene manipulators and also allow portions of the scene to be progressively streamed in order to reduce bandwidth requirements.

To provide an exchangeable format among content authors and at the same time preserve the author’s intentions, the **eXtensible MPEG-4 Textual format (XMT)** has been designed. It is a high-level textual format that provides an abstraction of the underlying MPEG-4 technologies.

The XMT framework consists of two levels of textual syntax and semantics: the XMT-A format and the XMT-Ω format.

The XMT-A is an XML-based format of MPEG-4 content, closely related to its binary representation. It provides a one-to-one mapping to the binary representations, including MPEG-4 specific features such as Object Descriptors (ODs), BIIFS update commands and 2D/3D composition. XMT-A was also designed to be interoperable with X3D, the evolution of VRML (will be discussed later in this chapter), containing actually a subset of X3D.

The XMT-Ω is a high-level abstraction of MPEG-4 features based on SMIL, that will also be described later. For every XMT-Ω element, there is a mapping to a sequence of XMT-A elements. However, there is no deterministic mapping between the two levels, for the obvious reason that a high-level author’s intentions can be expanded to more than one sequence of low-level constructs. Nevertheless, a standard mapping from XMT-Ω to XMT-A is provided. An example [28]: the next description uses the XMT-Ω language to show a rectangle whose color changes over a 6 second duration, beginning when the mouse button is pressed down.

```xml
<rectangle id="Square" size="50 50">
  <transformation visibility="true" translation="40 75"/>
  <material color="#ee0000" filled="true">
    <animateColor attributeName="color">
      <duration value="6s" begin="Square.click"/>
      <values value="#ee0000; #ffcc45; #ffffff"/>
      <keyTimes value="0; 0.3; 1" calcMode="linear"/>
    </animateColor>
  </material>
</rectangle>
```
MULTIMEDIA REPRESENTATION AND DESCRIPTION

Using XMT-A syntax, the above example could be mapped to MPEG-4 nodes and routes as follows. The necessary OD framework elements, such as InitialObjectDescriptor have been omitted for clarity.

```xml
<Replace>
  <Scene>
    <OrderedGroup>
      <children>
        <Switch whichChoice="0">
          <choice>
            <Transform2D translation="40 75">
              <children>
                <Shape>
                  <appearance>
                    <Appearance>
                      <material>
                        <Material2D DEF='SquareMat'
                          emissiveColor="0.93 0.0 0.0"
                          filled="TRUE"/>
                      </material>
                    </appearance>
                    <geometry>
                      <Rectangle size="50 500"/>
                    </geometry>
                  </shape>
                </TouchSensor DEF="Touch"/>
                <TimeSensor DEF="Timer" cycleInterval="6"/>
                <ColorInterpolator DEF="Coloring"
                  key="0.0 0.3 1.0"
                  keyValue="0.93 0.0 0.0, 1.0 0.93 0.27, 1.0 1.0 1.0"/>
              </children>
            </Transform2D>
          </choice>
        </Switch>
      </children>
    </OrderedGroup>
  </Scene>
</Replace>
```

<ROUTE fromNode="Touch"
  fromField="touchTime"
  toNode="Timer"
  toField="startTime"/>

<ROUTE fromNode="Timer"
  fromField="fraction_changed"
  toNode="Coloring"
  toField="set_fraction"/>

<ROUTE fromNode="Coloring"
  fromField="value_changed"
The XMT also provides an escape mechanism from XMT-Ω to XMT-A. The escape mechanism enables content authors to mix and match the two formats, XMT-Ω and XMT-A, overriding the standard mapping XMT provides. An evident parallelism with programming languages arises. While BIFS can be compared with binary machine-code, XMT-A can be compared with assembly language and XMT-Ω can be compared with structured languages like C.

Description and Synchronisation of Media Objects

The media objects may require streaming data, which can be transported in one or more ESs. An object descriptor object descriptor is used to identify all streams associated with the media object media object. Hence, each individual media object is delivered within its own Elementary Streams (ESs) and associated with the scene command stream through OD streams as depicted in Figure 7.

The OD stream carries instructions for the decoding of the media object, its synchronisation within the scene and facilities for scaling the object in response to bandwidth conditions. Each stream associated with media objects is, itself, characterised by a set of descriptors with configuration data. These descriptors are used, for example, to determine which decoder resources and precision of the timing information of the coder are necessary. The descriptors can also transport information about the QoS necessary for the transmission (e.g. maximum bit rate, error rate, priority, etc).

Besides this information, the media objects associated with elementary stream data through an object descriptor can as well have additional information about the content. This information is called Object Content Information (OCI). It is also possible to associate information about intellectual property and rights with Intellectual Property Management and Protection (IPMP).

ES synchronisation is achieved through the temporal identification of units of data, called individual Access Units (AUs). The Synchronisation Layer (SL) manages the identification of the AU and the timing marks. Independently of the information type, the SL allows to: identify the type of AU (e.g. video or audio frames, scene description commands) in a ES; recover the temporal base of media objects and of the scene description;
and establish synchronisation between access units. The syntax of this layer is configurable in several ways, allowing it to be used in wide range of systems. The interaction of the SL with other layers in the MPEG-4 model is described next.

**Adaptive Content**

As we have seen in the last two subsections, objects in a BIFS scene can be manipulated individually through streamed scene commands. This offers the possibility of extending media adaptation into video.

As previously stated, **Binary Format for Scenes (BIFS)** allows the composition of multiple arbitrarily shaped video objects in an audiovisual scene together with synthetic images, producing what appears to be a single video scene. This scene is, however, encoded as a multitude of video objects in separate streams with individual decoding instructions.

Scene commands can remove and insert media objects, or change the object descriptor stream that determines the decoding of an object. A system could alter dynamically those decoding instructions, for example, to adapt video to different networks and devices. Or it can change the inclusion and layout of the scene's objects accordingly to user preference or interaction.
Scenes may also contain other types of objects, such as audio and images (also encoded in their own streams with object descriptors) or text and graphics, which can be described inline in the scene description. Inline scene objects have properties which can be altered by scene commands, allowing dynamic inserting and altering of computer-generated text or graphics in response to interaction or content changes.

**Content Delivery**

The synchronised delivery of streaming data from source to destiny, taking into account different QoSs available in the network, is specified in a Synchronisation Layer (SL) and a delivery layer. The last is composed by a two-layer multiplexer, as shown in Figure 8.

![Diagram](image)

**Figure 8: MPEG-4 systems layer model [29]**

The first multiplexing layer is managed accordingly to the specification of the Delivery Multimedia Integration Framework (DMIF). This multiplexer can be complemented by a tool also defined by the MPEG-4 standard – **FlexMux**. This tool allows the grouping of ESs with a small additional overhead in multiplexing. The multiplexing on this layer can be used, for example, to group ESs with similar QoS requirements, reducing the network connections or the end-to-end delay.

The **Transport Multiplex** (TransMux) layer, models the layer that offers transport services compatible with the QoS requirements. Only the interface is specified by MPEG-4. The concrete mapping of data packets and control signalization is done specifically to each transport protocol. Any adequate transport protocol, such as (RTP)/UDP/IP, (AAL5)/ATM or MPEG-2 Transport Stream over an appropriate data link layer, can be a TransMux instance. The choice should be made by the end user or service provider,
allowing MPEG-4 to be used in a great variety of operational environments. As shown in Figure 8, FlexMux multiplexing tool is optional if the lower layer TransMux provides all required functionalities. However, the SL is always present.

Although the layer model specified by MPEG-4 can handle delivery of content, a format that can be used for off-line content storage and interchange in a flexible and protocol-independent way is needed. With that in mind, the MP4 file format was designed. It is able to contain media information for interchange, management, editing and presentation of the media. This presentation may be local to the system or may be instead delivered through a network or a stream delivery mechanism. The file structure is object-oriented, allowing it to be decomposed into constituent objects, with the structure of objects being directly inferred from their type. The MP4 format is defined as non-framing because the timing and structural data is stored separately from the media data, in compact tables. Moreover, the media data is stored in its natural or base state, not preferring one transport protocol or system over another.

Streaming of media data is also possible using the MP4 file format. Special instructions for the streaming protocol called hints can optionally be stored in the file. These assist the streaming servers in the formation of packets for transmission.

2.2.2 Synchronized Multimedia Integration Language (SMIL)

SMIL is a W3C recommendation [30] that specifies a format for integrating independent multimedia objects into a single multimedia presentation, with coherent temporal and spatial attributes. It uses a XML-based language used by authors to develop interactive multimedia presentations. The specification of SMIL is divided into ten functional areas: animation, content control, layout, linking, media objects, meta-information, structure, timing and synchronization, time manipulations and transition effects. Each functional area is itself divided into modules. A module is a set of elements, attributes and values that form an atomic set of tools to achieve a certain functionality.

SMIL provides to the multimedia author techniques for:
- temporal composition (synchronization of different media),
- spatial composition (layout), and
- association of hyperlinks with media objects.

The author is able to create a multimedia presentation, describing the composition of media objects. These media objects can be audio, video, still pictures, still text, text streams and animations. The presentation can be adapted to the settings of the user. Like all XML-based technologies, SMIL benefits from scripting, style sheets and linking.
However, no update mechanism like MPEG-4’s update commands exists. Hence, SMIL is not natively stream-oriented. Relying on the extendability of XML, the SMIL specification is not a complete multimedia description format. In fact, it does not define how the different types of media fit in its architecture. If, for example, one wants to have formatted text, the combined SMIL and XHTML specifications are needed. Or if 2D graphics are to be implemented with SMIL, SVG needs to be combined with SMIL. A specification that tells how to use a 3D XML language, like X3D in conjunction with SMIL, should also be required for the use of 3D content. This could lead to several interoperability issues. Moreover, the synchronization of media in SMIL is provided by references to external players – no decoding model is specified. Also, SMIL does not provide any means of compression nor of encryption of the scene description.

2.2.3 Scalable Vector Graphics (SVG)

SVG is a W3C recommendation [31] that defines an XML-based language for describing two-dimensional vector and mixed vector/raster graphics. SVG allows for three types of graphic objects

- vector graphic shapes, e.g., paths consisting of straight lines and curves (bézier or elliptical arcs);
- images; and
- text.

In addition to that, graphical objects can be grouped, styled, transformed and composited into previously rendered objects. The feature set includes nested transformations, clipping paths, alpha masks, filter effects and template objects. SVG drawings can be interactive (simple interaction based on pointing devices) and dynamic (using deterministic animations). Animations can be defined and triggered either declaratively (i.e. by embedding SVG animation elements in SVG content) or via scripting. In both cases, the animation is entirely contained in the SVG document and known at loading time.

Being based on the same technology and specified by the same specification body, SVG has roughly the same benefits and handicaps as SMIL. The extendability and the supporting techniques (scripting, style sheets, linking, etc) of XML are interesting advantages. Some initiatives like the specification of profiles for mobile terminals are also of special interest. However, the lack of an update mechanism and of streaming-oriented design are severe disadvantages. In fact, the initial delay to get the whole structure and graphics can become cumbersome. It is unlikely that a user will be willing to wait as much time to download a cartoon as the time to watch it, while it could be streamed and viewed.
MULTIMEDIA REPRESENTATION AND DESCRIPTION

simultaneously.

Another problem is that the scripting in SVG requires the use of Document Object Model (DOM) and another script language, which can increase dramatically the memory footprint and the complexity of the implementation.

As referred before, SVG can be used together with SMIL, benefiting from the SMIL animation module to have complex animation. In a more complex system, it is also possible to use SVG with SMIL and XHTML allowing mixed graphics and formatted text.

2.2.4 eXtensible 3D (X3D)

X3D is the current specification effort of the Web3D consortium that refers to X3D as "an extensible open file format standard for 3D visual effects, behavioral modelling and interaction". It is oriented to enable the deployment of 3D graphics in applications ranging from lightweight web clients to high-performance 3D broadcast solutions. Like XMT, SMIL and SVG referred previously, X3D provides an XML-encoded scene graph. It is also specified a language-neutral interface called Scene Authoring Interface (SAI). The SAI allows 3D content and controls to be integrated into a web and non-web applications.

X3D is the evolution of the VRML standard [?] focusing on more compact implementations and improved interoperability. VRML is a textual format for describing interactive 3D objects and worlds and intended to be a universal interchange format for integrated 3D graphics and multimedia. VRML is capable of representing static and animated 3D and multimedia objects with hyperlinks to other media such as text, sounds, movies, and images. To achieve authorability, VRML was designed as a textual format and with the ability to compose files together through inclusion, to relate files together through hyperlinking and to easily reuse complex contents through the use of author-defined macros.

The main added features of X3D to VRML are the use of XML as textual format for representing 3D worlds and the addition of 22 new nodes keeping X3D fully compatible with VRML standard.

Despite being thought to also be used in the Internet, VRML has not reached the expected success. One probable cause is the fact that VRML does not support dynamic composition – no server-side interaction is handled. Another problem is the absence of a streaming support, particularly notorious by the fact that a large number of primitives are usually needed for a smooth and realistic presentation. The only way to reduce delay is to zip the VRML file prior to sending it over the Internet. VRML also does not offer any 2D primitive nor the possibility to mix 2D and 3D worlds, and is not able insure fine synchronization on a per frame basis. Finally, VRML does not provide hooks to enable
encryption, watermarking and digital rights management of 3D content.

2.2.5 Flash

Flash is a proprietary solution for vector-based graphics to produce multimedia content. Currently it is the leading technology to design 2D graphics, animations and more recently 3D graphics. Flash uses an efficient publishing binary format called SWF (pronounced "swif"). Although the format is proprietary, it is accessible to the public domain [32] and tools for editing or conversion are available. In fact, some projects propose conversion solutions from Flash to MPEG-4 [33]. Flash’s binary format uses a concept similar to the QuickTime (and MPEG-4 binary format) atoms. Each atom is composed of a tag, length and payload. Along with the publishing format, another binary format is used for editing. However, unlike the SWF format, this format is not public.

The graphical primitives used in Flash are similar to the ones used in SVG and MPEG-4. The difference between these formats is the way the primitives are organised, reused or modified. The main graphic primitive in Flash is the path. The path is composed by a set of segments of many available types (e.g. straight line, spline, bézier curve, etc). Many of the basic 2D elements are mapped to paths.

The main reason for Flash’s popularity is its proximity with the techniques used by graphics designers. A display list, similar to an exposure sheet used by a cartoon designer is also used by Flash. It is also possible to define and reuse primitives. However, a common problem occurs when the amount of content increases and specific tools are needed to perform filtering and content adaptation.

Despite its success, Flash has some limitations. It does not support Digital Rights Management, it does not have real-time stream control, and is mainly oriented to Web applications – transported using only HTTP and it is not broadcast-capable.

2.2.6 Comparison

MPEG-4 combines some typical features of other MPEG standards, but aims to provide a set of technologies to satisfy the needs of:

- **Authors**, enabling the production of content with greater re-usability and flexibility than possible with today’s individual technologies such as digital television, animated graphics, Web pages, and their extensions. Also, it permits better management and protection of content owner rights.

- **Network service providers**, offering transparent information, interpreted and translated into the appropriate native signaling messages of each network with the
MULTIMEDIA REPRESENTATION AND DESCRIPTION

help of relevant standardisation bodies. However, the preceding excludes QoS considerations, for which MPEG-4 will provide a generic QoS descriptor for different MPEG-4 media. The exact translations from the QoS parameters set for each media to the network QoS exceed the scope of MPEG-4 and remain for network providers to define.

- **End users**, enabling many functionalities potentially accessible on a single compact terminal and higher levels of interaction with content, within the limits set by the author. One example of this is multimedia games [34].

Hence, MPEG-4 provides a single form of compression usable by all players. It supports graphics, text, animation and digital rights management in an object-based framework. Content can be created once and delivered to broadcast, cable, HDTV, DVD, Internet, mobile devices or PC.

