**FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO**

# MPEG21 DI Browser, an MPEG-21 based architecture for the consumption of Digital Items

Giorgiana Ciobanu

Licenciada em Engenharia de Sistemas e Computadores
pela Faculdade de Automatizações e Computadores
da Universidade Politécnica "Gh. Asachi" de Iaşi, Roménia

Dissertação submetida para a satisfação parcial dos
requisitos do grau de mestre
em
Tecnologia Multimédia,
Perfil Engenharia

Dissertação elaborada sob a supervisão de
Professor Doutor Eurico Manuel Elias Morais Carrapatoso
Departamento de Engenharia Electrotécnica e de Computadores,
Faculdade de Engenharia da Universidade do Porto

2006

## Dedication

*To my husband.*

## Acknowledgements

There are some persons who have contributed one way or the other to make this dissertation possible. I would like to express here my gratitude to them.

I wish to thank Professor Eurico Carrapatoso, supervisor of this dissertation, for his advice, guidance and patience.

Thanks to INESC Porto for all the useful information that they provided to me and also for sharing with me their experience.

Special thanks to Pedro Carvalho for the valuable suggestions, support, encouragement and for his help.

And last, but not least, many thanks and a warm gratitude to my husband Lucian, for his patience and constructive criticism, for teaching me how to find the optimized solutions and also for helping me to gain experience in Java.

## Abstract

MPEG-21 is an emerging standard that aims to define a normative open framework for multimedia delivery and consumption. Content creators, producers, distributors and service providers are the main benefiters from this framework as well as the MPEG-21 enabled open market. In the case of content consumers, MPEG-21 enables their access to a large variety of multimedia content in a flexible and interoperable manner. For the moment, the MPEG-21 framework lacks of tools to be used in content consumption.

The main goal of this thesis was to meet the need for an MPEG-21 terminal application used for processing and presenting the multimedia contents organized as Digital Items. The work for producing this dissertation was structured in two phases. First, a state-of-the-art of the MPEG-21 standard and an analysis of the most popular European projects, applications and terminals that are implementing the MPEG-21 concepts, were carried out. Then, in the second phase, a proposal and description of the development of a client-server architecture for a MPEG-21 terminal used for Digital Items consumption were presented.

The proposed application aims to enable the interoperability, accessibility, portability, flexibility and dynamic interactivity with multimedia contents, carried in Digital Items, in conformance with the MPEG-21 standard. These features lead to an easy integration with other systems and add a friendly user graphical interface provided via Web that makes it a challenging alternative to existing tools for visualizing large amounts of multimedia contents and meeting the ever increasing user demands.

## Resumo

O MPEG-21 é uma norma emergente que pretende definir uma plataforma aberta e normativa para a distribuição e o consumo de conteúdos multimédia. Criadores de conteúdos, produtores, distribuidores e fornecedores de serviços multimédia são os principais beneficiários desta plataforma bem como o mercado aberto do MPEG-21. No caso dos consumidores de conteúdo, o MPEG-21 permite o acesso a uma grande variedade de conteúdos multimédia de uma forma flexível e interoperável. Presentemente, a plataforma MPEG-21 tem poucas ferramentas desenvolvidas para o consumo de Digital Items.

O objectivo principal desta tese era de responder à necessidade de uma aplicação tipo terminal MPEG-21 para processar e apresentar conteúdos multimédia organizados como Digital Items. O trabalho para elaborar esta dissertação foi estruturado em duas fases. NA primeira foi realizado o levatamento do estado da arte para a norma MPEG-21 e depois foi feito um estudo das aplicações MPEG-21 mais conhecidas que implementam os conceitos da MPEG-21 . Na segunda fase foi proposto e descrito o desenvolvimento de uma arquitectura cliente-servidor para um terminal MPEG-21 usado para o consumo de Digital Items.

A aplicação proposta pretende fornecer suporte para interoperabilidade, acessibilidade, portabilidade, flexibilidade e uma interacção dinâmica com os Digital Items conformes com à norma MPEG-21. Estas características conduzem a uma integração fácil com outras sistemas e acrescentam uma interface gráfica intuitiva e amigável para o utilizador fornecida via Web, que tornam esta aplicação numa alternativa que lança um desafio às ferramentas recentes para visualizar conteúdos multimédia atendendo às necessidades dos utilizadores.

## Rezumat

MPEG-21 este un standard recent care urmăreşte să definească o platformă deschisă şi normativă pentru distribuţia şi consumul de conţinut multimedia. Creatorii de conţinut, producătorii, distribuitorii şi furnizorii de servicii multimedia sunt principalii beneficiari ai aceastei platforme precum şi piaţa deschisă MPEG-21. În cazul consumatorilor de conţinut, MPEG-21 permite accesul lor la o mare varietate de conţinut multimedia de o manieră flexibilă si interoperabilă. În prezent, plataforma MPEG-21 duce lipsa unor aplicaţii care să fie folosite pentru consumul de Digital Items.

Obiectivul principal al acestei teze este de a răspunde necesităţii pentru o aplicaţie de tip terminal MPEG-21 pentru procesarea si prezentarea conţinutului multimedia organizat sub formă de Digital Item-uri MPEG-21. Munca depusă pentru elaborarea acestei dizertaţii este structurată în două faze. În prima fază s-a realizat un „*state-of-art*" al normei MPEG-21 iar apoi s-a făcut un studiu al aplicaţiilor mai populare care implementează conceptele din MPEG-21 În faza a doua este propusă şi implementată dezvoltarea unei arhitecturi client-server pentru un terminal MPEG-21 folosit în consumul de Digital Item-uri.

Aplicaţia propusă urmăreşte să asigure interoperabilitatea, accesibilitatea, portabilitatea, flexibilitatea si o interacţiune dinamică cu Digital Item-uri in conformitate cu norma MPEG-21. Aceste caracteristici conduc la o integrare uşoară cu alte sisteme şi asigură o interfaţă grafică prietenoasă cu utlizatorul, furnizată via Web, făcând din această aplicaţie o alternativă competitivă la uneltele existente pentru vizualizarea de cantităţi enorme de continut multimedia satisfăcând necesităţile din ce in ce mai mari ale utilizatorului.

## Preface

This document presents the study and the work in the area of a specific multimedia technology that is MPEG-21, in order to obtain the academic master's degree title.

The idea for this thesis came out in the Summer of 2005 after a collaboration with INESC Porto (Institute for Systems and Computer Engineering of Porto) with implications in some internal projects but mainly related with the work in the European project ENTHRONE. ENTHRONE (abbreviation for "*End-to-End QoS through Integrated Management of Content, Networks and Terminals*") is an Integrated Project in the Thematic Priority 'Information Society Technologies' of EU Framework Programme 6 for Research and Development (IST-507637). A high-level goal of the ENTHRONE project is "*to bridge the divide between the content provision and the networking worlds, resulting in cross-industry co-ordination on both network and content management issues, and bringing focus to mutually advantageous standards such as MPEG-21*"[1]. Within the mentioned project, besides other contributions, INESC Porto implemented a MPEG-21 terminal for consuming Digital Items named INESC DIDBrowser; it was a Java desktop application for visualizing multimedia resources by navigating through MPEG-21 Digital Item Declaration content. Although it was created in Java to be a desktop application, it was built following the philosophy of Web pages; this fact revealed the possibility to transform the INESC DIDBrowser into a real Web application. But there was a pending issue regarding the application portability due to the fact that the Java used to implement INESC DIDBrowser is not compatible with all types of terminal devices. Moreover, in the case of "thin" devices such as PDAs or mobile phones, there are some resource limitations and poor performance that affect the support for complex processing of Digital Items. For these reasons the functionality of the INESC DIDBrowser can not be available entirely on such systems.

In this context, the idea to create a new application for processing and "browsing" Digital Items emerged, but this time it should be totally suitable for the Web and required a new architecture. The solution was to adopt a client-server Web service architecture where the processing of Digital Items is the responsibility of the server application and where the content visualization and the management of user interactions with the content is done by the client. The dynamic and direct interaction of users with the Digital Items is ensured by implementing part 10 of the MPEG-21 – Digital Item Processing, that specifies the necessary

methods and mechanisms to support this. The aim was to simplify and centralize the processing of Digital Items providing a common basic functionality designed for all types of user devices. This will increase the efficiency of the processing part. At the moment of writing this thesis there are no other similar MPEG-21 applications to do that. As MPEG-21 is a recent standard, still under development but with a promising future, and due to the very few applications for visualization of Digital Items content, it came out the motivation to implement such a system as it was proposed in this dissertation.

The only constraint encountered during the realization of the thesis was the limitation of time to put into practice all the ideas and some enhancement to be done to this work. For this reason the work was divided in two phases, where the second phase is considered future work.

# Table of Content

# List of Figures

# List of Tables

# List of Acronyms

| | |
|---|---|
| **API** | Application Programming Interface |
| **DI** | Digital Item |
| **DIA** | Digital Item Adaptation |
| **DIBO** | Digital Item Basic Operation |
| **DID** | Digital Item Declaration |
| **DIDL** | Digital Item Declaration Language |
| **DII** | Digital Item Identification |
| **DIM** | Digital Item Methods |
| **DIP** | Digital Item Processing |
| **DIXO** | Digital Item eXtended Operation |
| **DOM** | Document Object Model |
| **EC** | European Commission |
| **ER** | Event Reporting |
| **FTP** | File Transfer Protocol |
| **GUI** | Graphical User Interface |
| **HTTP** | Hyper Text Transfer Protocol |
| **IPMP** | Intellectual Property Management and Protection Components |
| **ISO/IEC** | International Organization for Standardization / International Electrotechnical Commission |
| **MIME** | Multipurpose Internet Mail Extensions |
| **MPEG** | Moving Picture Experts Group |
| **PC** | Personal Computer |
| **PDA** | Personal Digital Assistant |
| **QoS** | Quality of Service |
| **RDD** | Rights Data Dictionary |
| **RefSW** | Reference Software |
| **REL** | Rights Expression Language |
| **RFCxxxx** | Request for Comments (xxxx document number) |
| **SMIL** | Synchronized Multimedia Integration Language |
| **SOAP** | Simple Object Access Protocol |
| **UMA** | Universal Multimedia Access |
| **UML** | Unified Modeling Language |
| **URI** | Uniform Resource Identifier |

| | |
|---|---|
| **URL** | Uniform Resource Locator |
| **W3C** | World Wide Web Consortium |
| **WAP** | Wireless Application Protocol |
| **WML** | Wireless Markup Language |
| **WSDL** | Web Services Description Language |
| **XML** | eXtensible Markup Language |
| **XSL** | eXtensible Stylesheet Language |
| **XSLT** | eXtensible Stylesheet Language Transformations |

# Terms and Definitions

**Digital Item (DI):**

Structured digital object, including a standard representation, identification and metadata within the MPEG-21 framework.

**Digital Item Basic Operation (DIBO):**

Base operation providing access to functionality implemented within an MPEG-21 environment and used in authoring a Digital Item Method.

**Digital Item Declaration (DID):**

Declaration of the resources, metadata and their interrelationships of a Digital Item specified by part 2 of MPEG-21.

**Digital Item Declaration Language (DIDL):**

XML-based language including validation rules specified by part 2 of MPEG-21 for the standard representation in XML of a Digital Item Declaration.

**Digital Item Declaration Language element:**

XML element of the Digital Item Declaration Language specified by part 2 of MPEG-21.

**Digital Item eXtended Operation (DIXO):**

Operation allowing extended functionality to be invoked from a Digital Item Method.

**Digital Item Method (DIM):**

Tool for expressing the suggested interaction of a User with a Digital Item at the level of the Digital Item Declaration.

**Digital Item Method Language (DIML):**

Language providing the syntax and structure for authoring a Digital Item Method utilizing the Digital Item Base Operations.

**Digital Item Processing Engine (DIP Engine):**

Component within an MPEG-21 environment that supports part 10 of MPEG-21 and is responsible for providing such supporting functionality (including execution of Digital Item Methods).

**MPEG-21:**

A multimedia framework enabling transparent use of multimedia over a variety of networks devices and Users.

**Peer:**

Device or application that compliantly processes a Digital Item.

Note: “Terminal” is usually avoided within the MPEG-21 documents because of its connotation as being the end point in a chain of communication. Besides such applications, the term Peer explicitly also includes devices or applications that create or alter Digital Items, and that handle Digital Items “in transit”.

**User:**

Entity that interacts in the MPEG-21 environment or makes use of Digital Items

Note: The definitions are extracted from MPEG-21 documents [2] and [3].

# Chapter 1

## 1. Introduction

Currently, due to the technological evolution and the existent communications infrastructure, there is a growing necessity for multimedia content consumption and for easier access to information. Applications have been developed to satisfy the user requirements in this sense. However, some issues have to be met: authors and content owners whish to see their rights protected; and users wish to transparently access content through heterogeneous networks and terminal devices. Can all these requirements be simultaneous satisfied?

The availability of suitable descriptions about the content and the context of usage is presently one of the requirements to be addressed during the presentation and consumption of multimedia contents. Likewise, the use of common and open formats to store and transact contents and descriptions, while providing easy identification and use of both, is also necessary. Other issues to be addressed when accessing and consuming the information are related with the protection of the intellectual property management and protection and also with the protection of the multimedia content by dealing with the licenses.

### 1.1. Context and motivation

The current research activities are oriented towards finding new solutions for the realization of standards to enable the transparent and universal access in the use of multimedia resources (this is known as Universal Multimedia Access - UMA). New standards have to guarantee interoperability by focusing on how the elements of a multimedia application infrastructure should relate, integrate and interact. Interoperability is a very important aspect because it facilitates the exchanging of any type of information without technical barriers among users.

Among the available emerging standards for multimedia content representation, such as MPEG-7 and TV-Anytime, MPEG-21 is one of the most advanced, complete and flexible, and tries to find optimized solutions for all the previously mentioned requirements and

aspects. It tries to apply these solutions to the development of a truly interoperable multimedia framework that will enable users to access the multimedia resources in a transparent way. MPEG-21 is defining not only the means for expressing content description but is also describing the usage context for the respective content.

The user's goal is to get to the multimedia information using a variety of terminals with a wide range of capabilities, existent on different devices. Nowadays, one of the reasons for the networks popularity is the information accessibility that they offer: the consumers now can reach the multimedia contents from almost anywhere. This is a challenge for the development of new standards, technologies and business models, and therefore the new generation of tools and terminals for the multimedia content consumption tries to be more oriented for the Web.

The proposal for this dissertation was to meet some of these needs for new solutions in the usage, visualization, interaction with and description of the multimedia contents across various types of networks and heterogeneous terminals, in conformance with the MPEG-21 standard. The MPEG-21 framework does not specify how a terminal application visualizes the containers of multimedia contents especially in a friendly manner. In this context, the implementation of such system, suitable for the Web, is very motivating. It should also permit the correct validation and processing of Digital Items.

## 1.2. Objectives

The main objective for this thesis was to study, propose and develop an efficient and flexible architecture for an MPEG-21 terminal application used in the consumption of multimedia content represented conform with this standard.