SMIL was defined as an extension of XML to specifically address the issue of presenting content comprising several independent media objects. Some major differences distinguish SMIL and MPEG-4. SMIL:

- assumes media objects are in separate URLs;
- specifies synchronisation at the stream level (beginning and ending of a stream, leaving the fine-grain synchronisation to the browser implementation); and
- provides no explicit support for 3D objects.

![Diagram of multimedia representation](image)

**Figure 9:** Relation between MPEG-4, SMIL and X3D (based on [29])

X3D is an evolution of the VRML standard which was used as a base for the MPEG-4 scene description language. Additionally, XMT was designed in cooperation with the X3D team enabling interoperability. This close relation allows almost full compatibility between the two standards. However, all the previously referred problems in VRML are handled in MPEG-4. For example, while most implementations of VRML browsers assume that audio and video are first downloaded to the client terminal and then played, MPEG-4
integrates streaming of scenes and continuous media embedded in the scenes. X3D can be considered a subset of XMT as shown in Figure 9. Also SMIL and SVG are closely related to the MPEG-4's textual scene description format.

Flash is the de facto standard for multimedia presentations in the Web. Although it is proprietary technology, some of its specifications are public. Some disadvantages when compared to MPEG-4 are noticeable, mainly concerning the integration with environments other than Internet. As well as SMIL, SVG and X3D, Flash is focused on Web applications. Another difference is the absence of a per frame synchronisation of media elements.

As a summary, and taking into account several characteristics of multimedia representation, Table 1 presents a comparative analysis of the technologies referred in this section.
<table>
<thead>
<tr>
<th>Feature Description</th>
<th>VRML</th>
<th>BIFS Version 4</th>
<th>SMIL</th>
<th>SVG</th>
<th>Flash</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spatial and temporal composition of text,</td>
<td>3D composition of 3D graphics, text,</td>
<td>2D and/or 3D composition of simple and</td>
<td>2D composition only</td>
<td>2D graphics, text and images only. No</td>
<td>2D graphics, text and images only. No support for video or audio.</td>
</tr>
<tr>
<td>graphics, images, videos and sounds</td>
<td>video, sound and images</td>
<td>complex scenes</td>
<td></td>
<td>support for video or audio.</td>
<td></td>
</tr>
<tr>
<td>Composition of media coming from several</td>
<td>Supported but without any decoding</td>
<td>Supported but without any decoding model to</td>
<td>Supported only for SVG remote media</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sources</td>
<td>source without any decoding model to</td>
<td>insure synchronisation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Animation</td>
<td>VRML-like animation and adds the</td>
<td>Only in non-streaming. All anim. parameters are</td>
<td>Uses the SMIL Animation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>mechanisms of BIFS-Anim</td>
<td>known at time 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compression</td>
<td>Zip compression</td>
<td>Native binary format</td>
<td>Not supported</td>
<td>Zip compression</td>
<td>Native compression</td>
</tr>
<tr>
<td>Streaming facility</td>
<td>Not supported</td>
<td>Designed around this central point</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Fully supported</td>
</tr>
<tr>
<td>Dynamic composition</td>
<td>Not supported</td>
<td>Supported through the update mechanism</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Fully supported</td>
</tr>
<tr>
<td>Digital Rights Management</td>
<td>Not supported</td>
<td>MPEG-4 Systems in general provides hooks for</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
<tr>
<td>Integration with other environments (Broadcast, TV)</td>
<td>Restricted to Web environment</td>
<td>Supported through the mapping of MPEG-4 over</td>
<td>Restricted to Web applications</td>
<td>Restricted to Web applications</td>
<td>Restricted to Web applications</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MPEG-2 and IP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Authoring tools and players</td>
<td>Lots of software available on the market</td>
<td>Some tools are available</td>
<td>Some tools are available</td>
<td>Some tools are available</td>
<td>Many authoring tools and players exist</td>
</tr>
<tr>
<td>Feature</td>
<td>XHTML</td>
<td>ODF</td>
<td>XSL-FO</td>
<td>XFL</td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------</td>
<td>-------</td>
<td>--------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>Fine Synchronization between the media</td>
<td>Not supported</td>
<td>Enables synchronization of several medias on a per frame basis</td>
<td>Not supported</td>
<td>Not supported</td>
<td></td>
</tr>
<tr>
<td>Scripting facilities</td>
<td>Supported by means of Javascript</td>
<td>Supported by means of Javascript and Java</td>
<td>Supported through DOM interface</td>
<td>Supported through DOM interface</td>
<td>Supported via Action-Scripts</td>
</tr>
<tr>
<td>Independence of media source from scene descr.</td>
<td>Not supported, reference to the media are made directly in the text</td>
<td>Fully supported thanks to the ODF</td>
<td>Not supported, reference to the media are made directly in the text</td>
<td>Not supported</td>
<td></td>
</tr>
<tr>
<td>Broadcast Carousel</td>
<td>Not supported</td>
<td>Fully supported</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
<tr>
<td>Error Resilience</td>
<td>Not Supported</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
<tr>
<td>Formatted Text</td>
<td>Not supported</td>
<td>Under development</td>
<td>Through the use of XHTML</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
<tr>
<td>Decorated Text</td>
<td>Not supported</td>
<td>Under development</td>
<td>Not supported</td>
<td>Fully supported</td>
<td>Fully supported</td>
</tr>
<tr>
<td>Filtering Effects</td>
<td>Not supported</td>
<td>Partially supported</td>
<td>Not fully supported</td>
<td>Fully supported</td>
<td>Fully supported</td>
</tr>
<tr>
<td>2D Graphics</td>
<td>Not supported</td>
<td>Fully supported</td>
<td>Through the use of SVG</td>
<td>Fully supported</td>
<td>Fully supported</td>
</tr>
<tr>
<td>Interactivity</td>
<td>Supported for pointing devices</td>
<td>Fully supported</td>
<td>Supported for DOM events</td>
<td>Supported for DOM events</td>
<td>Fully supported</td>
</tr>
<tr>
<td>3D Graphics</td>
<td>Fully supported</td>
<td>Fully supported</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Under development</td>
</tr>
</tbody>
</table>

**Table 1:** Comparison between multimedia description languages  
*(based on [35])*
2.3 Describing Multimedia Content

To conclude this chapter, this section gives an overview of MPEG-7 and other metadata-related initiatives, namely Dublin Core and RDF. Many other content description standards are available, however, these are essentially application-targeted standards unlike MPEG-7, Dublin Core and RDF. Other standard also referred is AAF in spite of having a scope somewhat different from the previous standards. However, some inter-connecting points are obvious in the sense that it also aims at describing media from different sources.

2.3.1 MPEG-7

The MPEG-7 International Standard (ISO/IEC 15938), formally known as Multimedia Content Description Interface provides standardised technologies, for the description of audiovisual data content in multimedia environments. The multimedia content to which MPEG-7 descriptions can be associated, may include: still pictures, video, audio, speech, graphics, 3D models and composition information about how these elements are combined in a multimedia presentation (scenarios).

The MPEG-7 descriptions do not depend on how the described content is coded or stored. It is possible to create an MPEG-7 description of an analogue movie or of a picture that is printed on paper, in the same way as of digitized content.

Although MPEG-7 is independent of the coded representation, if the multimedia content is encoded with MPEG-4, the information already stored in the MPEG-4 material enhances the description scope. The inherent description of temporal and spatial relations between objects in MPEG-4 allows the attachment of descriptions to individual objects within the scene. Therefore, each object consists of its information (encoded in MPEG-4 Visual, AVC, JPEG, PNG, MP3, AAC, etc) and the description about its content or metadata (encoded in MPEG-7). These objects can then be combined together in a multimedia presentation with MPEG-4.

It is possible to have different granularity in MPEG-7 descriptions, providing the ability to discriminate within different levels. The description can be composed by a set of features. “A feature is a distinctive characteristic of the data that signifies something to somebody” [13]. The features can be divided in two types: low-level and high-level. Features like shape, size, texture, color, movement (trajectory) for visual content and key, mood, tempo, tempo changes, position in sound space for aural content are examples of low-level features. High-level features usually have a semantic meaning, e.g. participants, genre classification, rating or location.
Low-level features have three important characteristics:

- can be extracted automatically;
- are objective; and
- are native to audiovisual content.

This means that the great amount of low-level features that could be associated with the content can be extracted automatically by machine learning algorithms and tools. The work of describing existing and new content is eased, doing in a short period time what would otherwise take several months or even be completely impossible to perform if done by humans. Moreover, the objectivity and the native nature of the features allow queries to be formulated in a way more adequate to the content in question.

Nevertheless, searching at semantic level is surely interesting to users but automatic extraction of high-level features is a far more complex task. MPEG-7 allows the combination of both low-level and high-level features is a single description.

In short, the MPEG-7 descriptions of content may include [36]:

- information describing the creation and production processes of the content (director, title, short feature movie);
- information related to the usage of the content (copyright pointers, broadcast schedule);
- information of the storage features of the content (storage format, encoding);
- structural information on spatial, temporal or spatio-temporal components of the content (scene cuts, segmentation in regions, region motion tracking);
- information about low level features in the content (colors, textures, sound timbres, melody description);
- conceptual information of the reality captured by the content (objects and events, interactions among objects);
- information about how to browse the content in an efficient way (summaries, variations, spatial and frequency sub-bands, ...);
- information about collections of objects; and
- information about the interaction of the user with the content (user preferences, usage history).

The MPEG-7 standard consists of several parts, including Systems[37], DDL[38], Visual[39], Audio[40] and MDS[41]. The descriptors specified in the Visual and Audio parts are essentially related to low-level features and thus will not be referred in this dissertation. The next subsections will provide further features of MPEG-7, primarily focusing on the Systems and MDS parts.
Description Tools

MPEG-7 approaches the description of content from several viewpoints. The sets of Description Tools developed on those viewpoints are presented here as separate entities. However, they are interrelated and can be combined in many ways. Depending on the application, some will be present and others can be absent or only partly present.

![Diagram showing the relationship between Description Definition Language, Tags, Structuring, and Encoding and Delivery]

**Figure 10: MPEG-7 main elements [42]**

The main elements of the MPEG-7’s standard are:

- **Description Tools:**
  - **Descriptors (Ds)** – define the syntax and the semantics of each feature; and
  - **Description Schemes (DSs)** – specify the structure and semantics of the relationships between their components, that may be both Descriptors and Description Schemes;

- **Description Definition Language (DDL)** – defines the syntax of the MPEG-7 Description Tools, allows the creation of new Description Schemes (DSs) and, possibly, Descriptors (Ds) and allows the extension and modification of existing DSs; and

- **System Tools [37]** – support for:
  - binary coded representation allowing efficient storage and transmission;
  - transmission mechanisms (both for textual and binary formats);
  - multiplexing of descriptions;
- synchronization of descriptions with content;
- management and protection of intellectual property in MPEG-7 descriptions;
- etc.

Figure 10 shows the relationship among the different MPEG-7 elements described above.

The creation of descriptions, using the Description Tools provided by MPEG-7 resumes itself to instantiate a set of DSs and their corresponding Ds, chosen by the user and that best fits the application. If application specific extensions are necessary, they can be declared using the Description Definition Language (DDL). Finally, the deployment of the descriptions are performed using the System Tools.

MPEG-7 data may be physically located with the associated AV material, in the same data stream, or on the same storage system. However, descriptions can also be somewhere else on the globe. When the content and its descriptions are not co-located, mechanisms that link the multimedia material and their MPEG-7 descriptions are needed; these links will have to work in both directions.

**Multimedia Description Scheme (MDS)**

The MPEG-7 Part 5 or Multimedia Description Scheme (MDS) specifies a metadata system for describing multimedia content, using description tools. The MDS description tools are organized on the basis of functionality as shown in Figure 11.

The Basic Elements (shown at the lower level) form the building blocks for the higher-level description tools. The following basic elements are defined: schema tools (the root element, top-level types, description metadata, and packages), basic datatypes (integers, reals, vectors, matrices), linking and media localization tools (spatial and temporal localization) and basic description tools (language, text, classification schemes).

The content description tools (shown at the middle level) describe the features of the multimedia content and the immutable metadata related to the multimedia content. The following description tools for Content Description are defined: structure description tools (spatio-temporal segments of multimedia content) and semantic description tools (objects, events). Content metadata are defined: media description (storage format, encoding), creation and production (title, creator, classification), and usage (access rights, publication).

The tools for Navigation and Access (shown at the middle level) describe the browsing, summarization, and access of content.

The tools for Content Organization (shown at the top level) describe collections and models of multimedia content.
Finally, the tools for *User Interaction* (shown on the right) describe user preferences pertaining to consumption of multimedia and usage history.

### 2.3.2 Dublin Core (DC)

Dublin Core (DC) is a metadata set designed to promote discovery of electronic resources. Its development and refinement is assured by the Dublin Core Metadata Initiative (DCMI) [43]. The element set provided by DC allows simple and flexible description of documents, images, sound files, and other networked information objects.

The first objective of the DC was to enhance searching of document-like objects on the Web. Firstly, 12 descriptive elements common to most Web documents were identified: Title, Author or Creator, Subject and Keywords, Description, Publisher, Other Contributor, Date, Resource Type, Format, Resource Identifier, Source, and Language. The DC elements were designed to describe works generated by a wide variety of intellectual disciplines and in a number of formats; in addition to text, the element set applied to graphics, sound, and video files. Three more elements, Relation, Coverage, and Rights Management, were added later to enhance description of images.

The level of description provided by the Dublin Core is not exhaustive. The 15 DC elements are general in nature. This generality gives the Dublin Core its flexibility to
describe different resources. However, this same attribute limits the DC’s ability to describe a work at more than a basic level of detail. When details are needed, such as e-mail address of the author, the simple element field can grow long.

Unlike regulated metadata formats, the Dublin Core element values use natural language. The advantage of using natural language is that authors/creators can describe their works using language and formatting appropriate to their respective disciplines. Natural language values, however, can pose serious problems for search engines, namely different forms of expressing the same thing and the need to parse through long text strings looking for pertinent information.

Other problem arises with the description of collections and substitutes. Data describing a collection is linked to members of the collection through the Relation element. A search engine attempting to group the elements together will have difficulty identifying both the collection and all of its members. Describing the relationship between substitutes is also difficult.

To overcome some of these problems, a series of qualifiers have been proposed to refine the core element set. There are two proposed types of qualifiers: schemes and types. Whereas schemes describe the syntax used by element values, types refine the core element itself. Two limitations have been set on qualifiers: a qualifier can only refine an element, not re-define its semantics; the content must still be understood if the element is used without qualifiers. Qualifiers allow applications to increase the specificity or precision of the metadata but can introduce complexity that could impair the metadata’s compatibility with other DC software applications.

2.3.3 Resource Description Framework (RDF)

Resource Description Framework (RDF) is a W3C recommendation [44] to provide interoperability between applications that exchange information on the Web. The general goal is to define a mechanism to describe resources that makes no assumptions about a particular application domain, nor defines (a priori) the semantics of any application domain. It allows descriptions of Web resources to be made available in machine understandable form, enabling the semantics of objects to be expressible and exploitable.

RDF is based on a concrete formal model using directed graphs that refer to the semantics of resource description. The entity being described, named Resource (e.g. electronic documents or parts of them, a complete Web site, or even books) and identified by an Uniform Resource Identifier (URI) is described through a collection of Properties. This description is accomplished by Statements which are triples that include: a node
representing the resource (identified by its URI), arc nodes representing properties (such as Owner Name) and other node pointed by the arc which represents the value of that property for the given resource.

The characteristics of Properties are defined by RDF-Schemas and include: permitted values, relationship with other properties, applicability and, indirectly, its meaning. In basic RDF, properties are modeled by simple value pairs and the definition of properties and dependencies among them are not considered. RDF-Schema defines a schema language based on RDF primitives – can be seen as a simple type system.

Although frequently expressed in terms of XML, the concepts of RDF and RDF-Schema were designed independently of a language.