The aim was to implement a user friendly, Web accessible and interoperable system that would provide browsing, dynamical interaction with, and processing of multimedia content on any type of device, in order to give to the ordinary user an adequate, visual and very useful perception of the multimedia resources

The partial goals were:

1. Study the MPEG-21 standard.
2. Produce a state of the art survey of similar applications implemented for browsing the multimedia information represented in conformance with MPEG-21.
3. Specify and develop a Web service based system for processing, visualizing of and interacting with multimedia content, implementing the functionalities of parts 2 (Digital Item Declaration) and 10 (Digital Item Processing) of MPEG-21.

### 1.3. Structure of the thesis

This document is divided into seven chapters. Chapter 1 presented the proposed objectives of the thesis work and the context in which it is inserted. Chapter 2 introduces MPEG-21 and describes its key concepts. A state of the art survey of the implemented applications and European projects using MPEG-21 concepts is presented in Chapter 3. Chapter 4 describes the MPEG-21 terminal applications used for multimedia information consumption and creation that were implemented up to the present. Chapter 5 identifies the requirements, includes an analysis of the proposed architecture organized in several modules and their functionalities, and specifies the system to be implemented by using UML diagrams. Chapter 6 describes the implementation of the MPEG21 DI Browser and also illustrates the user's perspective of the developed system. The results from testing the system and an evaluation of these results are described in Chapter 7. The final chapter (Chapter 8) concludes the thesis with some remarks on the MPEG-21 framework and on the work developed, and also presents some suggestions for future work.

Annex A provides an example of a Digital Item Declaration in the context of MPEG-21. Annex B contains some essential terms definitions for part 5 of the standard (Rights Expression Language). Annex C briefly defines the description tools for Digital Item Adaptation. Annex D introduces the usage of Digital Item Processing information within Digital Items. And the last two annexes, E and F, contain the API of the main processing modules of the MPEG21 DI Browser that were detailed in Chapter 6.

# Chapter 2

## 2. Introducing MPEG-21

The Moving Picture Experts Group (MPEG) is a working group of ISO/IEC responsible for the development of standards for coding audio and video content in a digital compressed format. MPEG serves as the core technologies for digital interactive media distribution, broadcast, contribution and playback systems for digital television, computer, communications and consumer industries. MPEG's official designation is ISO/IEC JTC1/SC29 WG11.

MPEG group was founded in May of 1988 and since then has produced several standards: MPEG-1 [5] in which are based products such as Video CDs and MP3 (MPEG-1 audio layer 3 is the full name for the audio format MP3); MPEG-2 [6], used to encode audio and video for broadcast signals and which is also the coding format used by standard commercial DVD movies; MPEG-4 [7] focusing on the CD and web (streaming media) distribution and broadcast television, addressing a large range of applications; and MPEG-7 [8] for description and search for audio and visual content.

These standards MPEG-1, -2, -4, and -7 provide a complete set of tools for multimedia representation. Other related standards exist, such as JPEG and JPEG 2000, that are oriented towards the image compression. But the more recent multimedia applications require more than these standards. The problem is that the multimedia content consumption and transaction remain nontransparent and a new standard is required to solve that.

This chapter briefly presents the essential concepts and the structure of the recent standard, named MPEG-21.

### 2.1. MPEG-21

In late 1999 and early 2000, MPEG examined several questions: Do all existing multimedia-related standard specifications fit together and how do they fit together? Are there specifications for all the necessary technical elements for multimedia transactions? Who is

making the "glue" that will let these standards fit together? The conclusion was that there is an urgent need to develop a common multimedia framework that would facilitate a more efficient interaction between different communities and applications supporting their individual needs and contributing to the enhancement of the user's experience. In this context, in June of 2000, MPEG started to work on the new standard - MPEG-21, known as ISO 21000, that intends to define an open framework for multimedia applications.

MPEG-21 is an XML based standard and aims to describe how the various elements of the multimedia content delivery chain fit together to create a truly interoperable multimedia framework [2]. Whereas standards such as MPEG-7 and TV-Anytime focus on how to describe the content, MPEG-21 goal is to describe the context in which the content is delivered and consumed. MPEG-21 defines the description of content and also defines methods for processing, accessing, searching, and storing it, and for protecting the copyrights of content and the licenses referring to the rights of both creator and consumer.

## 2.2. Digital Items and Users

The essential concept of MPEG-21 is the Digital Item (DI), representing the fundamental unit of transaction, consumption and interaction of users with multimedia content. A Digital Item was defined as "a structured digital object with a standard representation, identification and meta-data within the MPEG-21 framework" [2].

A Digital Item is a combination of resources (e.g., videos, audio or images) of different types of information (photo albums, web pages, e-books, digital library, company's product catalogue, etc.) and metadata (such as descriptors and identifiers) and at the same time it is a structure for describing the relationships between these resources. It may contain information specific to other standards (e.g., MPEG-7, TV-Anytime) for describing the content.

Nowadays, it is becoming difficult to legitimate users of content, and to identify and interpret the different intellectual property rights that are associated with the elements of multimedia content. For this reason, new solutions are required for the access, delivery, management and protection of this content. In order to get the solution for this problem, a Digital Item may provide information related to the protection and intellectual property of the contents, to a possible assisted adaptation of the Digital Item, to the user configurations, the action response to a determined event, etc.

An example of a Digital Item might be a presentation of a university including various types of multimedia content: photos, videos, animation graphics and textual information, news related to the university's research activities, e-learning material and so on (see Figure

1). This Digital Item may interact with various types of users: students, graduates, staff and visitors, each user category having different rights and permissions on these resources.

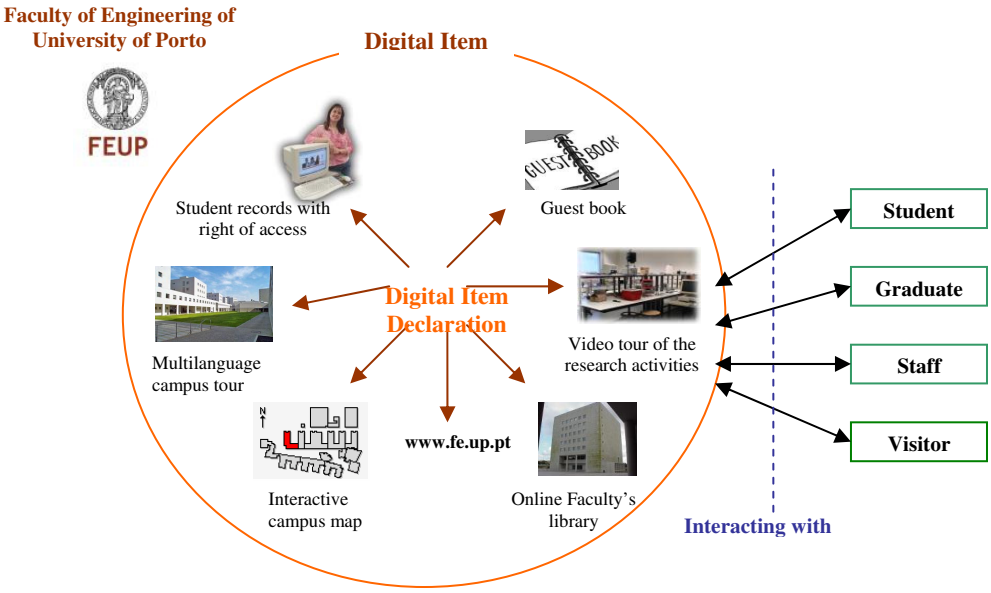



**Figure 1 Digital Item Example**

Another MPEG-21 essential concept is User, who interacts with Digital Items. MPEG-21 does not distinguish between types of users of Digital Items: they all are equally categorized as Users and can be consumers, creators, producers, individuals or organizations, communities, corporations, etc. The result is an open framework for multimedia delivery and consumption, with both the content creator and content consumer as focal points.

It can be considered that the main goal of MPEG-21 is to define the necessary technology to allow users to exchange, access, consume, trade or manipulate Digital Items in an efficient and transparent way over a variety of networks (dialup, broadband, mobile, wireless, broadcast, etc.) and devices (computers, PDAs, media players, mobile phones, digital TVs, etc.).

## 2.3. MPEG-21 Structure

As it was already mentioned, the objective for MPEG-21 is to provide a flexible solution when defining "a multimedia framework to enable transparent and augmented use of multimedia resources across a wide range of networks and devices" [8].

MPEG-21 is currently divided into 17 parts as follows:

- Part 1 - Vision, Technologies and Strategy
- Part 2 - Digital Item Declaration (DID)
- Part 3 - Digital Item Identification (DII)

- Part 4 - Intellectual Property Management and Protection Components (IPMP)
- Part 5 - Rights Expression Language (REL)
- Part 6 - Rights Data Dictionary (RDD)
- Part 7 - Digital Item Adaptation (DIA)
- Part 8 - Reference Software (RefSW)
- Part 9 - File Format (FF)
- Part 10 - Digital Item Processing (DIP)
- Part 11 - Evaluation Tools for Persistent Association Technologies (PAT)
- Part 12 - Test Bed for MPEG-21 Resource Delivery
- Part 14 - Conformance Testing
- Part 15 - Event Reporting (ER)
- Part 16 - Binary Format
- Part 17 - Fragment Identification for MPEG Media Types
- Part 18 – Digital Item Streaming.

All parts of MPEG-21 contribute to achieve the objectives, but more relevant for this thesis, due to the issues they address and to their development status, were the following parts: 2 (DID), 3 (DII) and part 10 (DIP). In the following, a summary description for each MPEG-21 part is presented, but a more detailed presentation will be made for parts 2 and 10 due to their importance for developing this dissertation.

### Part 1 – Vision, technologies, and strategy

Part 1 consists of a Technical Report that introduces the MPEG-21 multimedia framework and its architectural elements with the functional requirements for their specification. It defines the new "vision" for a common multimedia framework, the "strategy" for achieving it and the way by which "technologies" can be integrated to make up such a framework [2]. This part specifies the key concepts of MPEG-21, such as Digital Item and User.

### Part 2 – Digital Item Declaration

The Digital Item Declaration (DID) part of the MPEG-21 standard [9],[10] specifies the mechanism for declaring the structures of Digital Items which encapsulate or reference multimedia resources.

A DID is defined by a model, a representation and a schema.

The Digital Item Declaration Model provides a set of abstract terms and concepts for defining a scheme for Digital Item Declaration. The goal of this model is to be "*as flexible and general as possible*" [10].

The representation of a DID is the normative description of the syntax and semantics of each of the DID elements represented in XML.

The DID schema is the normative XML schema comprising the entire grammar of the DID representation in XML.

The DID model defines the following elements that can be included in a DID document:

- *DIDL* – is the root element of a DIDL instance document and may contain an optional *Declarations* element, followed by exactly one *Container* or *Item*.
- *Container* – is a structure that groups *Items* and/or *Containers* used to form logical packages (for transport or exchange) or logical shelves (for organization). The *Descriptor* elements are for the labeling of containers with information that is appropriate for the purpose of the grouping.
- *Item* – is the declarative representation of the Digital Item and is a grouping of sub-*Items* and/or *Containers*. The *Items* may contain *Choices* which allow them to be customized or configured. *Items* may also be conditional (on predicates asserted by *Selections* defined in the *Choices*). *Items* can be annotated. *Item* that does not contain sub-*Items* is considered as being a logically indivisible entity.
- *Component* – binds a *Resource* to a set of descriptors. These descriptors are information related to all or part of the specific resource instance. Such descriptors contain control or structural information about the resource and do not contain information describing the content within.
- *Descriptor* – associates information with the enclosing element. This information may be a *Component* or a textual *Statement*.
- *Statement* – is a literal textual value that contains information (e.g., descriptive, control, revision tracking or identifying information).
- *Condition* – describes the enclosing element as being optional and links it to the *Selection*(s) that affect its inclusion.
- *Choice* – describes a set of related *Selections* that can affect an *Item* configuration.
- *Selection* – describes a specific decision that affects one or more *Conditions* somewhere within an *Item*.
- *Annotation* – describes a set of information about another identified element of the model without altering or adding to that element.

- *Assertion* – defines a configured state of a *Choice* by asserting true, false, or undecided values to some number of *Predicates* associated with the *Selections* for that *Choice*.
- *Resource* – is an individually identifiable asset such as a video or audio clip, an image, or a textual asset. All resources must be available through unambiguous address locations.
- *Fragment* – designates a specific point or range within a resource. *Fragments* may be resource-type specific.
- *Anchor* – binds *Descriptors* to a *Fragment*, which correspond to a specific location or part of a *Resource*.
- *Predicate* – is a declaration that can be true, false or undecided.

Figure 2 illustrates an example of a DID structure containing some of the elements defined above and shows how these elements are related.

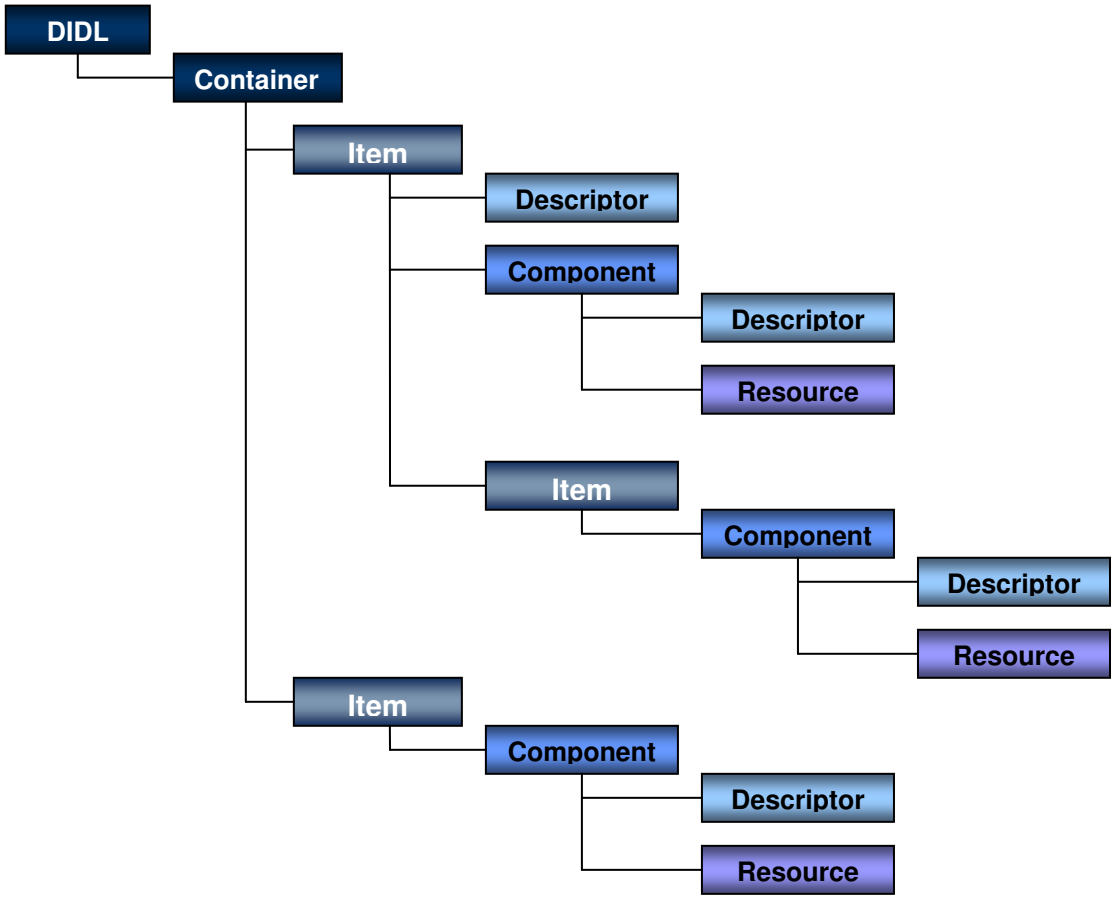



**Figure 2 Example of a DID structure**

The model is flexible and enables higher level functionality and interoperability by allowing the connection of the several parts of MPEG-21, the inclusion of other description schemes, etc. [9] [10].