2.3.4 Advanced Authoring Format (AAF)

Advanced Authoring Format (AAF) covers exchange of essence as well as it’s associated metadata [45]. It is mainly designed for the post production and authoring environment. AAF solves the problem of multi-vendor, cross-platform interoperability for computer-based digital production. A vast range of metadata categories are specified, including:

- Identification and location – how the item is uniquely identified.
- Administration – rights, access, encryption and security, etc.
- Interpretive – names, artists, etc.
- Parametric – signal coding and device characteristics.
- Process – editing and compositing data.
- Relational – relation between various pieces of metadata or essence.
- Spatio-Temporal – places, times, things, camera angles, etc.

The format allows complex relationships to be described in terms of an object model. It also facilitates the interchange of metadata or program content. AAF is not restricted to the definition a metadata set, it also provides a way to wrap all elements of a project together for archiving.

Material eXchange Format (MXF) can be described as a subset of the broader AAF with a vertical definition around the exchange of material rather than exchange of authoring information [46]. MXF was recently standardised by Society of Motion Picture and Television Engineers (SMPTE)[47]. It focus on simplicity for the exchange of essence (both video and audio) as well as metadata belonging to the essence. MXF is often described as the digital equivalent of tape. MXF can be seen as an end of the chain wrapper, designed for completed content which will not suffer many changes. This is especially important in a broadcast environment or for broadband content delivery.
2.3.5 Comparison

MPEG-7 uses XML as the language of choice for the textual representation of content description. XML Schema has been the base for the DDL, which is used for the syntactic definition of MPEG-7 Description Tools as well as for its extensibility (either new MPEG-7 ones or application specific). Considering the popularity of XML, its usage may facilitate interoperability with other metadata.

Dublin Core (DC) is a generic description standard intended to ease the discovery of electronic resources. It is conceived for author-generated description of Web resources and has specific target groups: museums, libraries and organisations. However, its generality may become a problem for more complex system as a multimedia authoring framework.

MPEG-7 comprises DC as it supports specification of the same types of metadata: semantic content, context and structure. It also supports specification of production and presentation of media objects in a wider and more detailed level. One difference between DC and MPEG-7 is the focus of the proposed descriptions toward descriptive and semantic content, respectively.

Besides DC, MPEG-7 is designed to take into account also other the viewpoints under consideration by other more specific metadata standards such as, among others, TV Anytime, SMPTE Metadata Dictionary, and EBU P/Meta. These standardisation activities are focused to more specific applications or application domains, whilst MPEG-7 has been developed as generic as possible.

Resource Description Framework (RDF) is an infrastructure that enables encoding, exchange and reuse for structured metadata. It does not define semantics for each resource description community, providing the ability for these communities to define metadata elements as needed.

One problem with RDF is to create a generic property such as contains by which the hierarchical structure can be defined, i.e., a video contains sequences, which contain shots, which contain frames, which contain objects and actors. Since each property requires a single range, then generic relationships such as contains can not be used. The lack of class-specific constraints on domain and range of properties is a major limitation of RDF, particularly when applied to complex multi-layered documents in which you want to specify constraints on structural, spatial, temporal and conceptual relationships between components.

AAF is mainly designed for a broadcast environment, providing a way to wrap both the data and the metadata. Although it is not as close to MPEG-7 as the other metadata
standards referred above, AAF (and MXF) should be taken into account and interoperability is important. A possible scenario where interoperability is possible is, for example, a scenario where a content producer creating MPEG-4 content and describing it with MPEG-7 descriptors, stores or exchanges it with a content provider using AAF.

In conclusion, Figure 12 shows the relationship between the description standards described in this section.

Figure 12: Relationship between content description standards
Chapter 3

Multimedia Authoring

In a somewhat simplistic definition, multimedia authoring is the assembly of multimedia elements in a presentation. This assembly is done using, possibly, a high-level graphical interface or a high-level scripting. In this chapter we define the authoring paradigms or authoring metaphors of an authoring system and its requirements. Finally, a brief overview evaluates existing authoring tools using different technologies and paradigms.

3.1 Overview

A multimedia presentation is composed by wide range of media items, combined together to provide coherent and appealing content to the consumer. A media item is therefore the data associated with a single playable object in the multimedia presentation, e.g. an image, a video object, a sound or text. Creating a multimedia presentation consists of three tasks:

- creating and editing the media items comprising the presentation;
- assembling the items into a coherent presentation, including the specification of the temporal and spatial layout of the items;
- specifying the interaction between the user and the presentation.

The first task requires the use of specific data editors, specialised in a range of media types. The authoring system described in this dissertation is not primarily intended to support this task, thus we will be focusing on the remaining tasks — assembly and interaction.

To a certain degree, we can compare multimedia authoring to word processing such as the writing of this dissertation. Both activities require the gathering or generation of
source material and the placement of these sources within the presentation environment. A typical word processor gives the user the possibility to change the presentation layout either to be printed or displayed. Usually other properties can be modified by the author like the font, size and color of the text, add lines to form tables or separators, add other graphic icons or images and higher-level structures like chapters and sections. The way the presentation is constructed varies from one word processor to other: a graphical layout engine or a script-based processor. These different approaches are termed paradigms and will be discussed later in this chapter regarding multimedia authoring. In the same way, multimedia authoring tools allow the author to integrate several types of information into a composite presentation. However, unlike word processing, the temporal dimension often dominates the multimedia authoring process. Authoring multimedia is essentially similar to movie making and editing. In this case, the author assembles the individual shots into scenes containing a single coherent thread of the narrative.

In short, a multimedia authoring system is a program that has pre-defined elements for the development of interactive multimedia presentations. The authoring systems vary widely in respect to interfaces, capabilities and learning curve. The methodology by which an authoring system accomplishes its task is termed authoring paradigm. The next sections defines which paradigms can be present and its advantages and disadvantages.

3.2 Multimedia Authoring Paradigms

An author of multimedia has the same goal as a writer of communicating a message to the reader. In order to achieve his goal, the author is required to specify the individual parts of a multimedia document. To ease the task for the author, the specifications should be as transparent as possible to retain the emphasis on the manipulation of the message, rather than on the documents parts. This requires the presentation of the document parts to the author in a way that supports high-level narrative manipulation. The different approaches are termed authoring paradigms. An authoring paradigm presents the author with a particular view of the document.

The majority of multimedia authoring systems can be classified according to a number of different underlying paradigms [48]: structure, timeline, flowchart and script. The paradigms provide different approaches to authoring. More than one paradigm may be present in one system. Each of these paradigms are described next.

38
3.2.1 Structure

Structure-based authoring systems support the explicit representation and manipulation of the structure of a presentation, as shown in Figure 13. With a structure representation, it is possible to group media items assembled in the presentation into sub-presentations which can in turn be grouped. Each of these sub-presentations can be manipulated as one entity. If the timing relations are derived from the structure, alterations to the durations will be propagated through the presentation.

![Structure-based paradigm](image)

Figure 13: Structure-based paradigm

The destinations of choice points, in other words, the points to which the presentation will go when a choice is made by the user, are given in terms of the structure. The structuring may group the media items indirectly if, for example, higher-level concepts are associated with one or more items or groups of items.

3.2.2 Flowchart

A flowchart gives the author a visual representation of the commands describing the presentation. Figure 14 shows an example of a flowchart-driven presentation. Flowchart authoring is similar to procedural programming of the presentation, but with an interface enhanced by icons allowing the visualisation of actions taking place. Presentation narrative can be reflected in the routines and subroutines used. The order of object displaying or removing actions and other events is shown. However, time is not represented explicitly. In spite of being a powerful way to express user interaction, this approach tend to become

39
unwieldy for large presentations.

The destinations of choice points are given in terms of jumping to a new procedure.

![Flowchart paradigm]

**Figure 14: Flowchart paradigm**

### 3.2.3 Timeline

Timelines show the constituent media items placed along a time axis, possibly on different tracks. An example is shown in Figure 15. This kind of representation gives an overview of which items are playing in each time instant during the presentation.

![Timeline paradigm]

**Figure 15: Timeline paradigm**

Timeline-based authoring systems allow the specification of beginning and end times
of each media item in relation to a time axis. Manipulation is possible for individual objects, rather than for groups of objects. If the start time or duration of a media item is changed then this change is independent of any other items placed on the timeline. If some timing constraints are required between these items, the user will have to enforce them manually. The destinations of choice points are given in terms of a new position on the timeline.

3.2.4 Script

A script-based system provides the author with a programming language where positions and timing of individual media items, and other events, can be specified. An example of a script describing a presentation is shown in Figure 16. Authoring the presentation using this paradigm consists basically of programming, which can easily become a difficult task for large presentations. Nevertheless, scripting multimedia presentations is a very flexible and powerful approach that is not restricted by the document model.

The destinations of choice points are given in terms of jumping to a new procedure.

```
set win=main.win
set cursor=wait
clear win
put background "backgnd.pic"
put text "header.txt" at 10,0
put picture "photo.pic" at 20,0
put picture "logo.pic" at 40,10
put text "contents.txt" at 20,10
set cursor=active
```

Figure 16: Script-based paradigm

3.2.5 Comparison

Each paradigm has a particular prominence regarding the creation and visualisation of a multimedia presentation. Although this special characteristic can be seen as an advantage it can also be presented as a disadvantage. The trade-offs between paradigms are: the flexibility of a behaviour that can be specified; easiness of behaviour specification; easiness to view the specified behaviour.

Structure-based systems are good for viewing, editing and navigating through the narrative structure of a presentation, allowing different levels of detail to be shown as
appropriate. Whilst these systems are not particularly suited for time editing, they can be used to edit time by allowing some timing relations to be derived from the structure. The structure itself defines the order of display of the media items. Layout information is specified per event, and an overview of the layout at a particular time is possible only by playing the presentation.

Timeline-based systems have no direct means of editing the narrative structure of the presentation directly, although it can be perceived and navigated as discontinuities of groups of objects along the timeline. The timeline is, however, the best way of showing when objects are displayed on the screen and synchronisation relationships between events. Despite the explicit representation of timing, editing time in a timeline is restricted to individual objects unless some type of structure is present. Layout is specified per object per time unit, so an overview of all objects at a given time is possible.

The flowchart and script-based systems have similar strengths but editing and visualising are usually less intuitive with scripting. Flowchart systems are best suited for rapid prototyping and short-time development projects. In spite of not being mandatory, the structure of narrative can be reflected in the flowchart or the script procedures. In the same way, the navigation through the narrative is possible if the correspondence between procedures and the narrative is maintained. However, timing information can not be shown. Layout information is specified per object, generally as part of the command to display the object. An overview of the layout at a particular time is possible only by playing the presentation. The flexibility of the interaction is high, with flowchart system helping in the specification. Nevertheless, the visualisation of interaction is only possible when playing the presentation.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Flowchart</th>
<th>Timeline</th>
<th>Script</th>
</tr>
</thead>
<tbody>
<tr>
<td>++ ++ ++</td>
<td>+ 0 + +</td>
<td>- - - -</td>
<td>- - - -</td>
</tr>
<tr>
<td>0 + 0 + ++</td>
<td>- - - -</td>
<td>++ + ++ ++</td>
<td>- - ++</td>
</tr>
<tr>
<td>++</td>
<td>- - - -</td>
<td>++</td>
<td>- -</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0 ++</td>
</tr>
</tbody>
</table>

Table 2: Properties of authoring paradigms
In Table 2 a qualitative evaluation of the paradigms is presented. As referred before, each paradigm has its strengths and weaknesses, and may not be suited to the same types of applications. A careful study is therefore necessary to choose the adequate paradigm and, wherever possible, combine them.

3.3 Requirements

The diversity of multimedia approaches analysed in the previous section reflects the large number of requirements that have to be covered by a multimedia authoring system. However, these needs are only partially fulfilled by existing applications. Multimedia authoring requirements can be divided in two distinct categories:

- **Expressive Power** – the ability of the system to cover a broad range of temporal scenarios required by the author;
- **Authoring Capabilities** – for example the level of expertise required for an author to use the system, the straightforward design or abstraction capabilities.

Considering this division, the requirements of a multimedia authoring system will be specified next.

3.3.1 Expressive Power

The expressive power is a criterion hard to measure, since defining an acceptable level of expressive power is strongly dependent on author practice and experience.

We can divide authoring requirements into three sets: (a) those arising from the intrinsic nature of the objects composing multimedia documents, (b) those arising from their composition and finally, and (c) those related to navigation. This analysis is independent from the authoring paradigm provided by the authoring application to build these documents.

(a) Media objects

Authoring environments must provide the author with:

- **A wide variety of basic objects** – objects like text, video, audio, still pictures, virtual animations, programs or applets are examples of what should be supported; these objects differ in nature, as they can be discrete: their content is delivered instantaneously such as text and still picture; or continuous: their content is delivered progressively such as video or audio; a presentation system must allow to
distinguish between mapping (and unmapping) of objects and execution or playtime of continuous objects;

- **The possibility to control the delivery of continuous objects** – ability to express that the content of a continuous object is to be completely delivered even if its end has to be synchronized with other time points;

- **Interactive capabilities** – ability to make any kind of objects achievable (using sensors) during a given period of time with which the user can interact during the visualisation of the presentation;

- **Temporal style definitions** – ability to add motion effects or audio with variation of volume are examples of temporal styles;

- **Support for unpredictable objects** – applets or some videos are examples of such kind of objects, they are either continuous objects or objects transformed into sensors;
  - continuous objects are unpredictable if their duration is not known beforehand or statically, because their effective duration at presentation time can be affected by external factors (resource limitations like machine load for instance).
  - Sensors are by nature always unpredictable since their effective duration is defined by the interaction of the reader during the presentation phase (and may change at every execution).

(b) **Temporal Composition**

As far as expressive power is concerned, temporal composition aims at expressing any arbitrary ordering between temporal intervals corresponding to the different objects.

(c) **Interactions**

Two kinds of interactions can be distinguished:

- User controls such as pause, resume, fast-forward, etc, which provide the user with a way to control the presentation rendering.

- Presentation interactions through sensors.

Presentation interactions can be classified in two classes according to their associated semantics. If the interaction involves all the active objects of the presentation at the visualisation time, it is named a global interaction; if only a subpart of this set is concerned by the interaction, it is a local interaction. Both kinds of interaction appear in a multimedia specification as:
3.3.2 Authoring Capabilities

Regarding its authoring capabilities, we consider the following requirements:

- **Adaptability to computer illiterate people** – how a system can be used efficiently by a large community of authors, in particular those having no particular skills in computer programming;

- **Straightforward design** – the author has a temporal organization of objects in his mind which is mainly expressed in terms of relative temporal placements between objects; an authoring system must allow the user to specify in any order the temporal relations;

- **Indeterministic scenario authoring capabilities** – a scenario can specify multiple presentations of the same document due to the presence of unpredictable objects; the authoring system must help the author to get a global perception of his document, since he can not manually explore all the possible solutions; this can be done by visualization tools and by static checking techniques which can inform the author about some global properties of the scenario;

- **Adaptability to the incremental nature of the editing process** – building an interactive multimedia document is a cyclic specify, test and modify process: one never reaches the right temporal layout at the first stage. Two requirements follow this observation:
  - **Ease of local modifications**: it must be easy for the author to make a local change in the specification, in particular the authoring system must undertake the global consequences of a local change into the document specification: both from the structural and temporal point of view. This feature depends on the used authoring paradigm; and
  - **Fast editing/presentation cycle**: it should be fast to switch between the specification phase of a document and its presentation.

- **Abstraction capabilities** – a multimedia authoring system must help the author to compose large documents by providing the means to abstract and reuse parts of
documents.

- **Multimedia document models** – generic models, such as XML, of textual documents have improved documents manipulation technologies; similarly, the ability to define classes of multimedia presentations eases the author's task and enhances multimedia environments by providing automatic document processing; and

- **Multi-grids reading support** – a multimedia presentation should be understandable by different categories of users; an authoring tool should help the author while designing such kind of documents by allowing the share of common parts, which can be objects as well as temporal scenarios.

### 3.4 Analysis of Available Authoring Tools

Multimedia authoring is not an easy task and has much more to offer than the typical video plus audio content creation. It has a great appeal to the home user as well as to the professional content producer. But the history of multimedia authoring is relatively recent, yet growing at a fast pace, along with the explosion of the Internet. This growth has become even more evident with technologies like Flash. The MPEG-4 standard has promising arguments to enter this market and even take it to a broader range of applications. Still, this technology has not yet convinced all, as the scarce number of commercial MPEG-4 multimedia authoring tools proves it. This section presents an overview of some MPEG-4 authoring tools as well as other tools based in other technologies.

#### 3.4.1 MPEG-4 Authoring Tools

Creating content directly with the scene description language specified in MPEG-4 BIFS, or even in the more friendly XMT textual format, is a very hard and time-consuming task. A tool that allows easy manipulation of objects composing the scene in a graphical environment and preferably in a WYSIWYG fashion is definitely necessary.

Until 2001, the research in tools capable of manipulating MPEG-4 scenes was nearly absent. One exception was an authoring tool for the composition of 3D scenes named MPEG-4 Toolbox [49, 50, 51]. This tool is still the only one that supports more complex 3D nodes, like facial animation. It is based in OpenGL and in the standard's reference tools, also known as IM1. Besides allowing the insertion of basic 3D nodes (box, sphere, cone, cylinder, etc) and of the more complex synthetic head, it is also possible to add simple interactivity to the scenes. More recently, techniques to use moving objects in this authoring tool have been studied [52]. A screenshot of the application is shown in
Figure 17(a).

The other exception in the early stage of MPEG-4 authoring was MPEG-Pro [53, 54]. MPEG-Pro supports 2D authoring of scenes directed by a timeline with the addition of temporal constraints. Given the eminently hierarchical form to organise media objects in MPEG-4, a structure-based view over the scene is also provided. It is based on multiple synchronised views. Figure 17(b) shows the user interface of MPEG-Pro. A template feature is also integrated, helping authors to customise previously created content and use it. A more recent addition to MPEG-Pro provides support for delivery – an interleaving tool [55]. It supports delivery from local system, HTTP streaming and streaming of content over RTP/IP.

(a) MPEG-4 Toolbox  
(b) MPEG-Pro  
(c) 3D authoring tool  
(d) 2D authoring tool

**Figure 17:** MPEG-4 authoring tools

The MPEG-Pro project, being the first MPEG-4 2D content authoring project, became a reference for other projects that followed it. References can be found to two other 2D authoring systems – [56, 57] and [58] (Figure 17(d)). Both use a timeline-based paradigm to edit the scene and fully support the route-based mechanism to add interactivity and
define the behaviour of objects in MPEG-4. Also a structure-based view is provided — presented in a form of tree view. Another project, referenced on [59], focuses on the 3D support but still uses a similar approach to edit the scenes as shown in Figure 17(c). A different approach is presented on [60], providing a simpler interface to the user. In this case the user is oblivious of the MPEG-4 structure and organisation.

Some projects put special emphasis on XMT, the textual language specified in MPEG-4 for scene description. At IBM Research, the Composite Media Group has developed cross-platform MPEG-4 standard compliant technologies [61]. One of its projects is a group of authoring tools that includes command line converters and GUI based authoring tools. The XMTEdit application is one of those tools. It allows the creation of MPEG-4 presentations based on the edition of XMT including also a previewer. Besides XMTEdit, other research projects invested primarily in XMT authoring [62, 63, 64].

From 2002 until now, few commercial solutions have appeared in the market. Envivio offers the 4Mation MPEG-4 Authoring Software [65]. Another commercial 2D authoring environment, named Studio Author, was developed by iVAST. At the time of writing of this text, the lack of information on this product suggests it has been discontinued. It allows a object-oriented composition of media in a 2D environment and uses a timeline-based approach to define the presentation's temporal evolution. On the other hand, Digimax proposes MAXPEG Author [66]. It is a visual authoring environment that supports various 2D/3D nodes, video, audio, animation. This product is based on the project referenced on [59].

3.4.2 Other Authoring Tools

Several commercial authoring tools are available in the market. These use different approaches and have different capabilities. We analyse next three popular multimedia authoring applications that have different authoring paradigms. It is not intended to be an exhaustive study but rather an overview of applications with which potential MPEG-4 authoring tools may compete with.

Director

Director is the most used tool for animation content (Figure 18(a)). It allows the integration of graphics, text, audio and video media items. The timing is defined by the placement of the media items in a timeline, or score, as named in Director. The timeline is divided into discrete time intervals, called frames and whose playing time is determined by the current rate of play, called tempo. The tempo can be changed at any frame. A
media item has an associated position in a frame and it is possible to define a path that the media item will follow across a number of frames. Jumps to other parts of the timeline are achieved through a goto frame command in a scripting language. This proprietary scripting language is called Lingo. Each frame, media item or anchor can have an associated script that will be activated by an event (e.g. click in the media item).

Besides the tracks where media objects are placed, it is also possible to append effects tracks to the presentation. One of these effects tracks is a transition track that allows the specification of transitions. The transition is associated with the first frame where the transition takes place. The transition is characterised by a type (e.g. dissolve, wipe or checkerboard), a duration and a choice of whether the whole display area is affected or only the differences between the frames.

Figure 18: Other multimedia authoring tools
MULTIMEDIA AUTHORING

Flash

Flash was developed by the same software company as Director and many similarities between both are notorious. Therefore, we will only be referring to the most important features that separate both. An overview of Flash technology was also previously presented in Section 2.2.5. Whereas, Director has its main deployment methods related to fixed media (CD/DVD, kiosk), Flash was designed from the ground up as a Web-oriented solution. Other difference is that, Director was primarily bitmap-centric, and Flash, on the other hand, was designed as vector-centric. In fact, vectors are the key to the small file size of most Flash movies. Because of this, Flash became an important element in the Internet environment.

Authorware

Authorware is a professional multimedia authoring tool (Figure 18(b)). Support for interactive applications with hyperlinks, drag-and-drop controls and integrated animation is provided. The process of creation is based on icons representing actions. These are placed in a flowchart defining the behaviour of the whole presentation. Flowcharts can be grouped in subroutines and nested in arbitrary levels. This is often necessary due to the obvious limitation of layout area. The hierarchy of subroutines can be used as an outline, or storyboard working top-down – by stating the sections in the presentation and first filling them in.

It is not possible to get an overview of which media items will be placed on the screen nor when will they be played. On the other hand, interactions are not limited to simple links and can be very complex.
Chapter 4

Design and Specification

After an overview of technologies that enable the representation and description of multimedia content and a discussion on multimedia authoring, we now have the basis for the design and specification of a framework for the manipulation of media objects. Having chosen the MPEG standards as the supporting technologies, the emphasis is on the definition of system requirements and architecture. The model for the data structures (media-data and metadata) is also defined. Finally the interfaces with the user are specified, especially the graphical interfaces due to its importance in a system of this kind.

4.1 System Requirements

Before the implementation of a prototype take place, several requirements, considered the founding basis for the system, were determined and are now characterized.

The first system requirement was that it should be flexible, extensible and built upon open and vendor and platform independent technologies. Open standards should have an important role in the system.

Secondly, it should be based in a distributed architecture, allowing users to access and share resources in a transparent, open and scalable way. Moreover, using a distributed architecture, the system should also allow its deployment in a wide variety of scenarios and environments.

Thirdly, it should be based in relatively small components that form the building blocks of the system and allow the testing and reusing of coherent portions of the system.

Fourthly, the system should support the dynamic multiplexing of media objects, allowing a dynamically variable number of objects, extraction of these objects and mixing of objects from local and remote sources.
The system should allow the use of these media objects to create and edit a multimedia presentation. It should allow the composition in time and in space of objects that can be either text, 2D graphics, still images or video.

It should be possible to change presentation characteristics for individual objects should be possible. This includes means for editing (e.g. cutting and pasting) or manipulating (e.g. translating, rotating, scaling) media objects and the ability to modify properties of these object (e.g. texture).

Seventhly, it should provide the means to associate dynamic and interactive behaviours to media objects.

Eighthly, it should be able to store the content, preserving information useful for re-authoring and re-purposing of contents. At the same time it should also allow to publish content in multiple forms, scaled according to different constraints.

Alongside the stored content information, or media-data, the system should also store additional information, or metadata, that can identify the content. This information can also be used to search and retrieve content as well as create profiled content to the end-user.

Tenthly, a seamless articulation between media-data and metadata should be maintained, always keeping the relationships and navigability between each other.

On the client side, the system should provide the ability to search published presentations using information about the content. It should allow the progressive download of media objects composing a presentation and should provide the means to have user controls (e.g. scan forward, reverse, pause) and to interact with the objects.

Support for a management application should also be present in the system. This application should allow the management of media objects and multimedia presentations (e.g. renaming, deleting, changing description, publishing).

Finally, although security is not a fundamental requirement at this stage, it should be taken into account not to imperil future stages of development. The three pillars of security should be taken into account: confidentiality, integrity and availability.

### 4.2 Use Cases

In the previous section we have defined the system requirements. However, until now we have only mentioned authoring capabilities. In this chapter, we will be considering a more complex system, including authentication and profiling features. Although the developed prototype which will be described described in the next chapter does not implement these
features, an extended specification allows future integration with full-fledged content management and distribution systems. The concluding chapter leaves some more clues on this. With this in mind, we start the definition of typical use cases by identifying the actors that interact with the system.

An actor is an entity external to the system that performs a role when interacting with the system. For a given use case, there is one initiator actor who starts interaction, and possibly several other participating actors. We only consider the initiator actors. At this stage only two entities are identified as having direct interaction with the system – the Network and the User – whose detailed description follows.

The Network actor represents a content delivery infrastructure such as the Internet whether the access is through ADSL, cable, wireless, or satellite.

The User can be defined as an entity that somehow interacts and performs some action within the system and for that purpose must be identified and therefore authorized to do so.

However, the User is an abstraction and needs to be instantiated in some role within the system. Four User instances were identified as having direct interaction with the system: Author, Publisher, Administrator and Client.

The Author creates and manipulates multimedia presentations using media objects previously imported to the system and then stored. The Author can also remove, rename or change other related information about objects and presentations. This is a human user interacting with the system with a GUI.

The Publisher can export previously created content to a multimedia service provider. This export process can be complemented with some profiling, namely associating the content with user profiles or device-specific profiles. In some configurations the Publisher and Author roles overlap. This is a human user, that interacts with the system using a GUI.

The Administrator is responsible for the content management and user management. The content management includes managing stored content and changing its description. Regarding user management, the administrator can create, remove system users or change the user configuration. This is a human user, that interacts with the system using a GUI.

The Client is the consumer of the content produced within the system. Typically this is a human user, using a client application with a content renderer, but there are no restrictions regarding it (e.g. the client can be a robot application collecting content automatically).

From the conceptual point of view, the system can be divided in five subsystems:
- **Authentication Manager** – The tools provided by the authentication subsystem allow identification of all Users that interact with the system. The authentication information can also be used for client profiling.

- **Authoring Tool** – All functionalities regarding creation of multimedia content are grouped in this subsystem. The mapping of this subsystem to an architectural point of view could be a graphical application where the author is able to create, manipulate and compose multimedia presentations using media objects. Browsing of objects and presentations should also be possible.

- **Storage Manager** – The storage and manipulation of content and information in Media Objects and in Multimedia Presentations are the main requirements of this subsystem. Other higher-level requirements include management (e.g. creation, modification and deletion) and transfer.

- **Profile Manager** – The management and verification of profiles are performed within this subsystem. The possible management tasks include creation, deletion and modification of profiles. Profiles allow that different multimedia presentations to be provided according to different profiles. These profiles can be, for example, user or device profiles.
• **Multimedia Service Provider** – The Multimedia Service Provider provides the means for the produced content to be made available as a form of multimedia service to the end-user. All issues concerning the transfer or streaming of content are also dealt within this subsystem.

All diagrams presented in this section use the Unified Modeling Language (UML) [67]. The first, in Figure 19 shows a dependencies diagram of the previously defined actors and subsystems. For clarity purposes not all dependencies are depicted. In the next subsections each of these subsystems is analysed in more detail.

### 4.2.1 Authentication Manager

The *Authentication Manager* manages the access to system resources. All users that somehow interact with the system need to authenticate. After user validation, based on its permissions, they should then have access to the resources asked. As suggested by the definition of actors with different roles, the authentication should follow a Role-Based Access Control (RBAC) model. Users are assigned to roles, permissions are assigned to roles, and users acquire permissions by being members of the roles. Figure 20 shows an UML diagram of the most important use cases that are predicted to happen within this subsystem.

![Authentication Manager Diagram](image)

**Figure 20:** Authentication subsystem use cases diagram

Each use case is briefly described in Table 3.

The first use case should be self-explanatory. As for the second use case, it must
be stressed that each user can change only its credentials. A special case happens with the Administrator which can change any user credentials. When the user credentials are accepted by the system it is established a session which expires after a predefined time. To avoid session expiration, the application that accesses the system on behalf of the user must refresh the session, resending the credentials. During this session, whenever a resource is being accessed by the user application, the Authentication Manager checks the user permission accordingly to the user’s role, accepting or rejecting the access. Figure 21 shows an example of this process.
4.2.2 Storage Manager

The Storage Manager keeps track of all content stored in the system. This content should be stored with additional information about it, which can used to browse and search through the content, be it simple media objects or more complex multimedia presentations.

![Diagram of Media Objects Handling](image)

**Figure 22:** Storage Manager subsystem use cases diagrams

In order to keep the diagrams as simple as possible, we further divide this system functionalities in two packages – Media Object (MO) Handling and Multimedia Presentation (MP) Handling. The possible actions regarding both MOs and MPs are essentially the same. Both use case diagrams are identical and thus only the diagram related to MO Handling is presented – Figure 22.

A brief description of this subsystem use cases is presented in Table 4. Each use case applies

The management of media objects and multimedia presentations include creation, deletion and modification of each element. Both the media-data and the metadata of MOs and MPs can be altered by the administrator and the author. Although we refer the Author as an initiator actor in all use cases of this subsystem, these actions should be intermediated by the authoring tool, as shown in the next subchapter.

57
Table 4: Brief description of the Storage Manager use cases

When browsing through the media objects, a list of objects are shown, as well as information about each one. The list of items may also be filtered by searching parameters defined by the user, either an Administrator or an Author. Whenever possible a preview should also be available (e.g. thumbnail, keyframe, audio segment). The same applies to MPs.

The export and import use cases consist in the transcodification of content from or to the format used within the system, respectively. When importing some content, whether MO or MP, all system-specific information should be created (e.g. system unique identifier, profile-related metadata) and discarded when exporting.

4.2.3 Authoring Tool

The Authoring Tool provides the means to manipulate a multimedia presentation. It is an application with a GUI that allows an author to compose in space and in time media objects, resulting in a coherent multimedia presentation. Figure 23 shows the two generic use cases within this subsystem.

![Authoring Tool Diagram](image-url)

Figure 23: Authoring subsystem use case diagram
In Table 5 the main use cases are described.

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Action</th>
<th>Initiator Actor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compose MP</td>
<td>Compose in space and in time media objects</td>
<td>Author</td>
</tr>
<tr>
<td>Edit MP Dynamics</td>
<td>Change dynamic behaviour of objects (e.g. movement and interaction)</td>
<td>Author</td>
</tr>
</tbody>
</table>

Table 5: Brief description of the Authoring Tool use cases

Although apparently simple, these use cases have complex behaviours that are difficult to specify. The composition of an object may include moving, resizing or rotating, defining its time of entering or exiting of the presentation or even changing innumerable properties such as color, texture and brightness. As for the scene dynamics, the Author can add movement or rotation to objects and attach behaviours that can be activated by the interaction of the final user, e.g. moving or clicking the mouse or pressing a key.