The DID model provides the support for a static User - Digital Item interaction. This is given by the possibility of Users configuring which parts of the DID to consume as a function of the settings made for the *Selections* conditioning the respective parts.

MPEG-21 does not specify the way that DID content should be visualized. Therefore the purpose of the present dissertation is to propose and implement a model for the presentation of the DID structure in a user friendly manner.

The Digital Item Declaration became an International standard in 2003 [9]. Since then, MPEG-21 has been making some modifications to the specification in order to prepare the second edition of the DID which is currently at the stage of Final Draft International Standard [11].

### Part 3 – Digital Item Identification

Part 3 provides a normative way of identifying Digital Items and parts thereof, Intellectual Property related to Digital Item and Description Schemes or a modality of identifying the different types of Digital Items. Such identification is made by encapsulating Uniform Resource Identifier (URI) [12] into the specified element and including it in a specific location in the Digital Item Declaration (DID). The Digital Item Identification (DII) does not specify new identification systems when others with the same purpose already exist and is flexible enough to include them.

Summarizing, the DII specifies how identifiers can be associated with a DID or parts thereof. Identification schemes and the respective generation are out of the scope of this part [13].

### Part 4 – Intellectual Property Management and Protection Components

This part provides the means to reliably manage and protect contents across networks and devices by defining an interoperable framework for Intellectual Property Management and Protection Component (IPMP), the MPEG-21 equivalent of Digital Rights Management (DRM). The objective is to achieve the so desired interoperability, independently of the IPMP tools and features used [14]. In this part of MPEG-21 it is indicated how to include IPMP information and protected parts of Digital Items in a DIDL document, but it does not specify protection measures, keys, key management, trust management, encryption algorithms, certification infrastructure or other components necessary for a complete IPMP solution. The

aim of IPMP Components is to allow the control of the flow and usage of Digital Items [14], [15].

The relationship with the DID part of MPEG-21 is defined in the Digital Item Declaration Language part of IPMP Components, which provides a protected Representation of DID, allowing the DID hierarchy, which is encrypted, digitally signed or otherwise governed, to be included in a DID document in a schematically valid manner [15].

This part of MPEG-21 is a Final Draft International Standard (N7717, Nice, October 2005, France).

### Part 5 – Rights Expression Language

To assure that the rights of the content owners are respected it is necessary to take into account conditions regarding time limits, addressee, fees, etc., and is necessary to exchange information concerning the rights that must govern the content's use by the players involved in the distribution and consumption of the multimedia resources.

The Rights Expression Language (REL) specifies a machine-readable language intended to provide flexible and interoperable mechanisms: for declaring rights and permissions using terms defined in part 6 of the standard (the Rights Data Dictionary); for ensuring that personal data are processed according to individual rights; to allow Users to express their rights while guaranteeing the protected exchange of information; and to grant the right to play a Digital Item in an adapted form [16].

IPMP relates to REL as the former defines how rights expressions can be unambiguously associated with their targets and also defines how to specify the locations from which applicable licenses can be retrieved and the methods or processes for acquiring them [15].

This part of MPEG-21 became an International Standard in April 2004.

### Part 6 – Rights Data Dictionary

The Rights Data Dictionary (RDD) comprises a set of key terms required to describe rights and permissions granted through the use of a standard syntactic convention and to enable the applications across all domains in which rights and permissions need to be expressed. RDD also specifies how further terms may be defined under the governance of a registration authority [17].

This part of MPEG-21 provides the definitions of terms for use in REL and RDD, and also supports the mapping and transformation of metadata from the terminology of one

namespace (or authority) into another namespace (or authority) in an automated or partially-automated way, with the minimum ambiguity or loss of semantic integrity [17].

Presently, the only specific relationship with other parts of MPEG-21 is with Part 5, the Rights Expression Language.

RDD became an International Standard in 2003.

### Part 7 – Digital Item Adaptation

Digital Item Adaptation specifies a set of description tools to allow Digital Items to be adapted in order to achieve interoperable and transparent access to multimedia contents, hiding the specifics of terminal devices, networks and contents formats. These tools address aspects of resource and descriptor (metadata) adaptation as well as aspects related with Quality of Service (QoS) management.

Digital Item Adaptation processes a Digital Item by making use of a resource adaptation engine and/or a descriptor adaptation engine, which together produce a modified Digital Item. When subject to the adaptation engine, three types of adaptation processes can be applied to Digital Item: adaptation of descriptions (e.g., reduction of metadata information so that it may be presented in a specific end-user device); adaptation of resources (e.g., video-to-audio, text-to-speech); adaptation of the DID as a whole [18].

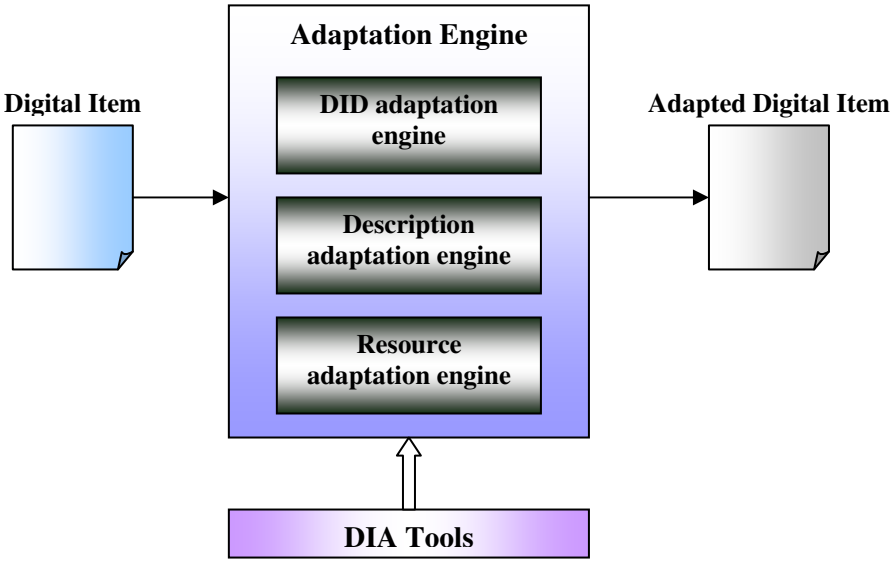



**Figure 3 Conceptual architecture of Digital Item Adaptation**

The tools provided for Digital Item Adaptation are description tools intended to convey information and assist the adaptation of Digital Items and not to perform the adaptation itself. Consequently, Digital Item Adaptation specifies tools to convey information for adaptation, but not the components responsible for the very adaptation.

The Digital Item Adaptation tools provide the means for describing the usage environment, for assisting the decision on tradeoffs between constraints and feasible adaptation operations, for facilitating the adaptation of metadata and also for preserving the state of consumption of a Digital Item. In Annex C, the DIA description tools are briefly defined.

This part became a Final Draft International Standard in December 2003.

### Part 8 – Reference Software

Reference Software is composed by a set of software modules developed for validating the specifications of all parts of MPEG-21, providing tools for conformance testing, serving as a base for building applications with validation purposes [19], [20].

The main goals of the Reference Software are:

- Validation of the written specification of the several parts of MPEG-21;
- Clarification of the specification for the several parts of MPEG-21.

The information supplied by this part of MPEG-21 is applicable for:

- Determining the Reference Software modules available for parts of MPEG-21;
- Understanding the functionality of the available Reference Software modules;
- Utilizing the available Reference Software modules.

Reference Software modules were developed for parts 2, 3, 5, 6 and 7.

Also, there were many contributions to develop the reference software for other parts of MPEG-21 but currently they have not been published. The reference software for Digital Item Processing that was used for this dissertation is presently available on the Enikos website [21].

Part 8 is at the stage of Final Draft International Standard since April 2005.

### Part 9 – File format

This part of MPEG-21 defines a file format for storing and distributing Digital Items. This file format inherits several concepts from the one defined in MPEG-4.

The File Format specifies how an MPEG-21 Digital Item can be a complex collection (single “package content”) of media resources (e.g., images and videos), metadata, textual and/or binary data and referenced information, etc. This enables the interchange, editing and play of MPEG-21 Digital Items.

The MPEG-21 File Format became an International Standard in January 2005.

The File Format represents the next generation file format that encapsulates everything. For the present it needs some maturation of all parts of the MPEG-21 standard in order to achieve its main purpose [22].

### Part 10 – Digital Item Processing

This part of MPEG-21 specifies the syntax and semantics of tools that may be used to process Digital Items. On receiving the static declaration, as it is the case of the DID, Users have nothing that indicates how the information should be processed. Digital Item Processing (DIP) intends to cover all aspects of processing a static Digital Item from the user perspective, by enabling the MPEG-21 Users to specify a selection of preferred procedures to handle the Digital Item when configuring, validating, processing and consuming it.

Digital Item Processing includes such processing as Digital Item downloading, IPMP handling and rights management, media resources downloading, printing or playing, presentation of the Digital Item, etc. [23][3]. It is also possible to extend the DIDL to allow the addition of user specified functionality to a DID, but maintaining the interoperability at the processing level.

The Digital Item Methods (DIM) are the DIP mechanism for enabling users to include sequences of instructions for adding functionality to a Digital Item. The method definition may be referenced from or embedded in the DID. Digital Item Methods can be regarded as a "menu" of User interaction possibilities.

DIMs are intended for working with parts of a Digital Item at the DID level only; they are not intended for lower level resource processing. In Figure 4 some operations available to DIMs are presented, related to the other parts of MPEG-21.

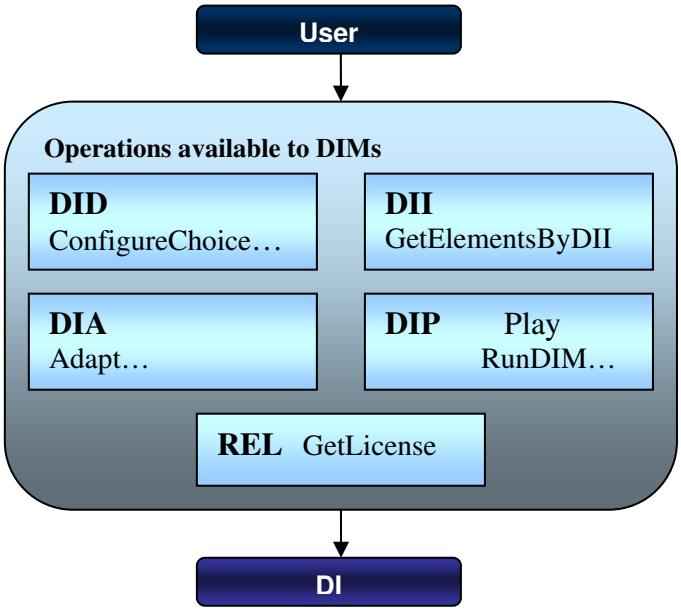



**Figure 4 Operations for DIMs involving other MPEG-21 parts**

DIMs are programmatic scripts (expressed using the Digital Item Method Language) included in the Digital Items, with access to DIDL elements and that contain a set of defined base operations (Digital Item Basic Operations) and also with mechanism for extension operations (Digital Item eXtended Operations). The Digital Item Method Language (DIML) is based on the ECMA Script [24] to which were added the DIP extensions (DIML object types and their functionality). DIML specifies the normative language for defining interoperable DIMs and from which the basic operations may be called.

The Digital Item Base Operations (DIBOs) are basic operations with a normative interface and normative semantics (but their implementation is left to the implementor decision) that access and manipulate the DID at the DIDL level or they are intended to load and serialize DID instance documents. The DIBOs specify a high level normative interface to the basic functionality provided to a User.

When referring to DIMs, there are several roles that a User might undertake. For example, as a consumer a User can interact with a Digital Item via the execution of a DIM. As a creator, a User can author a DIM to be included in a Digital Item. As MPEG-21 compatible hardware or software for processing a Digital Item, a User can provide an execution environment for DIMs, where DIBO implementations should be appropriatly included in the environment.

Digital Item eXtension Operations (DIXOs) specify a normative mechanism for enabling extended functionality in an efficient way. DIXOs are operations defined in a specified DIXO Language (it is chosen by the implementor) used to extend the functionality provided by DIBOs and that are called by a specified DIBO (runMyDIXO – e.g., runJDIXO in the case of Java implementation). DIXOs also include bindings to DIBOs in DIXO Language.

DIMs may choose to use DIBOs when they are available and DIXOs when no DIBO is available [23] [3].

Part 10 is at the stage of Final Draft International Standard since July 2005.

### Part 11 – Evaluation Methods for Persistent Association Technologies

In MPEG-21 there is a need for tools that can create and maintain (e.g., detect or extract) an association between contents, metadata and IPMP elements [25].

The associations between multimedia elements and the related information are made by tools based on techniques like “fingerprinting” and “watermarking”, where the association can be directly embedded within or inferred from the content itself. Also, these tools – named

Persistent Association Technologies (PAT) – allow such inferences to persist in the face of adaptation of the content [26].

The characteristics identified as being most important to be treated by PAT are: fingerprint size, watermark payload, granularity, perceptibility, robustness, reliability and computational performance. Presently, part 11 of MPEG-21 is focused on the evaluation of fingerprints and watermarks, when applied to audio or video content.

On March of 2004, part 11 was published as a Draft Technical Report.

### Part 12 – Test bed for MPEG-21 resource delivery

Part 12 provides a software-based test bed for delivering scalable media and for testing and evaluating this scalable media delivery in streaming environments [27][13].

The test bed is considered to have an MPEG streaming player, a media server and an IP network emulator. The aim for part 12 is to provide a flexible and fair test environment for evaluating delivery technologies for MPEG contents over IP networks [28].

The test bed for MPEG-21 resource delivery describes how various MPEG technologies can be integrated into a working system for testing and verification purposes.

This part of MPEG-21 is at the phase of Technical Report since March 2004.

### Part 14 – Conformance Testing

This part of MPEG-21 defines the criteria that must be satisfied and strategies that must be followed to verify the conformance of implementations of other parts of the standards (DID, DII, REL, RDD, DIA) with the respective specification [29].