![Sequence Diagram](image)

Figure 24: Typical authoring sequence diagram

To compose the presentation, the author can include media objects stored within the system. For that purpose the application tool, on behalf of the Author, asks the Storage
Manager for a list of objects as shown in Figure 24. This browsing and searching is similar to the one defined previously.

4.2.4 Profile Manager

The Profile Manager allows the system to provide specialised content to end-users. The profiles can be associated to end-users preferences or devices capabilities. Figure 25 shows the identified use cases within this subsystem.

![Profile Manager subsystem use case diagram](image)

Figure 25: Profile Manager subsystem use case diagram

Table 6 shows a brief description of the use cases depicted in the previous figure.

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Action</th>
<th>Initiator Actor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manage Profiles</td>
<td>Add or Remove profile; change characteristics of a profile</td>
<td>Administrator or Publisher</td>
</tr>
<tr>
<td>Associate Profile to Service</td>
<td>Associate a defined profile to a service provided by the system</td>
<td>Publisher</td>
</tr>
</tbody>
</table>

Table 6: Brief description of the Profile Manager use cases

The Manage Profiles use case can be divided in three alternative tasks. The Administrator or the Publisher can either create a new profile, remove a profile or change the profile characteristics. When a service is created, the Publisher can also associate it to an existing profile, although it is not mandatory.

4.2.5 Multimedia Service Provider

The Multimedia Service Provider provides the means for the content created and stored within the system to be published. The publishing process consists in providing a form of access to the content and optionally in associating the service to profiles. This subsystem
also conceals the transport and delivery mechanisms to the end-user. Figure 26 depicts all the expected use cases.

![Multimedia Service Provider use case diagram](image)

**Figure 26: Multimedia Service Provider use case diagram**

As previously done for other subsystems, we present a description of each use case in Table 7.

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Action</th>
<th>Initiator Actor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manage Multimedia Service</td>
<td>Add or Remove multimedia service</td>
<td>Administrator or Publisher</td>
</tr>
<tr>
<td>Browse Multimedia Services</td>
<td>Browse through available multimedia services</td>
<td>Administrator, Publisher or Client</td>
</tr>
<tr>
<td>Transfer Content</td>
<td>Transfer multimedia content from the service provider to the end-user terminal</td>
<td>Client</td>
</tr>
</tbody>
</table>

**Table 7: Brief description of the Multimedia Service Provider use cases**

Similarly to other management use cases, the Manage Multimedia Service comprises the creation and deletion of multimedia services. The browsing of multimedia services returns a list of available services. When a Client browses the service provider, its characteristics are compared to the profile associated with the services and a customised list is returned. This crossing of information results in an adaptive multimedia content. Finally, when a service is selected, the Client requests the transfer of content. Figure 27 depicts these steps in a typical access to the Multimedia Service Provider.
4.3 A Distributed Architecture

Having defined the generic system requirements and having specified the most relevant use cases, we can now turn to the system architecture.

The proposed architecture consists of a distributed system [68], allowing the development of a scalable multimedia authoring system. Several authors can access to media objects or multimedia presentations remotely stored. The storage can be assured also by a group of servers which are transparently accessed by the author application. The location of the servers is also transparent. A similar scenario is also possible regarding the client application.

Another important objective of using a distributed systems is openness. An open distributed system offers services according to the specified syntax and semantics of these services. The specification is done through the definition of interfaces. If properly specified, interoperability of different implementations is achieved.

The system has six different modules, as shown in Figure 28: Authoring Tool, Publishing Tool, Administrator Tool, Repository, Multimedia Server and Client Application.

The Repository module stores media objects that can be used in the system, as well as multimedia presentations. Each object and presentation has a unique identifier within the system allowing an univocal identification. The storage of both in the Repository follows
the data model specified in the next subsection. This module maps all use cases related to Storage Manager.

The Authoring Tool is the module responsible for the creation and modification of multimedia presentations. The manipulation of multimedia applications consists in the composition of existing media objects stored in one of the repositories. The produced presentation is then stored in one of the Repositories in a format that describes the presentation and its composing scenes. As expected, this module maps all use cases related to the Authoring Tool.

The Publishing Tool exports the presentations to a Multimedia Server. This export process uses the description of a composed multimedia presentation stored by the Composer in a Repository. If external references are used, the media objects used in the presentation, also stored in some Repository and identified by the unique identifier, will also be exported. The presentation data is then made available as a multimedia service in one Multimedia Server, which in turn can be accessed by a Client Application. This module maps all use cases related to Profile Manager and the ones initiated by the Publisher actor.

The Administration Tool is the module used primarily by an administrator, which can manage other users, media objects, multimedia presentations and multimedia services. Authentication is also managed within this module. This module maps all use cases related to Authentication and those initiated by the Administrator actor.

The Multimedia Server is the only boundary to other networks. If the Multimedia
DESIGN AND SPECIFICATION

Server is a file-based Server (HTTP, FTP), a file, including all information of the multimedia presentation and the composing objects, is created and then stored in the server. This file can then be downloaded by the Client Application, using an URI to locate it [69]. On the other hand, if the server is stream-based one possible option is to create a hinted file, adding additional information that aids the server in the streaming process. One other possibility for the Multimedia Server is one that is content-aware, that is, a server that, using the description provided by the system, can stream each object needed at some point in time using, for example, RTSP . This allows storing just the description of the multimedia application while the media objects remain only in the repositories and are fetched as needed. Another feature that is only possible with a content-aware Multimedia Server is remote interactivity. When remote interactivity is possible, the Client Application may send some commands to the server that may change some characteristic of the application being sent. This module maps all use cases related to Multimedia Service Provider.

The Client Application can access, through any network, one of the Multimedia Servers and start to download or ask to start the streaming of a multimedia service. The user can then view and interact with the application. This module maps all use cases initiated by the Client actor.

Although the authoring, publishing and administration tools modules are shown as single elements in the system architecture, these should in fact have a many-to-many relation with other modules.

4.4 Data Model

After the specification of a modular architecture, a structure for the data stored in the system is still missing. With the specification of a data model we intend to establish a set of rules that defines a particular way of organising a collection of digital items. To be useful, a digital item should be arranged according to the rules of an appropriate data model and stored in a file format that can represent this data model. Moreover, it should be accessed with software that understands the file format and the data model, and can present the data in an appropriate way. This section has the objective of making possible the previous assertions.

As already mentioned, the system has two types of data representation: media objects and multimedia presentations.

A media object is considered to be any kind of media information that has, by itself,
some meaningful representation. Examples of media objects that can be imported to the system are visual objects (natural or synthetic), audio objects (also natural or synthetic), still image objects (photographic or vector-defined), or textual objects. These objects can also be timed, i.e., modifications to the object are made with time. If present, this timing information is self-contained. The media objects are used to compose a multimedia presentation using the Composer. As already stated, alongside the media-data each object has also some information about it, metadata. The data model used for both parts is defined next.

Multimedia presentations are a structured representation of media objects in space and in time. They also include additional information that add dynamic and interactive behaviour to objects.

4.4.1 Media-Data

Multimedia presentations are stored in MPEG-4. Both binary and textual formats are supported in storage, but only the binary format is used for multimedia delivery to the end-user terminal.

The media-data of media objects is stored in its raw format. If, for example, the object is an image it can be stored in any of the standards for image compression and representation – JPEG, PNG, etc. As for audiovisual content, supported formats should include popular formats like MPEG-1, MPEG-2, MP3, AAC, MPEG-4 Video (e.g. XviD), etc. Each of these formats is directly supported by MPEG-4 Systems. The system also supports media-data for synthetic content that can be transcoded to the MPEG-4 scene description language. The formats reviewed on Chapter 2, like “swiff” (.swf) or VRML (.wrl), are some of the formats expected to be supported. Content can then be created with specific tools in these formats and imported to the system. When the author uses these media objects, the system translates the description from the original format to MPEG-4 description format – BIFS.

4.4.2 Metadata

The content description, or metadata, conforms to the MPEG-7 standard, described in Chapter 2. More specifically, the description of objects and presentations stored in the system uses the tools defined by the Multimedia Description Scheme (MDS) part of MPEG-7 [41]. The standard covers a wide range of applications and, as such, defines many descriptors that are not necessary in a system of this kind. The set of descriptors used are
specified in this subsection. However, it is not intended to be an exhaustive specification of descriptors and should always be referenced to the MPEG-7 MDS standard.

Root Element

The Mpeg7 element is the root element and provides metadata about the description as well as information that is common to the description, such as the language of the text and the convention for specifying time. The standard provides a choice of elements for creating either a complete description or a description unit. A Complete Description describes multimedia content composed by the top-level types defined later. A Description Unit can be applied in the representation of partial information from a complete description. Only the Complete Description is used. Table 8 summarizes the usage of descriptors.

<table>
<thead>
<tr>
<th>Element</th>
<th>Base Type</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mpeg7</td>
<td>Mpeg7Type</td>
<td></td>
</tr>
<tr>
<td>DescriptionMetadata</td>
<td>DescriptionMetadataType</td>
<td>Fully used</td>
</tr>
<tr>
<td>DescriptionUnit</td>
<td>Mpeg7BaseType</td>
<td>Not used</td>
</tr>
<tr>
<td>Description</td>
<td>CompleteDescriptionType</td>
<td>See Table 9</td>
</tr>
</tbody>
</table>

Table 8: MPEG-7 root elements used

Complete Description

The top-level types are used in complete descriptions to describe multimedia content and metadata related to content management. The following top-level types are defined:

- CompleteDescriptionType – Complete descriptions
  - ContentDescriptionType – Complete descriptions of multimedia content.
    * ContentEntity Type – Describes multimedia content entities such as images, videos, audio, collections, etc.
    * ContentAbstractionType – Describes the abstractions of multimedia content:
      - SemanticDescriptionType – Describes the semantics of multimedia content.
      - ModelDescriptionType – Describes the models of multimedia content.
      - SummaryDescriptionType – Describes the summaries of multimedia content.
· **ViewDescriptionType** – Describes views and view decompositions of audio-visual signals.
· **VariationDescriptionType** – Describes variations of multimedia content.
· **ContentManagementType** – Describes metadata related to content management:
  * **UserDescriptionType** – Describes a user of a multimedia system.
  * **MediaDescriptionType** – Describes the media information of multimedia content.
  * **CreationDescriptionType** – Describes the process of creating multimedia content.
  * **UsageDescriptionType** – Describes the usage of multimedia content.
  * **ClassificationSchemeDescriptionType** – Describes a classification scheme for multimedia content.

Figure 29 depicts the hierarchy of the types just described.

- **Figure 29: Hierarchy of top-level types**

Although some sub-types of **ContentAbstractionType** could provide valuable information when describing both media objects and multimedia presentations, the system should only support the **ContentEntityType**. The main reason for this is the simplification of description and search processes. Using all content description types implies,
DESIGN AND SPECIFICATION

for example, complex user interfaces for data insertion without getting greater benefits from it, at least at this stage. The system should nevertheless take into consideration the eventual addition of more complex description sets in future stages of development. For example, the VariationDescriptionType could be used to define sets of media objects derived from a source media object but with different resolutions or color spaces. On the other hand, ViewDescriptionType could be used to define a media object which is a view or part of another media object. Both would enhance the search and reuse of media objects.

The multimedia types the system used for content description are ImageType, VideoType, AudioType and AudioVisualType for media objects and MultimediaType for multimedia presentations. Refer to section 4.4.5 of the ISO/IEC 15938-5 standard for syntax and semantic details on each multimedia type.

<table>
<thead>
<tr>
<th>Type</th>
<th>Base Type</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompleteDescriptionType</td>
<td>(abs)</td>
<td></td>
</tr>
<tr>
<td>ContentDescriptionType</td>
<td>(abs) CompleteDescriptionType</td>
<td></td>
</tr>
<tr>
<td>ContentEntityType</td>
<td>ContentDescriptionType</td>
<td>Used</td>
</tr>
<tr>
<td>ContentAbstractionType</td>
<td>(abs) ContentDescriptionType</td>
<td></td>
</tr>
<tr>
<td>SemanticDescriptionType</td>
<td>ContentAbstractionType</td>
<td>Not used</td>
</tr>
<tr>
<td>ModelDescriptionType</td>
<td>ContentAbstractionType</td>
<td>Not used</td>
</tr>
<tr>
<td>SummaryDescriptionType</td>
<td>ContentAbstractionType</td>
<td>Not used</td>
</tr>
<tr>
<td>ViewDescriptionType</td>
<td>ContentAbstractionType</td>
<td>Not used</td>
</tr>
<tr>
<td>VariationDescriptionType</td>
<td>ContentAbstractionType</td>
<td>Not used</td>
</tr>
<tr>
<td>ContentManagementType</td>
<td>(abs) ContentAbstractionType</td>
<td></td>
</tr>
<tr>
<td>UserDescriptionType</td>
<td>ContentManagementType</td>
<td>Used for profiling</td>
</tr>
<tr>
<td>MediaDescriptionType</td>
<td>ContentManagementType</td>
<td>Not used</td>
</tr>
<tr>
<td>CreationDescriptionType</td>
<td>ContentManagementType</td>
<td>Not Used</td>
</tr>
<tr>
<td>UsageDescriptionType</td>
<td>ContentManagementType</td>
<td>Used for profiling</td>
</tr>
<tr>
<td>ClassificationDescriptionType</td>
<td>ContentManagementType</td>
<td>Not used</td>
</tr>
</tbody>
</table>

Table 9: MPEG-7 top-level types used

Although creation information (using CreationInformation DS) and media information (using MediaInformation DS) are supported by the system, it however does not support the top-level types MediaDescriptionType and CreationDescriptionType. The former describes the storage format and coding, while the later describes whom, when, and where, etc, the multimedia content was created. Both information types are therefore used as children of the multimedia content types (Image, Video, Audio, etc). On
the other hand, the UserDescription and UsageDescription are supported and the
respective information types are only used at the top-level. The description types are used
for profiling information management. For each profile, the user description tools should
be used, allowing the filtering of content. Finally, ClassificationSchemeDescriptionType
should not be necessary as the classification schemes defined at Annex B of ISO/IEC
15938-5 should suffice.

Table 9 summarises the use of descriptor types in the system. Abstract types are
identified by (abs).

Description Example

We now present a description example that should help clarify how a media object and
multimedia presentation are described in the system. This example describes a Media
Object that contains audiovisual content.

```xml
< MPEG7>
  <DescriptionMetadata>
    <Confidence>1.0</Confidence>
    <Version>1.1</Version>
    <LastUpdate>2004-07-09T03:20:25+09:00</LastUpdate>
    <PublicIdentifier type="UUID">
      09f2470-ba0-11cd-b579-08002b30bfeb
    </PublicIdentifier>
    <PrivateIdentifier>completeDescriptionExample</PrivateIdentifier>
  </DescriptionMetadata>
  <Creator>
    <Role href="creatorCS">Creator</Role>
    <Agent xsi:type="PersonType">
      <Name>
        <GivenName>Luis</GivenName>
        <FamilyName>Taixeira</FamilyName>
      </Name>
    </Agent>
  </Creator>
  <CreationLocation>
    <Region>pt</Region>
    <AdministrativeUnit>Porto</AdministrativeUnit>
  </CreationLocation>
  <CreationTime>2004-10-10T19:45:00+09:00</CreationTime>
</DescriptionMetadata>
<Description xsi:type="ContentEntityType">
  <MultimediaContent xsi:type="VideoType">
    <Video>
      <MediaLocator>
        <MediaUri>image.mpg</MediaUri>
      </MediaLocator>
      <CreationInformation>
        <Creation>
          <Title>Euro 2004</Title>
        </Creation>
      </CreationInformation>
      <TextAnnotation>
        <FreeTextAnnotation>Portuguese Fan</FreeTextAnnotation>
        <TextAnnotation>
          <MediaInformation>
            <MediaIdentification>
```
4.5 User Interfaces

In this section we define the expected requirements for the user interfaces. Within the system the user interfaces should be predominantly graphical. The user interfaces should be designed to match the skills, experience and expectations of its anticipated users. However, this kind of knowledge would have required a long iteration process with the final system users [70] and such study falls off the scope of this dissertation. We opted not to specify some graphical characteristics like the controls' layout and focus solely on feature enumeration.