Conformance testing is a methodology for checking if the set of requirements is met.

MPEG-21 Conformance is at the stage of Committee Draft, waiting to be approved as final version since July 2006.

### Part 15 – Event Reporting

The 15th part of MPEG-21 defines the metrics and interfaces that enable Users to understand precisely the performance of all reportable events within the framework, where an event is defined as the context in which, or as a result of which, something changes. Event Reporting contains the metrics and interfaces that refer to identified Digital Items, environments, processes, transactions and Users. It is used to report the occurrence of Events which may be directly or indirectly related to Digital Items and is limited to a inter-Peer reporting. Peers represent devices or applications that compliantly process a Digital Item [30] [31].

The Event Reporting specification defines two main entities: Event Report Request (ER-R) and Event Report (ER). The former (ER-R) contains information about which Events to report, what information is to be reported and to whom. The latter (ER) is created by an MPEG-21 Peer in response to an Event Report Request when the conditions specified by the ER-R are met. Examples where Event Reports may be requested are related with usage reports, copyright reports (e.g., copies monitoring, performance monitoring), technical reports (e.g., bandwidth availability, network congestion), financial reports (e.g., proof of purchase, license purchase and delivery), etc. [32].

This part of MPEG-21 is at the stage of draft for International Standard (Final Committee Draft) since July 2005.

### Part 16 – Binary Format

Part 16 specifies a binary format for XML-based descriptions for the compression and streaming of Digital Items or parts thereof. It provides an alternative serialization format for MPEG-21 descriptions as standardized in other parts of MPEG-21 and enables the efficient interchange or storage of these descriptions. Such format is based on tools specified in MPEG-7 Systems, to which some additions were made [33].

This part defines means to binary decode and reconstruct fragmented descriptions of MPEG-21, but does not specify how and when to consume these descriptions. The decoding process shall ensure that the representation of the reconstructed description is identical bit-by-bit with the representation of the original description [33].

This part of MPEG-21 is at a Final Draft International Standard stage, since April 2005.

### Part 17 – Fragment Identification for MPEG Resources

Fragment Identification for MPEG Resources specifies a normative syntax for URI Fragment Identifiers to be used for addressing MPEG resource having as Internet Media Type one of the following:

audio/mpeg [RFC3003]
video/mpeg [RFC2045, RFC2046]
video/mp4 [RFC4337]
audio/mp4 [RFC4337]
application/mp4 [RFC4337].

MPEG URI Fragment Identifier schemes offer comprehensive and flexible mechanisms for addressing fragments of audiovisual content. Therefore, their use may potentially be extended to other audiovisual MIME types [34].

Such URI Fragment Identifiers are to be used after the '#' character in a URI reference.

The syntax for URI Fragment Identifiers is based on the W3C XPointer Framework Recommendation and adds the ability to address [34]:

- Temporal, spatial and spatio-temporal locations;
- Logical unit(s) of a resource according to a given Logical Model;
- A byte range of a resource;
- Items or Tracks of an ISO Base Media File.

This part of MPEG-21 became a Final Draft International Standard in January 2006.

### Part 18 – Digital Item Streaming

Initially, this part was named Schema Files and contained the electronic versions of the XML Schema files for all standards within MPEG-21. It supplied users of MPEG-21 with a single place to obtain all XML Schema files that had already been standardized through all the other Parts of MPEG-21 [35].

Presently, part 18 of MPEG-21 is named Digital Item Streaming and specifies the tools for streaming Digital Items. The first tool is the Bitstream Binding Language, which describes how Digital Items can be mapped to the delivery channels. Consequently, Digital Item Declarations declared according to the DIDL specified in part 2 of MPEG-21 can be streamed using this tool. IPMP Digital Item Declarations stated according to part 4 can also be streamed. These Digital Item Declarations can contain or refer to XML conforming to other parts of MPEG-21 (DIA descriptions - part 7 and REL - part 5). Such XML can be streamed using the Bitstream Binding Language, if it is embedded in a DID, referenced by a DID or used separately from a DID [36].

This part has been created during the 71st meeting of ISO/IEC (Hong Kong, China), in January 2005. For the moment, the document for the Final Committee Draft (available to ISO/IEC since April 2006) is being studied.

# Chapter 3

## 3. MPEG-21 based applications

MPEG-21 is considered a powerful and flexible multimedia framework for the transparent delivery and consumption of various multimedia resources. Applications have been built to demonstrate and to put into practice the advantages that MPEG-21 framework offers.

The applicability of MPEG-21 is for many fields: multimedia repositories, broadcast usage, publishing, music and video releases, asset management, content filtering, media resource delivery, media streaming, digital libraries, in e-learning and other areas related with multimedia.

This section presents projects and implemented systems focused on applying the concepts and technology of MPEG-21. Some of them have contributed to the development of the standard's reference software.

### 3.1. MUFFINS

*Multimedia Framework For Interoperability in Secure (MPEG-21) environments* (MUFFINS) [37] is an European Community (EC) funded project (ended in December 2003) concerned with the problem of transparent improved delivery and use of rich-media resources across a wide range of networks and devices, by different categories of users in multiple application domains.

One of the objectives of the MUFFINS project is to contribute to the development of the MPEG-21 standard and to investigate the problem of description, delivery and protection of rich-media content in order to propose a complete solution framework for scenarios of usage of that content, such as search, delivery and related rights management handling. The application developed in this project was not intended to be MPEG-21 reference software, but a demonstration platform for the concepts of the standard. It is based on various parts of MPEG-21 and many of its use cases. The framework includes: a search application which

matches DII [13], DIA [18] and REL [16] descriptors of the content to those supplied by a user; a media streamer that can adapt the content to the capabilities of the terminal using the DIA descriptors; a terminal application that can pass a Digital Item to another terminal with different capabilities. The DIA descriptors attached to the Digital Item ensure that all terminals can play it; the REL descriptors are meant to be used by all the components to describe and enforce usage rights. A license management system is important for the framework as it communicates with the other components in order to purchase the User rights on a resource, to trade them or to get a license for a Digital Item.

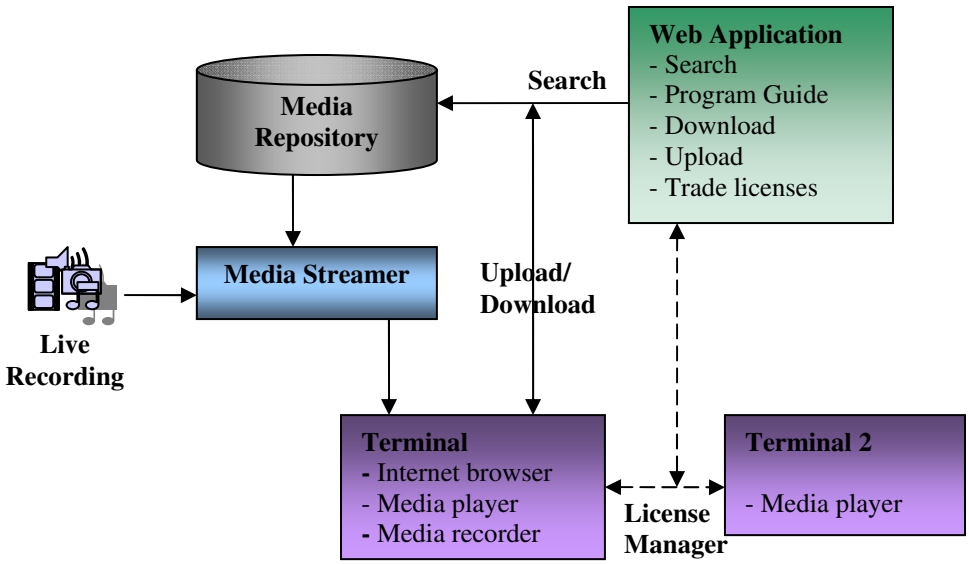



**Figure 5 Components of MUFFINS multimedia framework**

The terminal application developed in MUFFINS is addressed to Users for consuming the multimedia content. This terminal application does not perform a visualization of the content structured in a Digital Item Declaration form. It includes: a Web browser application for searching, uploading and downloading the content, it may also be used to buy or sell licenses; a media player which is stored locally, downloaded or streamed for immediate playback; and IPMP Components with features specified in the 4th part of the MPEG-21. The terminal can also have included a media recorder for the streamed content.

The processing of selection, acquisition and consumption of a Digital Item is briefly presented in Figure 6:

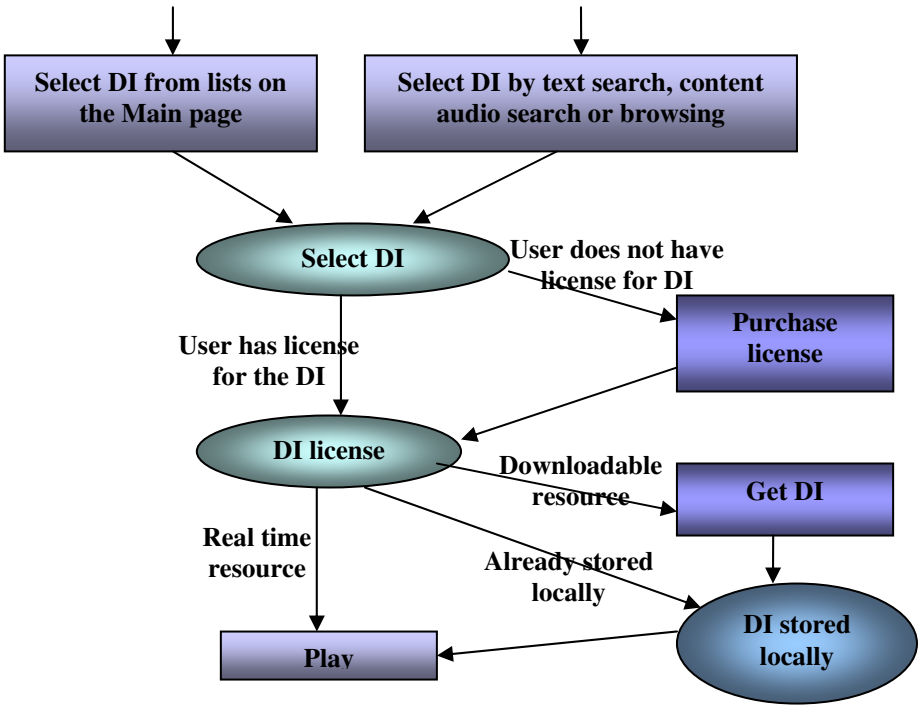



**Figure 6 A walkthrough in the Muffins Online Store (MOS)**

The MUFFINS consortium has built a first implementation of some parts of the MPEG-21 standard around a web store of media items, focusing on the following elements: the Digital Item Declaration and Digital Item Identification descriptors for representing each media item, the Digital Item Adaptation for matching the content to the terminal capabilities (display resolution, resource bitrate, number of audio channels, etc.) and the Rights Expression Language to describe the license for the content [38].

## 3.2. DANAE

DANAE [39] [40], an IST European Community co-funded project, was interested in the dynamic and distributed adaptation of scalable multimedia content in a context-aware environment. The objective of DANAE was to specify and develop an advanced MPEG-21 infrastructure for such an environment, to permit flexible and dynamic media content adaptation, delivery and consumption. The goal was to enable end-to-end quality of multimedia services at a minimal cost for the end-user, with the integration of features and tools available from MPEG-4 and MPEG-7, and DRM support in the adaptation chain. Their system focused on DIA and DIP and contributed to the MPEG-21 standardization effort. DANAE supports the application of DIP for DIA to adapt Digital Items in a globally, optimized and semantically aware way to allow a dynamically changing usage context as well

as to enable the adaptation of live content (the content may be delivered while still being created).

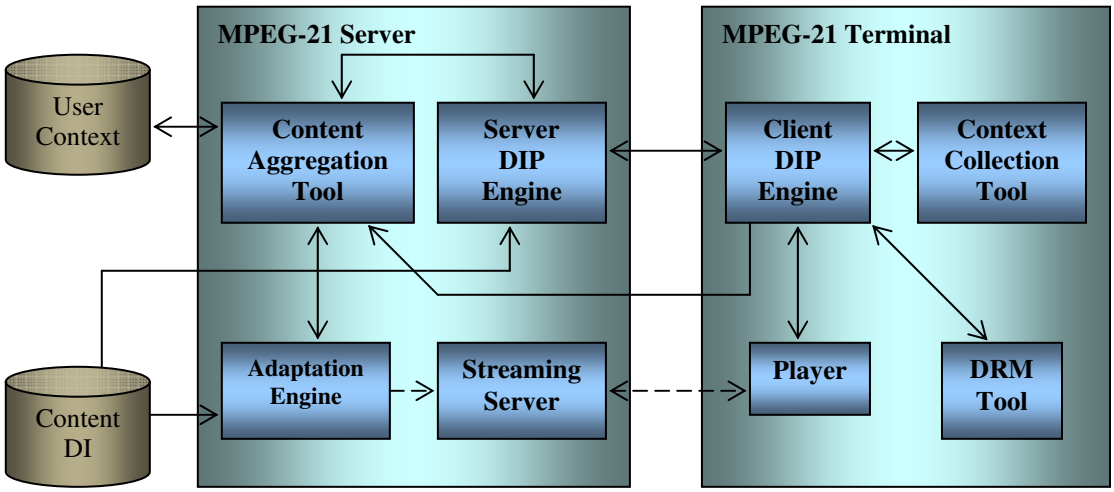



**Figure 7 DANAE architecture extracted from [39]**

The MPEG-21 terminal application proposed by DANAE enables Users to consume and create the Digital Items. It also provides DIP support and contains DRM tools implementing the IPMP functionality. By the time of writing of this thesis, there was no detailed information available about their MPEG-21 terminal.

One of the project partners, MUSEON, has developed in parallel an application based on the new technical possibilities and services conceptions opened by DANAE. An example of an usage scenario for this application is a visit to a museum (www.museon.nl). The visitors should be able to receive multimedia content (images, video, audio, text, virtual character, interactive, etc.) on their mobile devices (PDA, Tablet PC, etc.) in a location aware (for example specific to the current zone), personalized (depending an the visitor's age, thematic interest, expertise level) and adapted to the current usage environment (e.g., to network conditions).

The DANAE work is based on the following aspects:

- The definition of scalable media formats with the associated metadata;
- Their adaptation to the session context, i.e. making full featured distributed multimedia scene adaptation through global optimization of audio, video, 2D graphics, 2D/3D virtual characters;
- The transport and delivery of multimedia content to the end-user.

DANAE has contributed to the development of the basic DIP concepts by defining an API for calling DIBOs from DIMs. A proof of concept is performed by a software implementation of the most relevant tools of DIP in the DIA context [39].