The more complex user interface in the system is the Authoring Tool GUI. But besides this interface, four more user interfaces are expected – Media Object Management, Multimedia Presentation Management, Publishing Management and Administration Management. Although they are defined as separate interfaces, some could be merged in a single application.
The Authoring Tool GUI is expected to be based on two interaction styles: direct manipulation and menu selection. The former is only suitable where there is a visual metaphor for tasks and objects which is clearly this case. Although hard to implement, it is fast and intuitive. The latter suits general purpose systems, avoids user error but can become slow for experienced users and complex with many menu options. The combination of both should withstand the individual disadvantages. The main features expected from the Authoring Tool GUI are:

- A representation of the presentation being edited in a tree-like structure, closely related to the MPEG-4 scene hierarchy.
- A WYSIWYG control to edit the presentation.
- A Media Objects searchable browser used to select an object to be placed in the presentation being edited. Figure 30 shows the hierarchy diagram for the media object searching task.
- A Multimedia Presentations searchable browser. The search process is similar to the one used for Media Objects.
- Shortcut controls to add objects to the presentation.
- A timeline for time editing of objects.
- An object property browser and editor.
- An event browser.
- A route browser and editor.
- Property editor of the multimedia presentation.
- Editor of Multimedia Presentation metadata.

The other user interfaces can be implemented using either the menu selection or the form fill-in interaction styles. However, these interfaces, especially the MO and MP management interfaces, should be preferentially web-based using web forms. The main reason for this choice is the large set of dynamic and heterogeneous data being displayed. The interface can be built dynamically by the application using HTML without major headaches with information layout. The features for the remainder interfaces are described next.

- The Media Object Management interface should:
  - List all media objects in repository.
  - Allow the insertion of keywords for result filtering.
  - Show a preview of the media object (thumbnail, keyframe or audio segment).
  - Show relevant information of each media object (size, format, etc).
  - Allow the deletion of Media Objects.
  - Allow importing and exporting of Media Objects.
Figure 30: Hierarchy task analysis of Media Object search

- Allow the modification of Media Objects’ metadata.
- The Multimedia Presentation Management interface should:
  - Have equivalent features as the ones defined for media objects.
  - Show MPEG-4 specific information.
  - Provide a MPEG-4 track browser for files using the binary format.
- The Publishing Management interface should:
  - Lists all available multimedia presentations in the system.
  - Lists all available multimedia services publically available.
  - Provide a track browser.
- The Administration Management interface should:
  - Have the same features as the other management interfaces.
  - Lists all users.
  - Allow the addition of a new user.
  - Allow the deletion of an existing user.
  - Allow the modification of an user configuration (e.g. attach or detach a role).

The requirements just defined are high-level requirements, which can be further refined through the operation of a prototype, described in the next chapter, by system users.
A Multimedia Authoring Framework: edVO

Having specified the system functionalities of a multimedia authoring framework, the task of implementing a prototype that could fulfill the requirements was undertaken thereafter. The implementation project was code-named edVO. It is an ongoing project but with important functionalities already fully developed. This chapter gives an overview of the strategy followed during development and some implementation details on each module.

5.1 Overview

To assert the system requirements defined in Chapter 4 and to serve as proof of concept, an experimental prototype was developed. The prototype, or edVO (acronym for the dissertation’s title) as it became to be known in the project, deployed an initial version of the system. During the design process parts of the prototype were partly implemented in order to help explore options and develop a UI design. In fact, a prototype can be used to help with requirements elicitation and validation.

The development was performed in small releases. The minimal useful set of functionalities that provides demonstrability was developed first. The following releases of the system incrementally add functionalities to the first release. This allowed continuous validation and testing of the model.

Another important process, followed during the prototype implementation, was the reutilization of code. In modern software engineering, code reuse can constitute an important decrease of development time and cost. To allow the reuse of parts of the prototype, its implementation was based in components. Besides the benefits of code reuse, component-based systems have other characteristics:
Components are independent and do not interfere with each other. Component implementations are hidden. Communication is through well-defined interfaces.

A component is an independent, executable entity that provides a service without regard to where the component is executing or its programming language. Component implementations are opaque and can be made up of one or more executable objects. The services offered by a component are made available through an interface and all component interactions take place through that interface. Writing a wrapper component that provides functionalities of existing legacy systems through public interfaces is also possible using components. Similarly, if an improved implementation of certain components, or parts of a component, need to be integrated in the system, it can be done without affecting other components. Usually, only some glue code should be required.

Component testing, which consists on individual components testing, was ensued during all system development. On the other hand, system testing, consisting on the test of components’ integration and of functionalities of the system as a whole, did not have the same regularity. Although some system testing was done, this type of tests should be performed by an independent testing team when further development is completed.

Figure 31: edVO prototype architecture
Considering all the above premises, the requirements and use cases defined in the previous chapter, and the conceptual system architecture shown in Figure 28, the proof of concept can be attained with the prototype schemed in Figure 31. The prototype consists of four different modules: edvo.composer, edvo.admin, edvo.server and edvo.client.

The edvo.composer is the functional equivalent of the Authoring Tool conceptual module. It communicates solely with the edvo.server using two types of protocols: XML-RPC and HTML over HTTP. The former is used for control commands and the latter for structured transaction of user interfaces and commands. Only in the first case, well-known interfaces need to be specified. The prototype does not limit the number of edvo.composers.

The edvo.server is the steering module of the system and provides functionalities that directly map with the Repository and Multimedia Server conceptual modules. It also implements some core functionalities attributed to the Administration Tool and Publishing Tool. The edvo.admin acts as the front-end to these functionalities. It communicates with all the other modules using several protocols. Whereas for user interfacing and control XML-RPC and HTML over HTTP are used, for content transfer the server supports FTP, HTTP and RTP/RTSP. Currently, only one edvo.server is available in the prototype. Support for more than one server would require transparent localization of the servers for the other modules and eventually synchronisation between different servers.

As we have said, edvo.admin is a front-end to the administration and publishing functionalities. It communicates only with the edvo.server, using HTML. It basically consists of a web browser that accesses to the public HTML interfaces provided by the server. There is no functional limitation to the number of edvo.admin modules in the system.

Finally, the edvo.client acts as the functional peer of the Client Application conceptual module. It communicates with the edvo.server using HTML to browse and request a service. When content is requested by the player core, the server may transport it using FTP, HTTP or RTP/RTSP. It is a stand-alone application but can also be a plugin of a web browser.

Each of the above functional modules of the prototype are described next, where implementation details are also provided. Before that, an overview of the external tools used within the prototype is presented.
5.2 Integration of External Tools

In a project of this kind the use of external tools and software packages reveals itself as a valuable productivity enhancement. Faster prototype developments is achieved, and if care is taken regarding the separation in components, some of these tools can easily be swapped by one built from scratch or other external tool. To accomplish that, wherever possible, abstract or well-defined interfaces should be defined.

In this section an overview of the external tools used in the prototype is presented. It is also described the integration process within the system. In some cases, where the integration demanded the development of wrappers, some implementation details are also adduced.

5.2.1 GPAC

GPAC Project on Advanced Content (GPAC) [71] is an implementation of the MPEG-4 Systems standard. It is developed in ANSI C to achieve portability and to keep the memory footprint as low as possible. Currently, it runs under Windows, WindowsCE/PocketPC and Linux platforms. It is licensed under the GNU General Public License.

GPAC provides a small and flexible alternative to the MPEG-4 Systems reference software, known as IM1 and distributed in ISO/IEC 14496-5. The MPEG-4 Reference software is a large and not always consistent implementation of the standard. IM1 was primarily designed to verify the standard rather than provide a small, stable software.

The GPAC library can be divided in several modules, providing different functionalities at different levels. Figure 32 shows a possible layered representation of these modules. The main features provided by the modules depicted are summarized next.

- ISO Media File Format
  - Supports reading, writing/capturing, edition;
  - Highly flexible interleaving of media data;
  - Support for RTP hint tracks; and
  - Support for Movie Fragments – enables writing the file as a set of small meta and media data chunks.

- Object Descriptor (OD) Framework
  - Includes OD codec; and
  - Includes OCI codec.

- Binary Format for Scenes (BIFS) and Scene Graph
  - Support for MPEG-4 scene graph with configurable node set;
INTEGRATION OF EXTERNAL TOOLS

Figure 32: GPAC modules

- Support for interactivity types (events and routes, interpolators, conditionals and valuators);
- Includes scene manager for authoring, textual dumping (BT/XMT) and import (BT/XMT); and
- Includes BIFS coded.

- Elementary Stream (ES) Management
  - Supports scalable codecs (audio, video and systems);
  - Handles synchronization, media management and load scheduling; and
  - Handles time control and segment descriptors.

- Rendering
  - Supports direct and indirect rendering;
  - Implements most of 2D nodes defined in BIFS and most of the new amendments to the standard (Advanced Text and Graphics);
  - Supports experimental rendering of some 3D nodes; and

77
-- Audio rendering and software mixing (mono/stereo only).

**pygpac - Python bindings for GPAC**

When the implementation started it was clear that support for scripting within the system would be very important [72]. Using C/C++ for development would run faster than applications written in a scripting language. In fact we have opted to use C++ to develop the composer that will be described later. But developing in C/C++ is much more time-consuming than developing in scripting languages and, depending on the application, the speed penalty of using a script language is not very significant [73]. This is especially relevant when developing a prototype where application speed is often less important than development speed. Because of that, scripting capabilities were added during this project to GPAC. The relevance of using a scripting language hopefully will become even more evident ahead in this chapter.

Choosing between all available scripting languages like Perl, TCL, Ruby or Python among others, is often a matter of taste. In this case, the chosen language for scripting support was the Python programming language [74]. Python is an interpreted, interactive, object-oriented programming language that has a increasingly large supporting community. Its clean syntax, small number of powerful high-level data types, easy integration with C/C++, large set of supporting tools proved to be decisive [75].

Using the Python C API it is possible to create extensions to Python. The extensions are implemented in dynamic-load libraries whose functionalities are then accessed by Python [76]. Usually some repetitive and error-prone coding is necessary to implement the bindings. Several tools to automate the creation of bindings are available, and one of those tools were used to wrap GPAC: the Simplified Wrapper and Interface Generator (SWIG) [77]. SWIG generates the C or C++ source code for the extension based on the header files, generally with some help using additional annotation in an interface description file [78].

In spite of the C implementation of the GPAC library, special care has been taken to provide a more object-oriented scripting interface. Functions related to a specific C structure have been grouped as methods of a Python class. Considerable effort have been delivered to provide the first release where almost all library public functions and structures have been implemented as Python bindings.

To show how the bindings can be used, some small scripts are now presented. The script bellow opens an existing MPEG-4 binary file, prints its duration and, for each track composing the file, the respective media type is also printed.
import pygpac
file = pygpac.M4File('flowers.mp4', pygpac.M4_OPEN_READ)
print 'Total Duration:', file.GetDuration()
for idx in xrange(1, file.GetTrackCount() + 1):
    print 'Track %d: %s', (idx,
        pygpac.MP4TypeToString(file.GetMediaType(idx)))

The next example manipulates a scene graph adding as root node an
OrderedGroup and other child nodes. A route is also created between a field of a
PositionInterpolator2D and a Transform2D node.

import pygpac
sg = pygpac.SceneGraph()
node = sg.NewNode(pygpac.TAG_OrderedGroup)
sg.SetRootNode(node)
t2d = sg.NewNode(pygpac.TAG_Transform2D)
node.InsertChild(t2d, -1)
shape = sg.NewNode(pygpac.TAG_Shape)
t2d.InsertChild(shape, -1)
interp = sg.NewNode(pygpac.TAG_PositionInterpolator2D)
node.InsertChild(interp, -1, pygpac.M4OK
route = sg.NewRoute(interp, 2, t2d, 7)

The last example opens an MP4 file and, for each track, dumps its raw content. For
example, if a track contains the media-data of a visual object consisting of a still image,
the originally encoded image is stored in a file (e.g. JPEG, PNG files).

import pygpac
file = pygpac.M4File('flowers.mp4', pygpac.M4_OPEN_READ)
for iTrack in xrange(1, file.GetTrackCount() + 1):
    dump = open('track%d.bmp' % iTrack, 'wb')
    for iSample in range(1, file.GetSampleCount(iTrack) + 1):
        sample, sdindex = file.GetSample(iTrack, iSample)
        dump.write(sample.getData())

Unit testing was performed during the development of this Python extension to trap
possible bugs at early stages of development.

5.2.2 FOX Toolkit

FOX is a C++ based Toolkit for developing GUIs [79]. It offers a wide, and growing,
collection of controls' facilities such as drag and drop and selection. The toolkit author
has put effort on making FOX "one of the fastest toolkits around", and on memory usage
minimization. FOX uses techniques to speed up drawing and spatial layout of the GUI.
Memory is conserved by allowing programmers to create and destroy GUI elements on the fly.

FOX's license is essentially based on the GNU General Public License. Currently, FOX runs on a large number of operating systems, ranging from Linux, FreeBSD, SGI IRIX, HP-UX, IBM AIX, SUN Solaris, DEC/Compaq Tru64 UNIX, to all Windows versions. On Windows, the Toolkit does not use the native controls. Instead, it uses the WIN32 API drawing functions to render the controls while on all Unix flavors the equivalent X-Server drawing functions are used. This means that an application will look and work almost identically in all platforms. Moreover, most of the FOX implementation is completely oblivious to the underlying platform. However, not using native controls, especially in Windows, can be seen as a drawback and, if special care is not taken, the application can have an alien look.

After an appraisal of the pros and cons of FOX Toolkit and other equivalent toolkits (namely wxWidgets, QT, GTK), the choice came down to FOX. This choice is due mainly to its stable and lightweight implementation. The Toolkit concentrates on the GUI and leaves to the programmer the freedom to use other libraries for non-GUI functionalities.

FOX allows the creation of additional controls and GUI elements, simply by taking existing controls, and creating a derived class that adds or redefines a specific behavior. During the development of this project, several custom controls were implemented to fulfill special requirements of the Composer.

### 5.2.3 Mozilla

Mozilla is a project that continued Netscape Communicator as an open project [80]. It is a cross-platform, integrated framework of Internet applications, including web browser, e-mail client, address book, web page composer, chat software and calendar application. It is released under the Netscape/Mozilla Public Licenses (NPL/MPL). Mozilla is currently targeted for Win32, MacOS, and Unix/X systems but also runs on MacOS X, OS/2, and BeOS, and ports exist for some other platforms.

Gecko is the Mozilla's browser engine. It also has modules for networking, parsing, content models, among others. In other words, Gecko includes all functionalities that are not application-specific. Gecko allows developers to use the Mozilla technology in other applications, embedding a web browser in the application and opening channels and streams through the network back-end.
EMozilla

EMozilla is a generic Mozilla embedding library developed for the purpose of this project. As shown in the Figure 31, some prototype's modules use web capabilities to access the server. This is achieved by the EMozilla library that allows any application to use Mozilla's functionalities as a web browser. A specific control to be used in FOX called **fxMozilla** was also developed and used in the composer and client modules. Some implementation details are provided next.

![Diagram of EMozilla](image)

**Figure 33: Embedding Mozilla**
The Gecko Embedding API provides a set of interfaces to access services and components and to control the embedded application, and another set of interfaces that the containing application must implement in order to receive asynchronous notifications from the embedded browser. The public interfaces are accessed through Cross Platform Component Object Module (XPCOM) interfaces. XPCOM is a framework which allows developers to develop software projects in small modularized pieces, largely inspired on Microsoft’s COM technology.