### 3.3. ENTHRONE

ENTHRONE, *End-to-End QoS through Integrated Management of Content, Networks and Terminals* (IST 507637) [1], is an EC-funded project. The first phase of activity finished in March 2006 and the second phase started in September 2006. In the first phase an integrated solution to manage the functionality of various entities in the content distribution chain was developed. The aim of the project is to provide transparent access to multimedia resources while ensuring Quality of Service (QoS) and efficient usage of resources. The ENTHRONE system is based on the MPEG-21 concepts and it uses distributed technologies supported by middleware layers, in order to build an interoperable services-oriented architecture.

ENTHRONE has developed an integrated component – the *Integrated Management Supervisor* (IMS), based on distributed technologies and which is responsible for the decisions that are taken and actions approved in the delivery chain. The IMS has the main role in ensuring the optimum management of the whole content distribution chain. It supports the cooperation among various entities in the digital information distribution chain, from content generation and service creation to user terminals, using heterogeneous networks and based on the end-to-end QoS approach.

Three main entities were defined for ENTHRONE:

- The consumer (end-user) – uses a terminal that includes the DID Browser, a Media Player and other subsystems;
- The service provider (SP) – operates the web server that runs a portal, the Service Provider front-end (SP-FE) and the server side IMS subsystems (Dispatcher and Content Manager);
- And the content provider (CP) – a streaming server containing a TV and Multimedia (TVM) Processor tool.

From the terminal perspective the Digital Item retrieving is made through the following 2 steps:

1. DID browsing – during which the end-user browses a list of DIDs and the DID content from that list;
2. Digital Item selection – when the user chooses a certain Digital Item to be played.

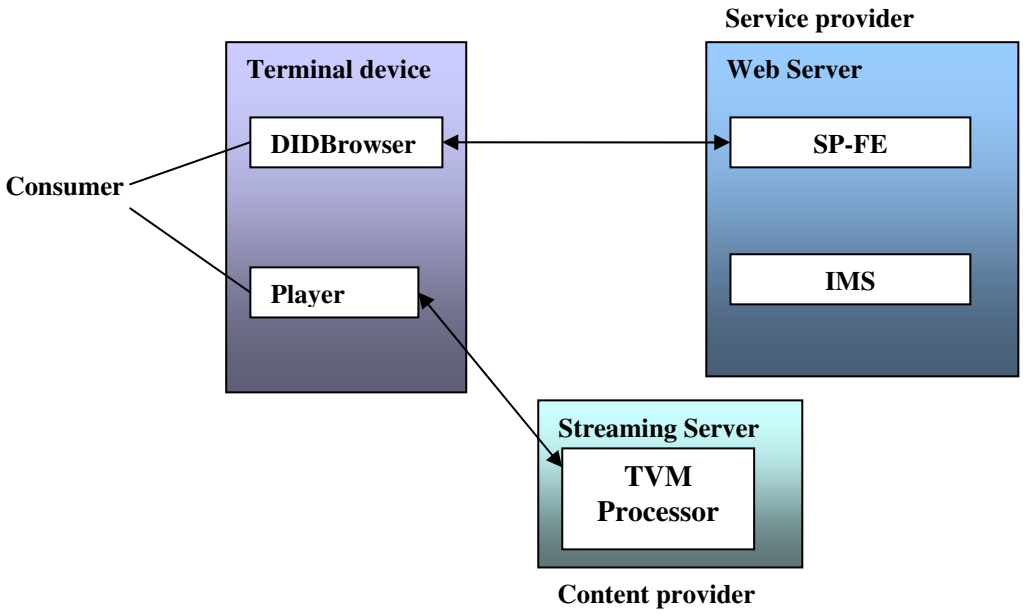



**Figure 8 Retrieving a DID with DIDBrowser, extracted from [41]**

The essential functionalities of the ENTHRONE modules are presented next [41]:

- DID Browser – is a terminal application used for searching and visualizing the DIDs that are received as XML files, and also for playing the contained resources.
- Media Player – receives the URL of the resource to be retrieved and played.
- IMS (Integrated Management Supervisor) – contains 4 subsystems: IMS Dispatcher (IMS Disp), IMS Network Manager, IMS Content Manager (IMS-CM) and IMS Terminal Device Manager (IMS-TDM).
    - IMS-Disp – used to search for content in other Service Providers (SP) and for determining the QoS of the service to be provided to the end-user and for monitoring the QoS of the service being provided;
    - IMS-CM – contains the metadata used for searching purposes and the metadata used for adaptation purposes;
    - IMS-TDM – sends context information describing the terminal capabilities and the characteristics of the network in which the terminal resides.
- Service Provider Front End – sends the requested DID, receives HTTP-GET requests for the DIDs, sends the location of the chosen variation (version with specific characteristics such as resolution, bitrate, etc.) of the resource to be played and receives the chosen identifier and the associated context information.
- ENTHRONE Terminal Middleware (EM3W) – is responsible for the communication among the several subsystems that form the ENTHRONE terminal.

The first prototype of ENTHRONE has already been built providing basic functionalities, but it serves as a proof-of-concept. This first prototype has support for browsing and selection of unprotected content. It also supports a basic form of content adaptation through which the best variation of the content, matching the terminal capabilities and user preferences, is transparently selected [41].

The project intends to improve the system in the second phase, introducing more features such as advanced forms of content adaptation, dynamic QoS monitoring, fault recovery, management and applying digital rights.

## 3.4. Ghent's MPEG-21 applications

Presently there is a considerable number of research activities, initiated by various institutes, universities and multimedia companies, dedicated to the development of the MPEG-21 technology. Several of them have already contributed for the specification of different parts of the MPEG-21 standard. One of these contributors is the research group Multimedia Lab [42] founded in 2001 by Ghent University (Belgium) and that succeeded in putting the first MPEG-21 application online. They also developed other MPEG-21 useful applications as it is mentioned below.

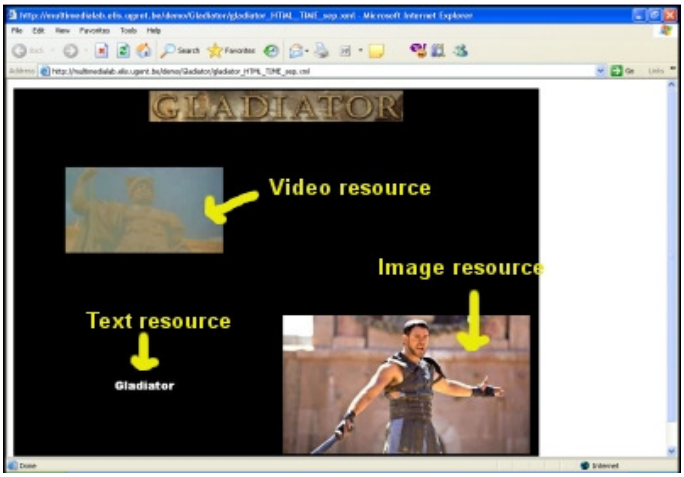
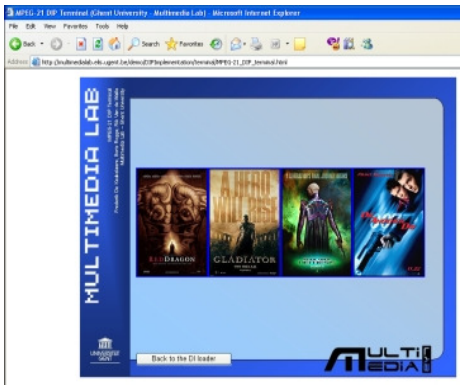



**Figure 9 Temporal synchronization of media within MPEG-21 Digital Item Declarations**

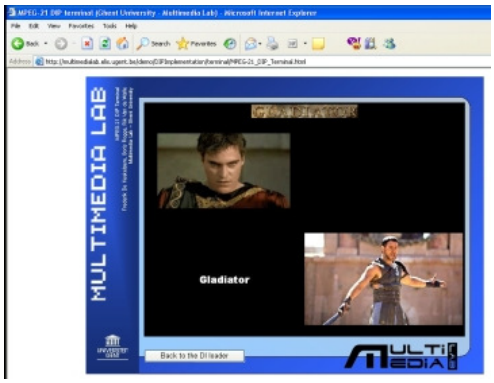
Ghent's first MPEG-21 Web application was a demo that illustrates how MPEG-21 technology can be used for the synchronization of multimedia data, where the temporal synchronization of media is made within MPEG-21 Digital Item Declarations (see Figure 9). This demo can be used to produce synchronized multimedia presentations containing various resources (text, images, video and audio) embedded or referenced by an MPEG-21 Digital Item. The presentation is made with a HTML layout. The demo can be viewed in Web

browsers supporting SMIL (Synchronized Multimedia Integration Language) and XSLT (Extensible Stylesheet Language Transformations).

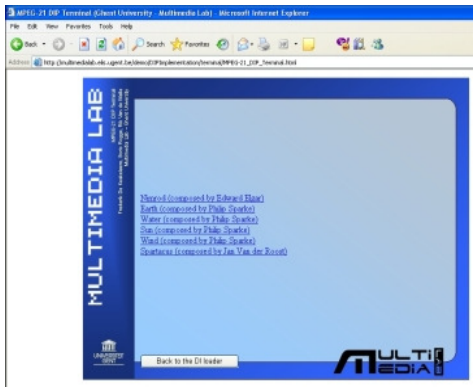
Multimedia Lab has also created a generic MPEG-21 terminal for the processing Digital Items and Digital Item Methods implementing the functionalities of part 10 of the standard. This is an online terminal which uses three MPEG-21 Digital Items for the demonstration and communicates with a streaming server, through SOAP messages, for dynamically configuring the video stream referenced by the Digital Items.




a. Movie trailers




b. Gladiator Digital Item




c. Music album tracks

**Figure 10 Demo MPEG-21 terminal for Digital Item Processing**

When accessing the terminal, the User is presented with a list containing three Digital Items. One Digital Item contains a list with four trailer movies and for each trailer are available various Digital Item Methods (DIMs) for playing the respective resource (e.g., watch the trailer, switch to full audio and full video, switch to full audio and reduced framerate video or switch to audio only). The second Digital Item has a DIM method for playing a demo presentation (this presentation is also a Digital Item, as it was described in the previous demo application). And the last Digital Item from the list contains a musical album. When loading this Digital Item the User has to select the DIM for playing a track, then a list with the available tracks will be displayed and from which the User can choose one.

Ghent University has contributed to the development of the reference software for Digital Item Processing (DIP) – part 10 of MPEG-21. This demo terminal application exemplifies a way of using the DIP reference software when creating an MPEG-21 terminal for processing the Digital Items (it may be considered a player for Digital Items). The terminal for the Web was implemented with ActiveX controls and its functionalities are accessible using the Internet Explorer browser.

Other applications developed by Multimedia Lab and related with MPEG-21 are: a MPEG-21 scalable Video-on-Demand which uses MPEG-21 Digital Items and Time-Dependent Metadata to deliver MPEG-4 videos (it works also on mobile devices); a demo application that demonstrates how MPEG-21 Session Mobility can be realized between different devices, allowing transparent transfer of multimedia sessions; and an MPEG-21 wireless application for creating a device specific interface for WAP applications (those interfaces are created using one MPEG-21 document and different WML-Stylesheets) [43], [44].

With its implemented demo applications, Multimedia Lab has demonstrated how Digital Item Declaration and Digital Item Processing functionalities can be included in different multimedia applications in the context of MPEG-21.

### 3.5. Other MPEG-21 applications contributions

Klugenfurt University (Austria) also brought a considerable contribution to the MPEG-21 DIA standardization [45]. Some examples of their implemented MPEG-21 applications are: the tools for multimedia adaptation (e.g., gBSDtoBin, BSDLink Webeditor) based on the DIA reference software; the ViTooKi operating system that has MPEG-21 support for describing terminal capabilities and user preferences [45]; the DIBuilder application for creating Digital Items; and the DIConsumer application for the consumption of Digital Items. Klugenfurt University and Ghent University have also collaborated to elaborate the specifications for some parts of the MPEG-21 standard, mainly the DIA.

Wallongong University (Australia) is an example of another institution that manifests interest in implementing MPEG-21 applications and investigating this area in order to contribute to the development of MPEG-21. More about their research can be found in [46]. Among all their projects, one that caught our attention is about an MPEG-21 Peer for mobile devices [47]. The objective was to develop an MPEG-21 Peer (see the definition of Peer in Chapter 4) for mobile phones for demonstrating an efficient and flexible architecture which encompasses a sufficient range of MPEG-21 technologies to realize the consumption,

authoring and transmission of Digital Items. This application accesses and downloads Digital Items via Web, ensures hierarchical browsing, authoring and storage, provides the presentation of contained assets, and processes the protected content declared by IPMP DIDL. The navigation in the Digital Item hierarchy is made at the *Item* level. The Peer scans each sub-*Item* for a title located in a *Descriptor* and also displays the first image it finds associated with each sub-*Item* in a *Descriptor* or *Component*. The application has a modular and decoupled architecture which increases the portability and flexibility (referring to extensibility). This modular architecture determines a well defined separation between presentation and user interaction, and the processing of Digital Items functionality. A static representation of a Digital Item is made by the same module used for the MPEG-21 processing [47]. A similar architectural modularity concept will be used in the development of the application proposed in this thesis.

A group of research institutes, IBBT (Interdisciplinary institute for BroadBand Technology – Ghent University, Belgium) and IMEC (Interuniversity MicroElectronics Center, Belgium), collaborating with the leading Belgian public broadcast company VRT (Vlaamse Radio en Televisie), have developed a multimedia publication system based on the XiMPF document model that aims to provide the content author with a maximal control of the output publication enabling the maximization of reuse possibilities and allowing support for multiple platforms [48].

The XiMPF document model is an XML file that follows the MPEG-21 DID model. Each item in the XiMPF file, actually an XML file that can be transformed in the DID using XSLT transformations, contains a number of presentations, descriptions (including SMIL synchronization information and alternatives for different output channels of various types of client devices), template instances and sub-*Items* with different text versions, images, audio and video. This XiMPF model is flexible, permitting reusability, it is device independent and makes a well defined separation between content, layout, behavior and synchronization. Also it allows the use of MPEG-21 tools oriented for multimedia adaptation (DIA).

The developed publication system was applied to the news site of the leading Belgium broadcaster VRT and is generating adapted content for the following client platforms: PC, TV with set-top box, PDA and SMIL based players. More popular are the versions for PC and for PDA and these are shown in Figure 11. The PC version can display the full text and multiple multimedia items, while the PDA version displays only images and captions because the text is available on a separate page.