The embedding layer consists of several components built on top of XPCOM and its services. Much of the Gecko functionality is exposed through a component called the nsWebBrowser. Embedding applications can use this component to easily access many of Gecko’s features. Each WebBrowser instance represents the client-area of a typical browser window. The WebBrowser exposes a set of interfaces which allow the embedding application to control activity and respond to changes within this client area. Figure 33 depicts the process of embedding Mozilla’s browser functionalities.

5.2.4 Zope

Zope is an open source web application server [81] whose core is implemented in the Python programming language [74]. It is based in a transactional object database which can store, besides content and custom data, dynamic HTML templates, scripts, a search engine, and relational database (RDBMS) connections and code. Zope has a tightly integrated security model. The transactional model applies not only to Zope’s object database, but also to many relational database connectors, allowing strong data integrity. This transaction model happens automatically, ensuring that all data is successfully stored.

Zope includes its own HTTP, FTP, WebDAV, and XML-RPC serving capabilities, but can also be used with the Apache or other web servers. There are several plug-ins available that extend the basic set of tools: new content objects; relational database and other external data source connectors; advanced content management tools; and full applications for e-commerce, content and document management, or bug and issue tracking.

Having chosen Python as the supported scripting language, the choice of using Zope as the server’s foundation was firstly due to the possibility of using Python scripts to generate dynamic HTML. Moreover, Zope provides a FTP server and XML-RPC capabilities. Zope views a web application in terms of objects. Objects are bundles of content and behavior that are referred by an URL. Every Zope object can respond to HTTP requests as well as XML-RPC commands. All Zope objects are publishable and thus all are XML-RPC aware. In fact, Zope will encode a response, so it is sufficient to return standard Python
objects and Zope will marshal them into XML-RPC format. This is an important feature as it allows an application to send control commands to the server.

Instead of Zope, another web server could be used, like the Apache Web Server. The access to the repository could be performed by CGIs programmed in C or Perl. The use of Perl required also the creation of Perl bindings to the GPAC library. After careful consideration Zope proved to be the best choice.

5.2.5 Darwin Streaming Server

Darwin Streaming Server [82] is the open source version of the QuickTime Streaming Server, allowing streaming of hinted QuickTime, MPEG-4, and 3GPP files over the Internet via the industry standard RTP and RTSP protocols. The streaming server runs on Windows NT and Windows 2000 and several UNIX implementations, including Mac OS X, Linux, FreeBSD, and the Solaris operating system.

The purpose of the Darwin Streaming Server in the system is solely to provide streaming capabilities to content delivery from the server to the client application with full support for MPEG-4 hinted binary files and 3GPP files. Other open-source alternatives were considered, namely MPEG4IP [83] and VideoLAN [84] but, in the end, Darwin Streaming Server proved to be the easiest streaming tool to integrate in the prototype.

5.3 Design and Implementation Analysis

In this section we analyse some design and implementation issues of the edvo prototype. As the focus of this project is on multimedia authoring, we concentrate on edvoComposer and edvoServer. The other modules are complementary from the authoring point of view and hence the details are not deeply considered. Furthermore, the edvoAdmin and edvoClient prototypes are fairly simple, and were mainly used, to test the main modules.

5.3.1 Composer

As previously stated, the edvoComposer is the module responsible for the creation of multimedia presentations using the MPEG-4 description framework to compose objects in time and in space. This module implements the requirements defined in the last chapter and the UI specification in Section 4.5. It is implemented in C++ [85], using the previously described FOX Toolkit for GUI rendering and control, and EMozilla to HTML access to
the server. GPAC library is also used within the composer to handle MPEG-4 import and export functions, as well as scene graph manipulation and rendering.

**Encapsulation**

Taking into account the first requirement defined in Section 4.1 – flexibility and extensibility – edvo::Composer applies some design patterns [86] to overcome problems concerning these requirements. One important issue is the *encapsulation*, of classes and components. By definition, a contained implementation (type, data or function) that is not accessible or detectable programmatically through the logical interface of a component is said to be *encapsulated* by that component [87]. In other words, any change to the implementation, guaranteeing that the logical interface remains the same, will have minor, if no impact on other components. In edvo::Composer every object related to scene manipulation follows this rule. If, for some reason, a particular functionality provided by GPAC is to be developed from scratch or replaced by other library, only the implementation needs to be refactored and the impact on the GUI is minimised.

![Diagram of encapsulation](image)

**Figure 34: Creating an object in Composer**

One design pattern used to provide encapsulation is the *Abstract Factory*. Figure 34 depicts the class diagram of this design pattern applied to the edvo::Composer example. An AbstractFactory declares an interface for operations that create other abstract objects. The implementation of concrete object creation is provided by a concrete factory, named GPACFactory. In this case, the objects produced by the abstract factory are SceneRender objects. SceneRender declares an interface that is implemented by the
GPACSceneRender class. Any UI class, represented by UIControl, can use an abstract factory to instantiate any scene handling and rendering object.

Figure 35 shows the relationships between the classes related to scene manipulation and rendering. An UIControl instantiates a SceneRender using the abstract factory process described above. Each SceneRender is associated with a SceneGraph object which is also created using an abstract factory. Finally, the SceneGraph is composed by zero or more SceneNodes and SceneRoutes. These are created only by the SceneGraph and are destroyed when it is destroyed. In all cases the GUI classes only access the objects through the defined interface. Implementation is done by the GPACxxx classes.

![Figure 35: Relationship between classes in Composer](image)

**Server Connection**

As previously said, the connection with the server is established using XML-RPC as the communication protocol. XML-RPC is a remote procedure calling specification [88] that uses HTTP as transport protocol, and XML as the encoding format. XML-RPC is designed to be as simple as possible allowing at the same time the transmission and processing of complex data structures. The choice for XML-RPC as structured communication protocol was mainly due to its lightweight implementation, as opposed to, for example, CORBA. The other decisive argument favoring XML-RPC was the native
support provided by Zope which left aside the other main contender, SOAP.

Following the same principle of encapsulation described previously for scene handling and rendering, the prototype composer hides the details of communication protocols with the server. For that, another design pattern was used – the Proxy – which provides a surrogate or placeholder for an object controlling its access. Figure 36 shows the class diagram of this design pattern applied to the composer. ProxyServer maintains a reference that lets the proxy access the real server, represented in the diagram by ConcreteServer. It provides an interface identical to the one of ConcreteServer, so that a proxy can be replaced by the real server. Remote proxies, as in this case, are also responsible for encoding a request and its arguments and for sending the encoded request to the real server.

![Class Diagram]

**Figure 36: Server interface with composer using a proxy**

With this implementation, changing the way the composer accesses the server is fairly easy since only the ProxyServer class needs to be reimplemented using a new protocol – SOAP, CORBA, etc – or even a new mechanism – calling the server methods from dynamic loaded library.

**Scripting**

Other important characteristic implemented by the composer is the ability of using a scripting language to programmatically change the scene. As stated in Section 3.2.4, scripting is a powerful way of creating multimedia applications, offering a very wide range
of applicability. However, this is a feature mainly oriented to savvy users.

To interpret the scripts, the composer embeds the Python interpreter. When embedding Python, direct access to the Python interpreter is given to the embedding application, providing it the power to load and execute Python scripts and services. This is accomplished using directly Python’s API.

For the embedded Python interpreter to have access to functionality from the application itself, the Python API allows extending the interpreter. This is achieved by extending the embedded interpreter with routines provided by the application.

In this case, the implementation uses the **pygpac** bindings to operate on locally defined C++ classes. Recalling that **pygpac** uses SWIG to wrap the GPAC library, we now expose how this is accomplished. SWIG uses proxy classes to wrap the original classes or structures. Proxy classes are always constructed as an extra layer of wrapping that hold a pointer to the underlying object and dispatches methods and member variable access to that object using low-level accessor functions. The pointer is stored in the Python object as a specially encoded string. Associated with proxy objects, an ownership flag determines who is responsible for deleting the underlying object.

Some glue-code written in C++ initiates the embedded Python interpreter and extends it with some functions that are used to access the application. For example, to access the current SceneGraph a function `getSceneGraph` could return a Python object with the encoded pointer to the original memory structure, which is then used by the wrapper SWIG code to perform operations on it. The simple example shows a script that adds a Transform2D to the root node of the scene, if one is present. Note that the scene graph object is obtained using the expression `composer.getSceneGraph()`, where `composer` is the Python module that has all routines used to access the composer’s functionalities.

```python
import pygpac
sg = composer.getSceneGraph()
root = sg.GetRootNode()
if root.this:
    root.InsertChild (sg.NewNode (pygpac.TAG_Transform2D), -1)
```

The possibilities for the use of scripting in the composition of multimedia programs are endless. All the tools provided by Python can be used to create complex presentations. Some examples are: a script that searches the web or a local disk for some content and automatically builds a presentation showing the results, including text, images, etc; a script that parses some data and builds statistical presentations, showing animated charts;
A MULTIMEDIA AUTHORING FRAMEWORK: EDVO

a script that collects images from several cameras, processes the videos and builds an off-line summarised presentation. And many other examples could be presented. This can be seen as an automated process to build a first draft for a presentation, because after the script has been processed, the presentation can always be modified using the WYSIWYG editor.

5.3.2 Server

The edvo.Server module is responsible for the storage of Media Objects (MOs) and Multimedia Presentations (MPs). It also provides the delivery mechanisms of the multimedia content to the client. This module implements the requirements defined in the previous chapter. It is partly implemented in Python and uses some of the previously described tools, like Zope, as the application engine, the Darwin Streaming Server, providing streaming capabilities, and the pygpac wrapper for MPEG-4 handling.

Storage

The storage of MOs and MPs implemented in the edvo.Server prototype complies with the data model defined in Section 4.4. With respect to metadata, the standard MPEG-7 is used for the description of MOs and MPs; regarding media-data, the original encoding format is used for MOs while MPEG-4 format is used for MPs. The storage of both on this prototype uses a simple method that relies on the file system, i.e., each MO is stored in a folder named using an unique identifier. This folder is itself stored in specific folder for MOs. Similarly, each MP is stored in individual folders identified by an unique identifier which itself is held in a MPs folder. Figure 37 depicts the structure used for the storage of content in the edvo.Server.

Each MO or MP folder contains a file where the content’s metadata is stored and other file containing the media-data. The unique identifier follows the Universal Unique IDentifier (UUID) format [89]. A UUID is an identifier that is unique across both space and time, with respect to all UUIDs. A UUID can be used for multiple purposes, from tagging objects with an extremely short lifetime, to reliably identifying very persistent objects across a network. The generation of UUIDs does not require a registration authority for each single identifier. The mechanism used to guarantee that UUIDs are unique is through the combination of hardware addresses, time stamps and random seeds.

The storage structure contains other folder where published multimedia content is stored. Any presentation stored in this folder is available for browsing and eventually to be transferred to a client application.
Application Engine

Zope is used as the application engine of the server. Commands and requests are sent to Zope using XML-RPC or HTML and are handled by Python scripts. An example of this process is shown in Figure 38. In this case, a module that needs to perform some operation on the repository, first communicates with the edvo.server sending a HTTP request to Zope. Taking the URL associated with the request, Zope queries the object addressed by the URL. Accessing to the repository, as we explain briefly, requires the use of an external method, which then executes a Python script. In this case, the response from the Python script contains, for example, information about a media object. To complete the process, Zope returns to the requesting module a HTML that results from applying a XSL file to the XML response.

Instead of textual, the response could have been binary containing, for example, an image or an audiovisual segment. In this case, the Python Script would return a string
of binary data, and a MIME type identifying the content. The response to the calling module would then be sent back, using HTTP with binary content in its body. This method is applied when the user browses media objects and MPEG-4 tracks, and preview is requested. The latter is done without demultiplexing the MPEG-4 binary file, using the facilities provided by pygpac to read the samples that compose a specific track.

A mechanism of External Methods is needed due to the security constraints imposed by scripts. For example, to read files from disk, or access the network, or use some advanced libraries for things like regular expressions or image processing, the use of External Methods is required. To create and edit External Methods, access to the file-system is needed which provides an important security control. By requiring that unrestricted scripts be edited on the file-system, Zope ensures accessed only to people who are already trusted.

**Encapsulation**

Although encapsulation is not enforced by Python (no equivalent to C++'s protected or private), strict rules were applied to guarantee the separation of functionalities in composing sub-modules. The fact that the Python language is loosely typed also helps in keeping encapsulation and defining generic interfaces consisting of stable methods. By stable we mean that the method signature remains the same.
Repository Handling

Several Python scripts handle the access to the repository. The Python library is used to browse through the repository.

Regarding MOs, media-specific parsers are used to extract structural information about the content. MPs are handled using pygpac which, in this case, is essentially used to browse, create and modify MP4 files.

As for metadata handling, the edvo.Server prototype uses XML direct manipulation with the tools provided by Python. Metadata queries can be made using SQL, XQuery or specific query strings depending on the supporting technology. In this case, simple specific query strings are used. However, the metadata handling and storage can be replaced by more complex and efficient tools like XML-enabled databases (e.g. recent versions of Oracle).

As previously exposed, generic and stable interfaces are enforced, which allows easy replacement of support provided for the metadata handling without changes to higher-level scripts that construct the responses to other modules.

Multimedia Content Delivery

When a multimedia presentation is published, it is stored in a specific folder that can be accessed by the delivery tools. One important delivery mechanism is streaming. As stated in the previous section, the Darwin Streaming Server (DSS) is used to stream content to the client application. However, to aid the DSS, special tracks called hinting tracks are added to the MPEG-4 file. This operation is also performed by pygpac.

5.4 Graphical User Interfaces

In multimedia authoring, Graphical User Interfaces (GUIs) perform an important role. In this section we show some of the graphical interfaces implemented in the edvo prototype. Clearly edvo.composer is the most GUI-intensive and thus we will only focus on it.

Figure 39 shows the layout of the edvo.composer prototype. The following elements are labeled:

- 1 – Scene Explorer containing a tree structure representing the current scene hierarchy. Using the tabs below the Scene Explorer, the user can browse the routes and the commands composing the multimedia presentation. Copy, cut and paste mechanisms can be used to change the node hierarchy.
• **2** – WYSIWYG editor that allows the manipulation in space of each visual object using simple click and drag methods. Copy, cut and paste mechanisms are also available as well as z-order positioning.

• **3** – Node properties browser that allows easy editing of each field of the selected MPEG-4 node.

• **4** – Shortcut toolbar that contains shortcuts to ease the creation of the multimedia presentation.

Changing the properties of each MPEG-4 node is fairly intuitive and is immediately reflected by the scene editor, as seen in Figure 40, where the user changes the colors of a LinearGradient node and in Figure 41, where a node containing some text is changed.

In Figure 42 the process of adding a route is shown. First, a dialog containing the scene nodes and fields in a tree structure is shown, where only the available out events are enabled. After selecting an out event, a similar dialog is shown for the selection of a in field or event. This time only the ones having the same type of the out event are enabled.

Browsing the repository is also possible, as depicted in Figure 43. In this case the multimedia programs available in the server are shown. This figure also shows the integration of the protocols used to communicate with the server. The left frame of the search window is a EMozilla control showing some information sent by the server in HTML, and related to the MP selected in the right frame. This frame shows a list of multimedia presentations sent by the server after a XML-RPC call and finally rendered by a FOX control.

The administration of the edvo.server is performed using a web browser. Figure 44 shows the user browsing a multimedia presentation, in this case a MP4 file. It is depicted the track browser, showing some information related to the track and a preview. In this case, the track contains a video encoded in MPEG-4 Visual.
Figure 39: Composer – main window

Figure 40: Composer – changing the gradient colors
Figure 41: Composer – changing the text

Figure 42: Composer – linking a route
Figure 43: Composer – browsing the repository

Figure 44: Server – browsing a multimedia presentation
Chapter 6

Summary and Conclusions

We have reached the terminus. Until arriving here we have studied the available technologies for multimedia representation and description, presented an overview on the state-of-the-art of multimedia authoring, proposed a specification for a generic multimedia authoring system and described a prototype system developed to assert the specification. Hopefully the path has not been too tortuous and the many crossroads pointed along the way did not led to dead-ends. This chapter wraps up what preceded it and points to the future.