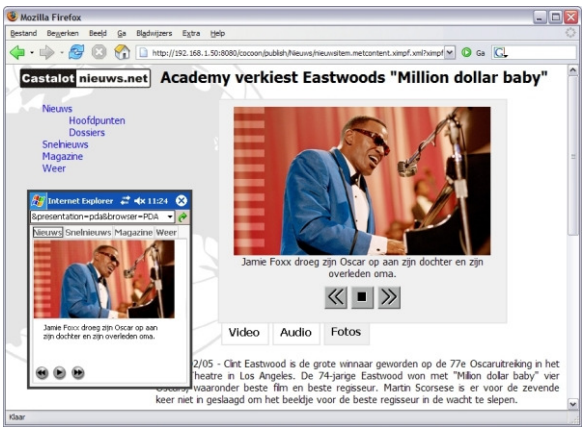



**Figure 11 Demonstration of the publication of a news item on a PC (main image) and PDA (smaller inset on lower left)[48]**

## 3.6. Remarks

In the literature, other references to MPEG-21 implementations can be found but these are not relevant for the present dissertation. The purpose of the system proposed and developed for this thesis was to provide a mechanism for presenting MPEG-21 information to end-users and to allow them consuming the multimedia content. The applications and EC projects that were presented in this chapter aim to put into practice the specifications for parts of MPEG-21 and also to present MPEG-21 information to Users (end-users). All the applications mentioned here are only a few examples of contributions that were made to demonstrate the applicability of MPEG-21. This standard is still under development and a great amount of work is required to meet the expectations.

# Chapter 4

## 4. MPEG-21 Terminals for Digital Items consumption

This chapter provides a description in terms of delivered functionality and usability of various software applications representing MPEG-21 Terminals (Peers), developed by research groups worldwide to present multimedia information, wrapped as MPEG-21 Digital Items, to end-users.

An MPEG-21 Terminal allows different Users to connect to multimedia applications within the MPEG-21 framework. The Terminal is a tool where multimedia content can be shared, always with the agreed/contracted quality, reliability and flexibility [2].

Peer is a term introduced in MPEG-21 for defining a "*device or application that compliantly processes a Digital Item*". In the more recent specifications of MPEG-21, the term Terminal is deliberately avoided because of its connotation as being the end point in a chain of communication. Peer explicitly also includes devices or applications that create or alter Digital Items [49].

### 4.1. Enikos's products

Enikos [21] is a Australian company focused exclusively on MPEG-21 technology and solutions. It develops and supplies software tools for multimedia applications based on MPEG-21. Enikos has been launched as the first company worldwide to build solutions based on the MPEG-21 Multimedia Framework. Their products consist in a set of tools that allow users to browse, create and customize Digital Items compliant with the MPEG-21 Part 2 standard. The two important projects of this company are DICreator and DIBrowser. By the time this thesis was being written, Enikos released a new product: the MPEG-21 DIP Desktop Peer which is a demo application implementing the DIP functionality.

#### DICreator

DICreator, launched in November 2003, is a tool that permits to easily create, edit and view Digital Item Declarations and provides also a fully validating parser. It hides the

complexity of the Digital Item XML schema behind the graphical user interface in such a manner that it minimizes the author's work and maximizes the output.

The DICreator is a client Java application that presents to the User a simple and hierarchical view of Digital Items, a summary *Description* of a selected *Item,* including text and other media descriptors, an editor window, for adding valid elements, and a feature to display and render media *Resources* in content view. In the Editor window the User is editing the DID and working with the DIDL element, and hence the Navigation view in the Editor window shows the structure of the DIDL elements in the DID. In the Browser window the User browses the Digital Items and the *Items* contained inside, and hence the Navigation view in the Browser window shows only the *Items* described by the DIDL elements of the DID.

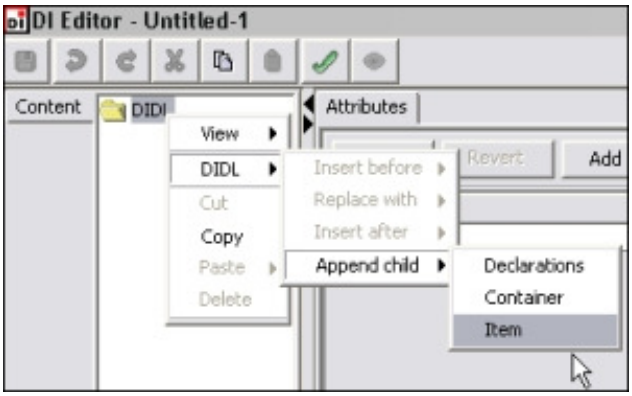



**Figure 12 Inserting an Item in a DID by using DI Editor view of DICreator application [21]**

The contents of the *Choice, Component, Descriptor, Item* and *Selection* elements (defined in the 2.3.2 sub-section) of the MPEG-21 Part 2 (DID) are displayed in dialog forms and the user can make his selections by completing radio button forms.

DICreator is an essential tool for those who are developing, authoring and experimenting with MPEG-21.

### DIBrowser

The Enikos DIBrowser, released on January 2004, is a Java desktop application capable of browsing any local or remote DID, presenting its internal structure as a tree. It performs an initial validation of the DID and if any errors or inconsistencies are detected they are graphically presented to the user. The DID is displayed using a tree-view which maps its XML structure. The browsing process is focused on the above mentioned tree. *Descriptions* are presented upon the user click on the correspondent tree node, independently of its nature. Only associated local files are retrieved. If another DID is included as a resource this application opens the DID as a text resource preventing its browsing [21]. The user may

statically interact with the DID and specify his preferences by configuring the *Choice* elements.

A screenshot of the application interface while browsing a DID document is presented in Figure 13.

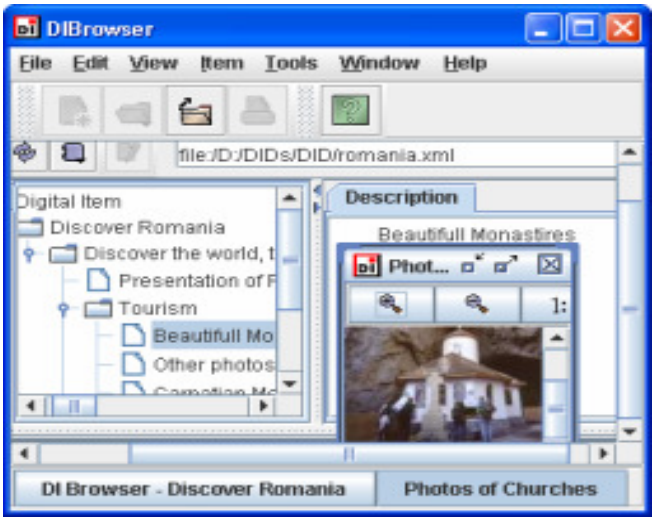



**Figure 13 Rendering a DID using Enikos DIBrowser**

DIBrowser and DICreator have internal multimedia capabilities and the ability to launch external viewers for extra file types and there is no need to abandon the user's favorite media players to use Digital Items. The common part of the two applications is that the Browser window of the DICreator is based on the DIBrowser. The difference between them is: the DICreator ensures creating and editing (during the browsing) of Digital Items, while DIBrowser is strictly dedicated for navigating through Digital Items.

### DIP Desktop Peer

In 2006 Enikos has made available on its website [21] a demo exercising the DIP functionality which is a desktop Java application based on the reference software provided for the MPEG-21 DIP. This reference software is also available on Enikos's website but only for MPEG members.

This demo opens local or remote DID files containing DIP information, then presents the User with a list of DIM methods that can be run. The User may choose a DIM for execution. DIP Desktop Peer has a simple and intuitive interface, allowing the Users to set their preferences: they can choose whether the auto run DIMs are automatically executed or not, they can view the XML source of the DID or they can setup the Web browser (Internet Explorer or Firefox) to be a browser for DID or DIM content.

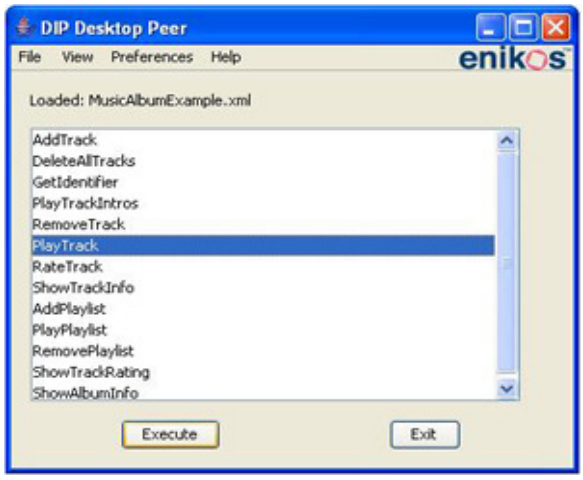



**Figure 14 Using the Enikos's Desktop Peer**

The Enikos's Peer provides support to play (using DIBO's play operation) various type of resources (text, audio, images, video, documents). Enikos has also made available two DID templates to be used with the demo Peer. The first template is a music album with some added functionality like the management of playlists and ratings for the tracks. The second template contains a Learning Objects example, being a structured representation of a number of lessons, notes and related material that a student can access. The Learning Objects template illustrates the applicability of MPEG-21 in the various related multimedia areas, among which the e-Learning area that is currently very popular.

The system proposed in this thesis intends to develop the DIP functionalities based on the reference software used by Enikos. Therefore some similarity may exist between DIP Desktop Peer and the part of the proposed system implementing these specific functionalities.

## 4.2. Klagenfurt demos

Klagenfurt University (Austria) has been constantly contributing to the standardization of MPEG-21, especially to the part 7 – DIA, and in July 2006 they organized the 77$^{th}$ meeting of MPEG. The Department of Information Technology (ITEC) of the Klagenfurt University developed the DIBuilder and DIConsumer applications: the former is used for creating specific Digital Items containing video files and the latter for viewing the previously created Digital Items [50].

### DIBuilder

The Digital Item Builder (DIBuilder) is a Java applet application that allows Users to create simple Digital Items according to MPEG-21 specifications. These Digital Items are generated for video resources and may include *Descriptor* elements containing general

descriptions about the *Items*, *Choices* defining the subset of available resources that can be selected by the User, *Component* elements with MPEG-7 descriptions of the video resources, and may also have included *Licenses* to grant rights to *Principals*.

The DIBuilder applet has an intuitive interface and may be used by any User. It makes the process of Digital Items creation very easy.

The advantage of using the DIBuilder is that it can export a video file and the other Users may consume it during the processing of the corresponding Digital Item. It also includes a viewer for video files. The application is limited to create Digital Items only for MPEG-4 video resources and this affects its applicability within the MPEG-21 framework.

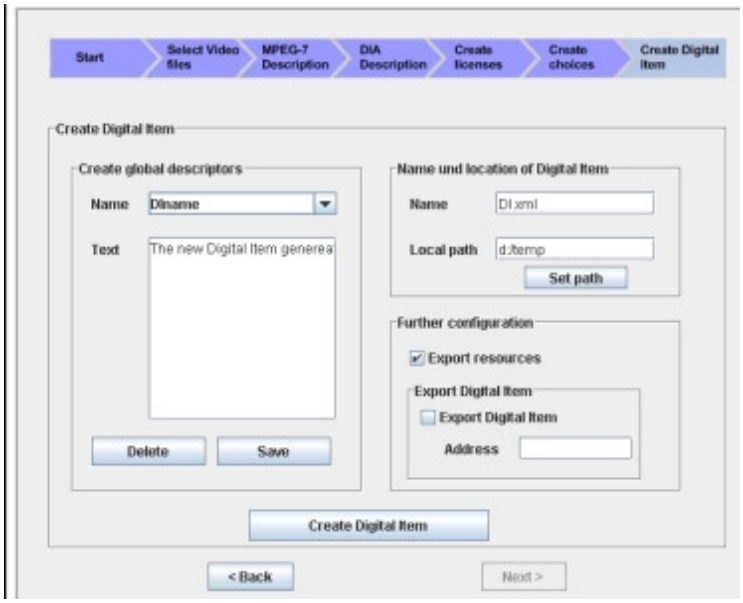



**Figure 15 Generating a Digital Item using DIBuilder**

### DIConsumer

The other terminal application developed by Klagenfurt University is the Digital Item Consumer. It is also a Java applet that can be viewed in various Web browsers.

The Digital Item Consumer (DIConsumer) enables the Users to consume simple Digital Items, but only those created with DIBuilder, and to view the video files with the corresponding metadata, at the same time respecting the rights granted by the *License*.

When using DIConsumer, the consumption of the Digital Items is made in three steps: 1) the User selects the Digital Item to be consumed; 2) then the resources are selected; and 3) the resources are processed and the video is shown in a simple viewer and the corresponding metadata is displayed in an XML viewer.

The main characteristics of the DIBuilder and the DIConsumer are: easy to use, accessibility through Web browsers, fast runtime performance and support for parts 2, 3, 5, 6 and 7 of MPEG-21.

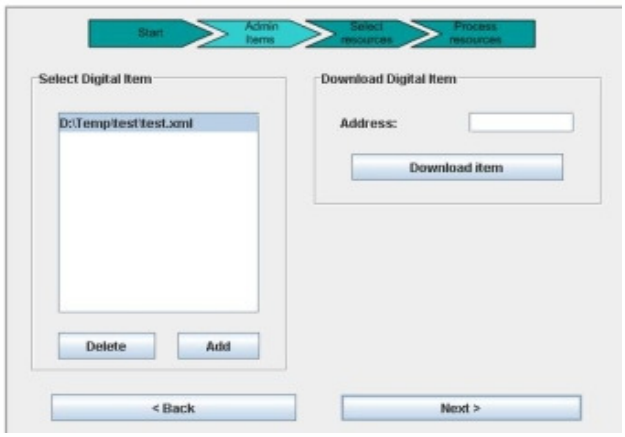



**Figure 16 Selection of the Digital Item to be processed in DIConsumer**

## 4.3. AXMEDIS Player

AXMEDIS is a large Integrated Project of Research and Development [51] that aims to build a framework that permits the digital content producers, aggregators and distributors, to access a wide range of innovative technologies, creating new markets and distribution openings. Within this project the AXMEDIS Tool Core for MPEG-21 content authoring and playing was developed [52], and may be used for browsing MPEG-21 Digital Items and for editing the attributes of the Digital Item elements. An authoring tool enables the User to manage and compose contents. The User's actions on the contents are controlled and validated by the authoring tool. The validation is made against the licenses owned by the User and related to the managed content. The AXMEDIS authoring tool is also a Digital Item player as described below.

When rendering Digital Items by using the AXMEDIS player, the User is presented with: a tree view window containing the hierarchical structure of the MPEG-21 DID; a viewer/editor dialog for editing the attributes of a specific DID element; and a manager window which shows all the opened Digital Items with all the activated views. In the tree view of the MPEG-21 DID structure the User can add/delete elements, by using the cut-copy-paste and drag-drop functionalities. This tool also supports the rendering of the embedded resources of Digital Items. The Digital Item creation is based on modifying other existing Digital Items, when the needed licenses have been acquired.

In Figure 17, on the left side, the tree view window for Digital Item browsing is illustrated and, on the right, the dialog for editing the attributes of the Digital Item elements.

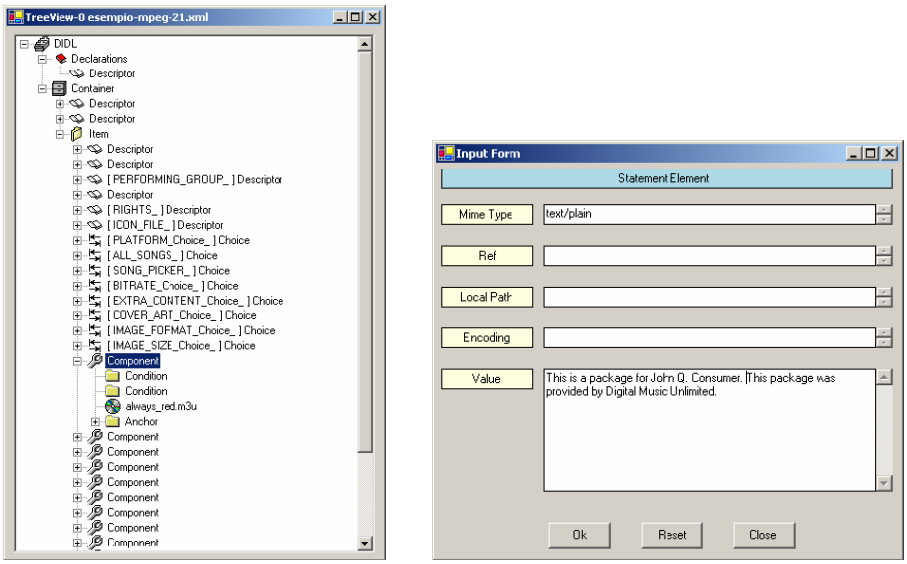



**Figure 17 Tree view window of the Digital Item structure (on the left side) and the dialog for attribute editing (on the right side) [51]**

This AXMEDIS tool is flexible regarding data access and it has a high level of security when resources are used for authoring or for visualization, playing or execution. It implements parts 4 and 5 of MPEG-21.

## 4.4. INESC DID Browser

In the ENTHRONE project [1] (more information about ENTHRONE can be found in section 3.4), INESC Porto [4] has implemented the DIDBrowser application which initially started as part of a Master degree thesis [14], developed within an internal project of INESC Porto named Multivision. The Multivision system has three sub-systems: DI Consumer, Multivision Server and DI Management. It has a client-server architecture created for demonstrating the concept of multimedia content adaptation using a set of tools based on MPEG-21. When implementing the DIDBrowser for ENTHRONE, the DI Consumer subsystem of Multivision served as a starting point. The DIDBrowser is a system that ensures the Digital Item search, resource selection and consumption, and also ensures the DI rendering, focusing on the DID part of MPEG-21.

As the present thesis was developed at INESC Porto the resulting work took advantage of some of the basic ideas of the INESC DIDBrowser and therefore a more detailed description for this application will be given here.

The first version of INESC DIDBrowser was a Java desktop application that can be run on a PC and is capable of accessing local and remote Digital Items containing media resources. Later, in February of 2006, the application was transformed into an applet. Thus the DIDBrowser became an application accessible by common Web browsers. A month later INESC developed a demo version of the DIDBrowser for PDA devices, which was based on client-server architecture due to the poor processing capabilities for this type of devices.

In the next sub-sections some of the main concepts of the DIDBrowser are described: the mapping of DIDs into Java Object Model and the browsing mechanism, as well as the functional architecture, enumerating all the internal modules and their roles.

### 1) The implemented MPEG-21 DID Object Model

For retrieving and fast rendering of the Digital Item content, it is necessary first to process it, but to do so an object model representation for the DID (which is an XML file defining the Digital Item structure) is required. The developed Object Model module used by the DIDBrowser consists of a set of hierarchical classes, implemented in an object oriented language (Java). This set maps the elements of a DID instance, defined in parts 2, 3 and 7 (partially implemented) of MPEG-21. Once created the objects of this model take into account the syntactic and semantic rules defined in the corresponding DID and implicitly in the MPEG-21 standard.

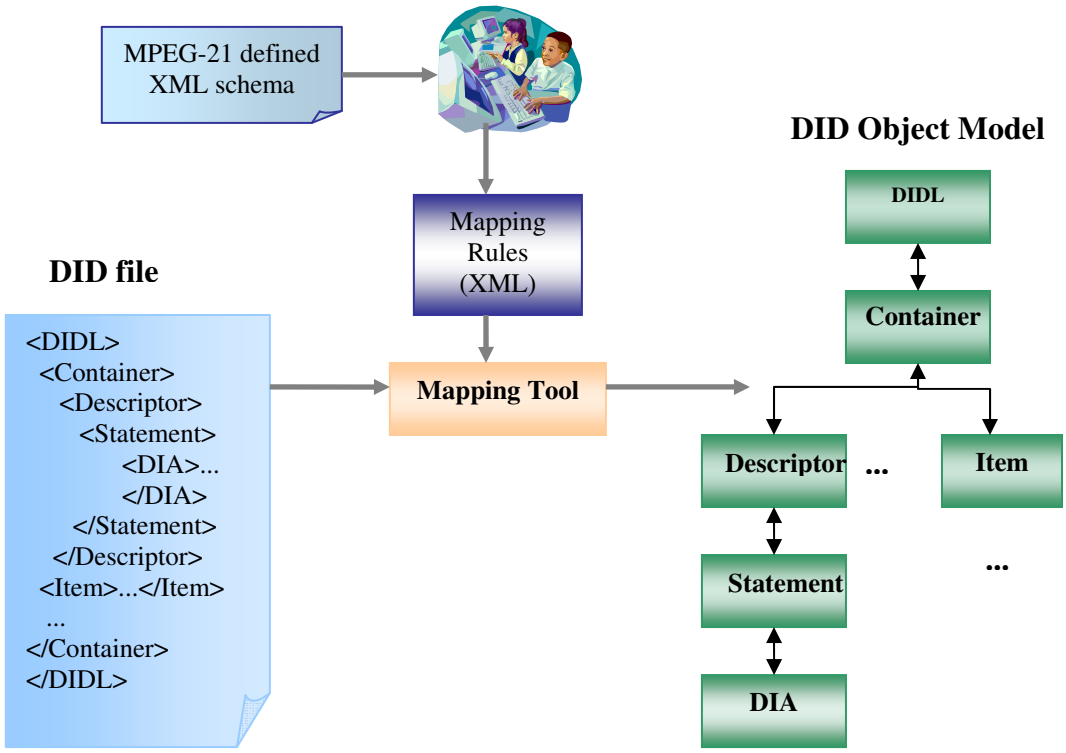



**Figure 18 Mapping a DID into an object model [14]**

For mapping a DID into the correspondent Object Model instantiation and vice-versa at runtime, an external tool is used (Sourceforge JiBX [53]) that needs a set of rules for driving the mapping. The mapping rules are defined in XML and specify which object corresponds to which defined MPEG-21 element [54]. The mapping rules must be specified at the implementation time and, once these rules are defined, they are used by the external mapping tool to automatically convert declarations of Digital Items into objects and vice-versa [14]. The process of mapping DID to Object Model is illustrated in Figure 18.

The great advantage of using the Object Model module is that no additional processing of XML at runtime is necessary, and that would be time consuming.

### 2) Digital Item Browsing

The DIDBrowser has the look and feel and some functionalities of the ordinary Web browsers. It presents the User with a Web page style view of the Digital Item, showing the descriptions and the resources contained in the corresponding DID. The presentation is made progressively following a step by step philosophy. More exactly, the User will not see the whole DID content at once, as it is the case of other similar applications (e.g., Enikos DIBrowser). At each step the User visualizes an *Item* or a *Container*. The navigation between the *Items/Containers* can be considered as being almost the same as the navigation between Web pages, where the navigation from one step to another is enabled by specific hyperlinks used to indicate the parent and children DID elements (*Items* and *Containers*) at the current step.

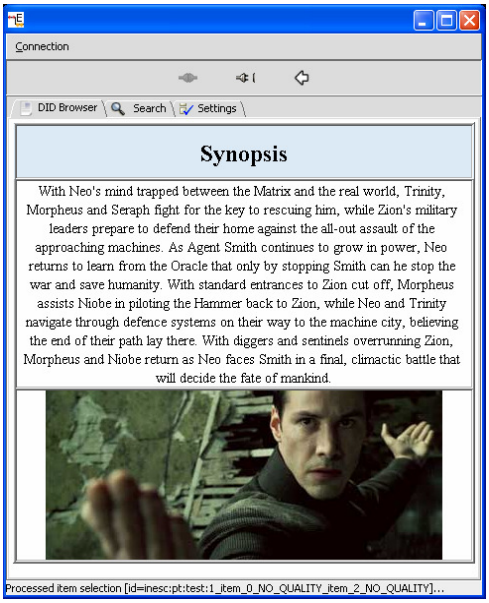



**Figure 19 Presentation of the generated HTML content in DIDBrowser, extracted from [54]**

The content of the descriptions, independently of their nature (text, image, etc.) or location, is retrieved automatically upon the opening of the DID. The application supports

automatic display for images as Resource elements (see definitions for Item, Container, Descriptor and Resource element in sub-section 2.3.2). The DIDBrowser has support for: simple text, TV-Anytime and media resources (e.g., images, videos). The TV-Anytime information is formatted using XSLT stylesheets for a more intuitive look. The DIA metadata are filtered and not displayed because they contain terminal specific information that has no meaning for the ordinary User.

The User graphical interface is based on HTML pages produced as a result of the rendering process using some internal templates. These templates mainly specify the layout of the content to be displayed.

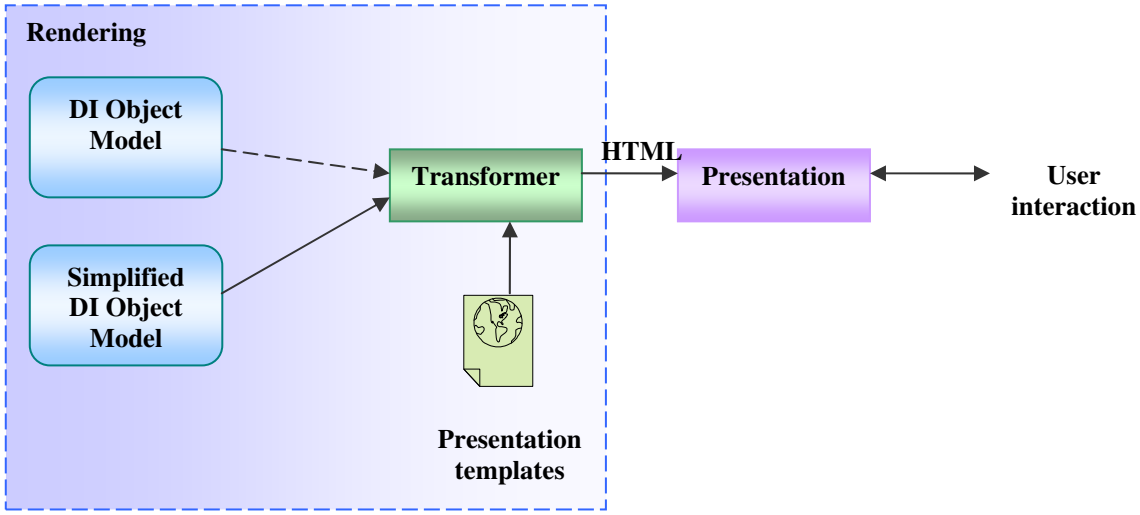



**Figure 20 Obtaining the HTML view of the DID content**

In a typical interaction of the User with the DIDBrowser, he can write the URL of a DID in the corresponding text-area field or can search for a specific resource contained in a DID introducing the values for publication time, publisher, genre, title, service information name or synopsis (these parameters are TV-Anytime specific) corresponding to the resource.

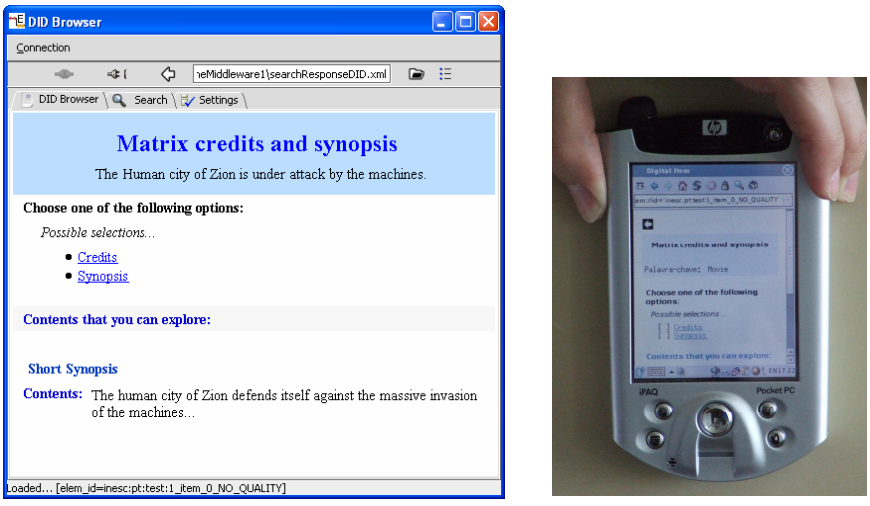



**Figure 21 Browsing a Digital Item with INESC DIDBrowser; on the left is the desktop PC version and on the right is the PDA version; extracted from [54]**

When opening a Digital Item, the DIDBrowser sends the URL of the desired DID and receives as response the respective DID. When searching for a DID, another specific request is sent, containing the search parameters/fields provided by the User. The response passed to the DIDBrowser will be a list of DIDs containing the matching Digital Items. The list itself is automatically generated as an ordinary DID to be consumed in the DIDBrowser. The User can select a DID and then browse its content. Then he may get back to the DID list and start all over again.

When it is the case, the requested media resources will be played by an external player, possibly after some adaptation.

### 3) Managing User selections

At each browsing step the User also interacts with other specific hyperlinks corresponding to the MPEG-21 *Choice*(s). They are used to register the User preferences and influence later the DID browsing by dynamic content rendering according to those choices. The DIDBrowser manages such selections and assures a way for rolling back to previous states by pressing the Back button. As a consequence, the preferences are discarded.