6.1 Summary

With current technology each and every one of us is a potential multimedia content creator. However, experiencing the content is not as universal as we would probably expect.

The MPEG-4 standard provides the means to fill some of the current gaps. Unlike other technologies, like the popular format Flash, MPEG-4 is not focused solely on the Web. It aims reaching all end-user devices, with special attention to mobile devices. Besides MPEG-4 and Flash, other multimedia representation technologies include SMIL and X3D.

With multimedia content being produced at a fast pace, the question of how one can effectively process and search the growing offer of content is immediately raised. Additional information, usually called metadata, is necessary. To address this question the MPEG-7 standard was created. MPEG-7 defines a wide set of description schemes for multimedia content. Other technologies can also be used to describe multimedia content like Dublin Core and RDF, among others focused on more specific domains. MPEG-7, however, has been developed to be as generic and extensible as possible.
SUMMARY AND CONCLUSIONS

The proposed objective for this project was the development of an authoring framework that would allow the edition and the description of multimedia content. The creation of multimedia content essentially consists in the combination of media objects that result in a coherent, possibly interactive and, ultimately, appealing presentation to the consumer. Most authoring systems can be classified according to a number of paradigms: structure, timeline, flowchart and script. Each has its advantages and disadvantages concerning learning curve, expressibility and capabilities. In some cases the combination of some of them can overcome the individual limitations.

The current offer of MPEG-4 authoring tools is still somewhat incipient. Few commercial applications are available and the ones available have a diminished visibility. Recent research projects, such as MPEG-4 Toolbox [49] and MPEG-Pro [53] focused on multimedia authoring, solely in the perspective of creating MPEG-4 content.

A new approach was proposed in this dissertation for the handling of media objects which are used to create multimedia presentations. This approach combines the modularity and scalability of MPEG-4 with the flexibility of MPEG-7 for providing suitable descriptions. Several system requirements were defined, as well as the most important use cases. The main requirement is that the system should allow a group of authors to create, manipulate, reuse and repurpose multimedia in a modular, scalable and distributed architecture. Five subsystems comprised the system: Authentication Manager (identifies the system's users, whether they are authors, publishers or consumers), Storage Manager (stores media objects and multimedia presentations and its descriptions, in a shared environment), Authoring Tool (allows the manipulation of media objects in order to create multimedia presentations), Profile Manager (manages user and device profiling information) and Multimedia Service Provider (provides the means to deliver content to the end-user). A functional system architecture was also proposed, as well as the data model for both media-data and metadata.

To assert the specified requirements, a prototype code-named edvO was implemented using MPEG-4 and MPEG-7. It also provides scripting capabilities for content authoring. The prototype consists of four modules: Composer, Admin, Server and Client. During implementation, encapsulation of functionalities in independent packages was applied. These packages are accessed through generic and well defined interfaces, allowing a platform and technology independent implementation.
6.2 Application of the Work Done

Developing this work was a challenging task that involved the integration of many concepts, technologies and tools. Few projects aim reaching such large spectrum, that crosses many areas related to multimedia authoring. The legacy left by it is especially important by this fact.

From the beginning, the integration with open-source projects was considered a fundamental piece of this project. Many enhancements and new tools, created specifically for this project, can be merged in the original projects, helping to its growth and contributing to its stability. This is especially true for the GPAC project with which feedback endures. GPAC is a promising project in an area where the interest has not yet reached the desired level. Although there is a large interest in MPEG-4 around the world, it is taking some time for widespread adoption. Maybe because it is such a wide-ranging standard and there is some uptake in understanding its capabilities and licensing. When tools as the ones developed in this project, showing the potential of MPEG-4 as a multimedia standard and usable in a wide range of environments, maybe the interest may begin to sprout. If the industry is somewhat lethargic maybe the open-source community can give it a push.

The integration of MPEG-4 with MPEG-7 in a multimedia environment has just started. This is a difficult task due to the wide range of applications also targeted by MPEG-7. Supporting all descriptors may result in an unsustainable implementation. Recent efforts, carried out by the MPEG-7 team, try to define profiles and levels of utilization [90]. The main objective is to limit the use of description tools, in order to support a specific class of applications, with different levels of complexity. It is expected that the distinction in areas of application and levels of complexity may bring some clarity within MPEG-7. The integration of MPEG-7 with MPEG-4 – metadata with content – is therefore one important contribution of this project to the area of multimedia authoring.

As thoroughly mentioned in the previous chapter, care has been taken to ensure that changing one part of the implementation would require minor changes to the rest of the architecture. Thus, the prototype can also act as a test bed for the implementation of other multimedia authoring and description technologies. Although the system was specified considering the use of the MPEG technologies there is no constraint that withholds the use of other supporting technologies.

This was a very rewarding project to work on, mainly because it was a broad and transversal project. Even though, the reward will be greater if it proceeds further, contributing more to the dissemination of multimedia technologies, especially MPEG-4 and
SUMMARY AND CONCLUSIONS

MPEG-7.

6.3 Further Developments

The association of description with multimedia authoring will allow in the future the emergence of a semantic system. Semantic authoring allows content to be adapted, regarding available resources (specially important in heterogeneous systems with different network topologies and terminal characteristics), and user interaction. This work is a further step toward this goal. In order to evaluate these ideas and to provide feedback for an improved specification, a prototype was implemented – edVO.

We analyse which paths are still open regarding the implementation of the current prototype and the integration with other projects and concepts.

6.3.1 Implementation-wise

To allow fast prototyping, the system was implemented using off-the-shelf libraries and tools. Moreover, the prototype was also designed to provide a test bed for other technologies and implementations. However, to reach a stable application the software meanwhile developed should evolve from the current state of prototype. For that, more stages need to be covered. Figure 45 shows the spiral model of software evolution. It is a cyclic process that iterates through four distinct stages: software cycle!Specification, Implementation, Validation and Operation.

![Software evolution model](image)

Figure 45: Software evolution model
Until now, only the first iteration of the specification and partly the implementation and validation have been completed. Thus, a first operable release version is not yet available. To advance to future developments, the prototype should be operated during a test period by users, giving eventually some input to a new specification.

The immediate efforts should be on resolving some pending issues, like adding support for 3D objects, for multiple scenes and for MPEG-4 PROTO nodes. The current semi-functional timeline also needs to be further developed to help the manipulation of objects in time.

The support for searching and filtering of metadata is rather simple, yet functional. Its simplicity can constitute a virtue but if more complex features are required it can be easily updated to other supporting technology. Regarding it, the use of, for example, relational databases for the storage of MPEG-7 descriptors is a possibility to be studied in the future.

6.3.2 Integration-wise

The information gathered along the creation process can be very valuable. The most obvious gains are for the search and retrieval of multimedia content stored in databases [91]. This information provides important clues to the search engine. The edVO prototype uses mainly semantic information that is inserted by the authors. But, besides semantic information, low-level description of content could also be used, providing fine-tune controls to the search engine. Adding a plug-in for automatic extraction of low-level features of both visual and audio objects could bring improved searching capabilities. This is in fact an area where research has been very active ([92] and [93] just to mention some examples).

Additional information can also be used to develop multimedia systems that offer a higher level of interactivity than the one offered currently. When interacting with a multimedia presentation, comprising individual media objects, if visual, audio or semantic description is provided per object, the content becomes itself more interactive and cognitive. Evaluation of user interactivity [94] is an important research area to attain improved user experiences.

Other important area is the adaptation of content. Content adaptation comprises two topics: adaptation to variable resources (terminal/network) and adaptation to users. Using additional information alongside the multimedia content, assists content adaptation engines. The concept of multimedia adaptation leads to a broader one called Universal Multimedia Access (UMA). The objective of UMA technology is to make available different presentations of the same information, more or less complex, for example, in terms of...
SUMMARY AND CONCLUSIONS

media types, suiting different terminals (PCs, PDAs, mobile phones, interactive television, other mobile devices, etc), networks (Ethernet, 802.11, Bluetooth, UMTS, etc) and user preferences [95]. The MPEG-7 description attached with the multimedia content created with the edVO system provides the means to accomplish this. Also, the MPEG-4 object-oriented and scalable architecture plays an important role when integrating edVO with content adaptation engines. In other words, the system provides the adaptive content and the clues for adapting it, which can then feed an UMA Engine.

It can be said that, whereas the information used for search and retrieval and for interactivity enhancement can be perceived by the end-user, the one used for content adaptation is transparent to the end-user.

In the area of content management and adaptation, one of the current research projects, with which INESC Porto collaborates, is ENTHRONE [96]. The ENTHRONE project aims at providing a solution which covers an entire audio-visual service distribution chain, including content generation and protection, distribution across networks and reception at user terminals. It is expected to support several encoding formats, including MPEG-4, as well as MPEG-21 for content management. Considering this example, the edVO multimedia authoring system can be integrated with a broader system, like ENTHRONE. In this case, the support for MPEG-21 digital item management framework within the edVO publishing module, could be the next step toward integration. No other changes would be necessary.

6.4 Conclusion

Multimedia Authoring and Description are challenging areas that require the integration of multiple technologies from different fields.

In this project we have specified a modular, scalable, distributed and flexible multimedia authoring framework for the edition and description of media objects. A group of authors can use the framework to create, manipulate and repurpose multimedia presentations. The composition of presentations is accomplished using media objects. During the creation process, an author can associate additional information, or metadata, to presentations and to media objects. This process is performed in a shared, distributed environment, providing an open, flexible and scalable architecture.

A prototype was developed as proof of concept using the MPEG-4 and MPEG-7 standards to represent and describe multimedia content, respectively. However, other technologies can also be supported by the prototype. Additionally, the prototype includes
scripting capabilities which provide powerful tools to create complex multimedia content. The development of a full-fledged system is an ongoing task and the integration with other projects comprising automatic description of multimedia content, multimedia management and adaptation for Universal Multimedia Access is foreseen.
List of References


LIST OF REFERENCES


[31] ——, "Scalable Vector Graphics (SVG) 1.0," http://www.w3.org/TR/SVG/.


LIST OF REFERENCES


LIST OF REFERENCES


[86] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, 1st ed. Addison Wesley, 1995.


Index

Symbols
3D graphics, 22
3D objects, 22
3GPP, 83
4Mation, 48

A
AAF, 34
access conditions, 12
adaptive content, 18
Administration Tool, 62
Analysis, 3
Apache, 82
Archiving, 3
AU, 17
authentication, 52
Authentication Manager, 54–56
authoring framework, 98
authoring paradigm, 38
  – flowchart, 39
  – script-based, 41
  – structure-based, 39
  – timeline, 40
Authoring Tool, 54, 58–60, 62
authoring tools, 11
Authorware, 50

B
BIFS, 14
  – update commands, 15
broadcast environment, 34

C
Capture, 3
Client Application, 62
Coding, 10
Commission, 3
Complete Description, 66
component testing, 74
component-based systems, 73
composite objects, 13
Composition, 3, 10
Consumption, 3
content
  – cataloguing, 2
  – creation, 7
  – natural, 13
  – searching, 2
  – synthetic, 13
content adaptation, 101
content creation
  – production, 12
  – publishing, 12

111
INDEX

content representation, 12
content ubiquity, 7
CORBA, 85, 86

D
Darwin Streaming Server, 83
data model, 64
DC, 32–33, 97
DDL, 30
Delivery, 3
delivery layer, 19
Description Schemes, 30
Description Tools, 30
Description Unit, 66
Descriptors, 30
descriptors
  – low-level, 9
design pattern, 84, 86
  – abstract factory, 84
  – proxy, 86
digital item, 64, 102
Director, 48
distributed architecture, 51
DMIF, 19

E
edVO, 73, 98
edVO.Admin, 75
edVO.Client, 75
edVO.Composer, 75
edVO.Server, 75
Elaboration, 3
elementary stream
  – data, 17
EMozilla, 81–82, 92
encapsulation, 84, 90
enhanced content, 7
ENTHRONE, 102
ES, 12
ES Management, 77
event sink, 15
event source, 15
exchangeable format, 15
External Methods, 90

F
features
  – high-level, 28
  – low-level, 28
Flash, 23, 50, 97
FlexMux, 19
FOX Toolkit, 79–80
FTP, 64, 82
FXMozilla, 81

G
Gecko, 80
GPAC, 76–79
GTK, 80

H
HTML, 75
HTTP, 64, 75, 82

I
Interaction, 3, 10
interactive multimedia content, 9
interoperability, 7
IPMP, 17

L
legacy systems, 74

M
INDEX

MAXPEG Author, 48
MDS, 31
media item, 37
media object, 8, 12, 52, 88
– primary, 12
media-data, 52, 65
metadata, 52, 65
– interoperability, 35
MIME type, 90
MO Handling, 57
mobility, 7
Mozilla, 80–82
MP Handling, 57
MP4 file format, 20
MPEG-21, 9, 102
MPEG-4, 2, 9–20, 97
– authoring tools, 46
– handling, 88
– programs, 12
– scene, 12
MPEG-4 and MPEG-7
– joint use, 3
MPEG-4 Toolbox, 46
MPEG-7, 2, 28–32, 97
MPEG-7 descriptions, 28
MPEG-Pro, 47
MPEG4IP, 83
multimedia, 2
– information, 2
– timed-based, 1
multimedia authoring
– system, 38
– tools, 38
multimedia authoring framework, 5
multimedia authoring requirements
– authoring capabilities, 45
– expressive power, 43
– interactions, 44
– media objects, 43
– temporal composition, 44
multimedia authoring system, 43
multimedia content, 7, 11
– describing, 28
– searching, processing and filtering, 9
multimedia presentation, 1, 14, 37, 52, 83,
88
Multimedia Server, 62
Multimedia Service Provider, 55, 60–61
Multiplex, 10
MXF, 34
N
natural language, 33
O
object
– edition, 13
– extraction, 13
OCI, 17
OD, 15
P
Packaging, 3
Perl, 78
primary objects, 13
Profile Manager, 54, 60
profiling, 52
prototype, 73
publishing process, 60
Publishing Tool, 62
pygpac, 78–79, 87, 88
Python, 78, 87

113
INDEX

Q
QoS, 10, 17, 19
QT, 80

R
RBAC, 55
RDBMS, 82
RDF, 33–34
Repository, 62
rich-media, 7
routing mechanism, 15
RTSP, 64, 75
Ruby, 78

S
scene
   – commands, 19
   – description, 19
scene description, 9
scripting language, 78
SL, 17, 19
SMIL, 9, 20–22
SOAP, 86
software cycle
   – Implementation, 100
   – Operation, 100
   – Specification, 100
   – Validation, 100
SQL, 91
Storage Manager, 54, 57–58
streaming protocol, 20
streaming servers, 20
SVG, 9, 21–22
SWF, 23
SWIG, 78
Synthesis, 3
system actors
   – Administrator, 53
   – Author, 53
   – Client, 53
   – Network, 53
   – Publisher, 53
   – User, 53
system architecture, 62
system requirements, 62, 73
system testing, 74
System Tools, 30

T
TCL, 78
television paradigm, 8
TransMux, 19

U
UMA, 101
URI, 64
user interfaces, 70
UUID, 88

V
VideoLAN, 83
VRML, 22

W
W3C, 9
Web3D, 9
WebDAV, 82
wxWidgets, 80

X
X3D, 9, 22–23, 97
XHTML, 22
XML-RPC, 75, 82, 85, 92
XMT, 15
XMT-Ω, 15
XMT-A, 15
XPCOM, 82
XQuery, 91

Z
Zope, 82–83