### 4) DIDBrowser Architecture

The application has a modular architecture structured on various layers as presented in Figure 22:

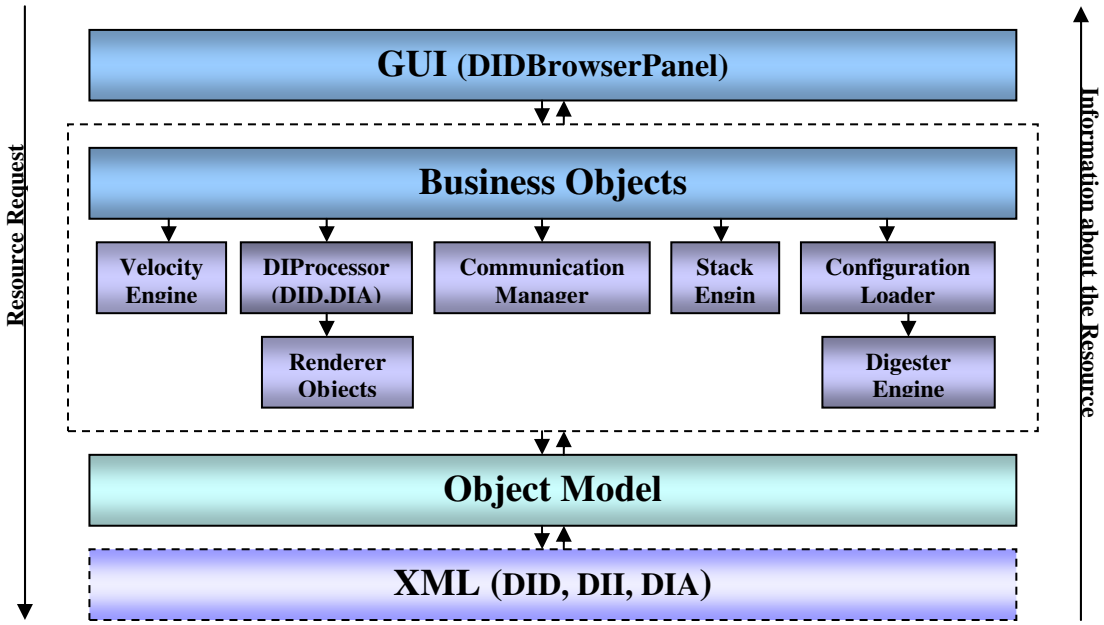



**Figure 22 INESC DIDBrowser architecture**

The GUI level is dedicated to the interaction and represents the interface with the User; the Business Objects level is intended to provide the DIDBrowser internal processing. The Object Model consists of a set of related classes, defined in an object oriented language, which maps the elements defined in the DID, DII and partially DIA parts of MPEG-21. The next level (XML) contains the Digital Items represented as an XML structure. When the User requests a certain resource, the communication between the architecture levels is top to bottom, from the GUI to the Digital Items (XML layer), and the response containing the information about the requested resources is sent upwards to the top level GUI, and finally passed to the User.

Figure 23 presents the structure of the modules which implement the functionality for the browser application developed by INESC Porto.

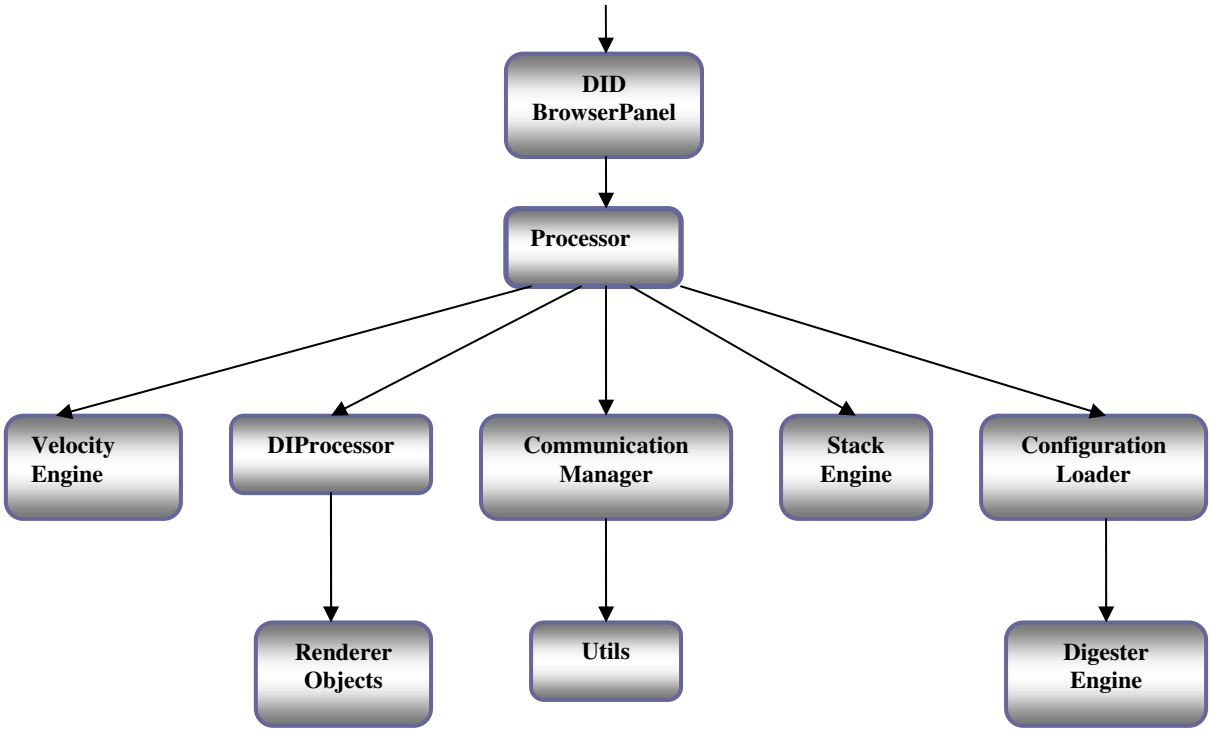



**Figure 23 INESC DIDBrowser modules**

Next the meaning of each module is described:

**Velocity Engine**: generates the HTML view of the browser based on a provided template.

**DIDBrowserPanel**: shows the browser panel (the interface with the user). It contains the GUIManager and Objects Collector sub-modules as represented in Figure 24.

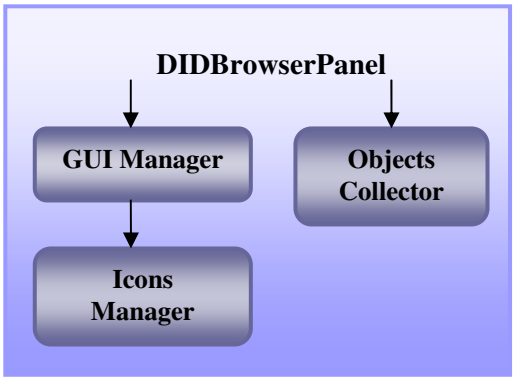



**Figure 24 Sub-modules of DIDBrowserPanel**

**GUIManager**: contains the GUI components for the DIDBrowser.

**Icons Manager**: is a container for all the icons used in the GUI.

**Objects Collector**: contains general purpose objects like those representing dialog windows, display resolutions, terminal information, resource rendering mappers (that make the mapping between resources and the correspondent players).

**DIProcessor**: processes Digital Items (includes DID and DIA parts of MPEG-21).

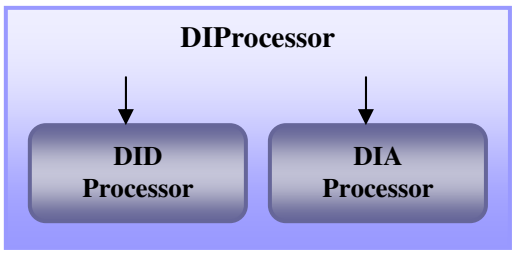



**Figure 25 Modules of DIProcessor**

**Renderer Objects**: contains the simplified objects of the Digital Item corresponding to those from the Java Object Model presented above, that is more complex; in this way it provides faster and more flexible processing.

**Communication Manager**: is composed by the Communication Client and DIReceiver Engine modules.


**Figure 26 Sub-modules of the Communication Manager module**

**Communication Client**: establishes the communication with an external application used basically to provide download support to the DIDBrowser.

**DIReceiver Engine**: presently is not in use because it was replaced by the Communication Client; it provides the same functionality, without calling an external application. It is possible that in the future functionality of the DIDBrowser it will be included again.

**Utils**: contains utility functions like the conversion of the file content to string.

**Configuration Loader**: loads configuration files.

**Digester Engine**: is used for the mapping of XML to objects, being similar to the JiBX tool.

**Stack Engine**: used to collect and manage the states of the DIDBrowser for the Back button functionality.

*

INESC DIDBrowser was designed as an easy-to-use, friendly graphical interface and a flexible front-end for Users to browse Digital Items. Due to the Java implementation it benefits from the ability to be ported to many operating systems supporting the Java platform. Other ability is the fast processing of the Digital Items as a consequence of mapping the Digital Item content into a hierarchical Java object model, as a preprocessing phase before Digital Item rendering.

### 4.5. Remarks

Implementations of MPEG-21 Terminals (Peers) for visualizing and interacting with Digital Items are still scarce. Solutions have been achieved for ensuring accessibility, flexibility and performance of the Peers on specific terminal devices. Among all the MPEG-21 Peers presented in this chapter, none is portable to all types of devices. The client-server architecture, such as the one used by the MPEG-21 Peer for mobile devices developed at Wallongong Universtity (presented in Chapter 3), could be an advantage in future works to increase the portability on heterogeneous terminal devices. This architecture delegates on the server the responsibility of Digital Items processing, leaving the client with the content presentation.

Although for many of the applications presented here, functionalities for various parts of the MPEG-21 standard were developed, none provides support for all the published parts in a single Peer. As MPEG-21 is a novel technology, in the near future these applications are expected to encompass all the MPEG-21 concepts and functionalities of the multimedia framework.

When referring to navigation and dynamical User interaction (this functionality is specified by MPEG-21 DIP) with Digital Items, until the moment a separation was realized between these two functionalities. Each one works on a different Peer and they cannot be achieved simultaneously within the same application, as in the case of Enikos products: they launch an application for navigating/browsing within Digital Item content named DIBrowser and they create a separate demo application only for the DIP functionality. The Enikos DIBrowser has support for the static User interaction with the DID consisting in the possibility of configuring the User's preferences for some resources, on a DID *Choice* element.

It is also important that an MPEG-21 Peer supports the MPEG-21 Digital Items created by different applications. For example, the Klugenfurt DIConsumer is restricted to the visualization of specific Digital Items created by the other Klugenfurt system – DIBuilder; as a consequence this will affect its general compatibility with the other tools within the MPEG-21 multimedia framework, although the created Digital Items are valid according to the standard.

The main objectives of the present dissertation are related with solving the portability problem for this type of applications and also with finding a solution to integrate the DIP functionality in a browser system used for Digital Item consumption. This new browser will be named MPEG21 DI Browser and will support the navigation through the content of all MPEG-21 Digital Items. The MPEG21 DI Browser is intended to be as flexible as possible, separating the internal processing part from the graphical interface and centralizing all the processing on a server sub-system, which will permit the ulterior addition of new functionalities, including support for other parts of MPEG-21 (Digital Item Adaptation – DIA, Right Expression Language Components – REL, Event Reporting – ER, etc). For these considerations the proposed application will implement a client-server architecture.

# Chapter 5

## 5. MPEG21 DI Browser

Although the work on the MPEG-21 standard started some time ago, some of its parts are still evolving as well as the interest in using its technologies. The need for a system or application that allows users to consume content as MPEG-21 Digital Items is arising. In this context it is motivating to implement an advanced system for the visualization of the multimedia resources and of the corresponding descriptions. Such a system has to permit multiple interactions in order to provide an adequate visualization in the User's context: his preferences, the terminal characteristics or constraints and the capabilities of the distribution network.

The proposed MPEG21 DI Browser intends to provide a browser application for consuming multimedia contents as MPEG-21 Digital Items in a Web environment. It will follow a new philosophy regarding the architecture for this type of applications when compared with the very few similar tools that are currently available. The processing and browsing of Digital Items will be made "step by step" and not all the content at once as it was done until now on most of the implemented MPEG-21 terminals. This approach is expected to reduce the presentation time (the time until the content is loaded and presented to the User) of the Digital Item chosen by the User.

This chapter presents and analyzes the requirements for the proposed system, as well as the corresponding specification.

### 5.1. Requirements for the MPEG21 DI Browser

The system should provide browsing through the content of the Digital Items and has to allow Users to consume the multimedia resources contained in the respective Digital Items. The consumption and User interaction should be achieved in a dynamic way. The User must have the freedom to specify operations that may be applied to the Digital Items or parts of them.

### 5.1.1. Basic requirements

Related to the processing of Digital Items and the implementation of the new MPEG21 DI Browser, several requirements were identified:

- Browsing of Digital Items stored on a remote repository – the User must be able to access and consume Digital Items that are stored on a remote repository;
- Download of Digital Items – it is required to transparently download the Digital Items from a remote repository in order to process them and to ensure their navigation;
- Validation of Digital Items – before being displayed, the Digital Items must be validated to assure that they are conformant with Part 2 (DID) of the MPEG-21 standard;
- Choices processing – the system must support the static User interaction with the Digital Item content by supporting the mechanism developed by part 2 of MPEG-21 which regards the User's selections;
- DIP support for dynamical interaction of Users with Digital Items, basic operations defined in MPEG-21 DIP will be available for the User permitting him to interact with and alter the Digital Item or its component parts. Examples of basic operations are: print, execute, play, wait, adapt, alert, etc.

### 5.1.2. Extended requirements

The scope of this thesis is the use of MPEG-21 technologies in order to provide solutions for the requirements enunciated above, materializing them in the implementation of the MPEG21 DI Browser system. If possible, as additional work, other features could be developed, such as:

- Browsing of Digital Items stored locally on the User's machine;
- Searching for Digital Items in remote (Web) repositories;
- Implementing REL support necessary due to the existence of protected contents and author rights to be guaranteed;

- Basic support for ER that can be integrated with DIP, in order to notify about the occurred events when the User is interacting with the Digital Item; this is necessary for better management and control of events;
- Creation of a database as a repository for Digital Items;
- Implementing a Session Management module to control the user sessions and for user authentication.

## 5.2. Considerations for specifying the system

The main aspects that need to be specified before developing the MPEG21 DI Browser system are related with: the modality of representing the parts of MPEG-21, the processing of Digital Items and the presentation of the content to Users. These aspects will be presented in the following subsections.

### 5.2.1. Object Model

During the processing of Digital Items it is required to manipulate the MPEG-21 information in a manner that ensures fast and easy access to its elements, contributing to overall performance of the IDI Browser subsystem. Therefore it is essential to create a corresponding object representation for mapping the XML elements of the MPEG-21 parts. The object model maps the elements defined in parts of the standard and also the relationships between them, into a set of related classes, implemented in an object oriented language.

The object model used in this dissertation will provide the respective representation of the DID and of other MPEG-21 parts (e.g., DII and DIP), meeting the rules and constraints defined in the schema files as they are defined by the standard.

However, the original XML content of the DID will be stored into the object model and will be used as an alternative in some particular situations (e.g., looking for some specific attribute in a tag element that is not defined in MPEG-21 and as a consequence does not have any representation in the object model).

### 5.2.2. The processing of Digital Items

The processing of Digital Items consists in preparing them for browsing. This means that their contents will be simplified in order to enable a fast navigation. The base elements included in Digital Items are the subjects of this simplification. In the context of the MPEG21

DI Browser, a base element represents the fundamental unit of a Digital Item that can be consumed during the browsing process (this is not normatively defined). The justification for choosing to browse a Digital Item by its base elements is given in 5.2.3.

From the variety of the MPEG-21 elements contained in the object model, the *Item* and *Container* can be considered the base elements for the browsing system that was developed for this dissertation. The decision of using these two as base elements is also justified in 5.2.3.

For optimization meanings, the simplified base elements (*Item* and *Container*) will contain only the relevant information for this version of the MPEG21 DI Browser. In Figure 27 can be noticed how the XML structure describing a base element will be altered during the processing.

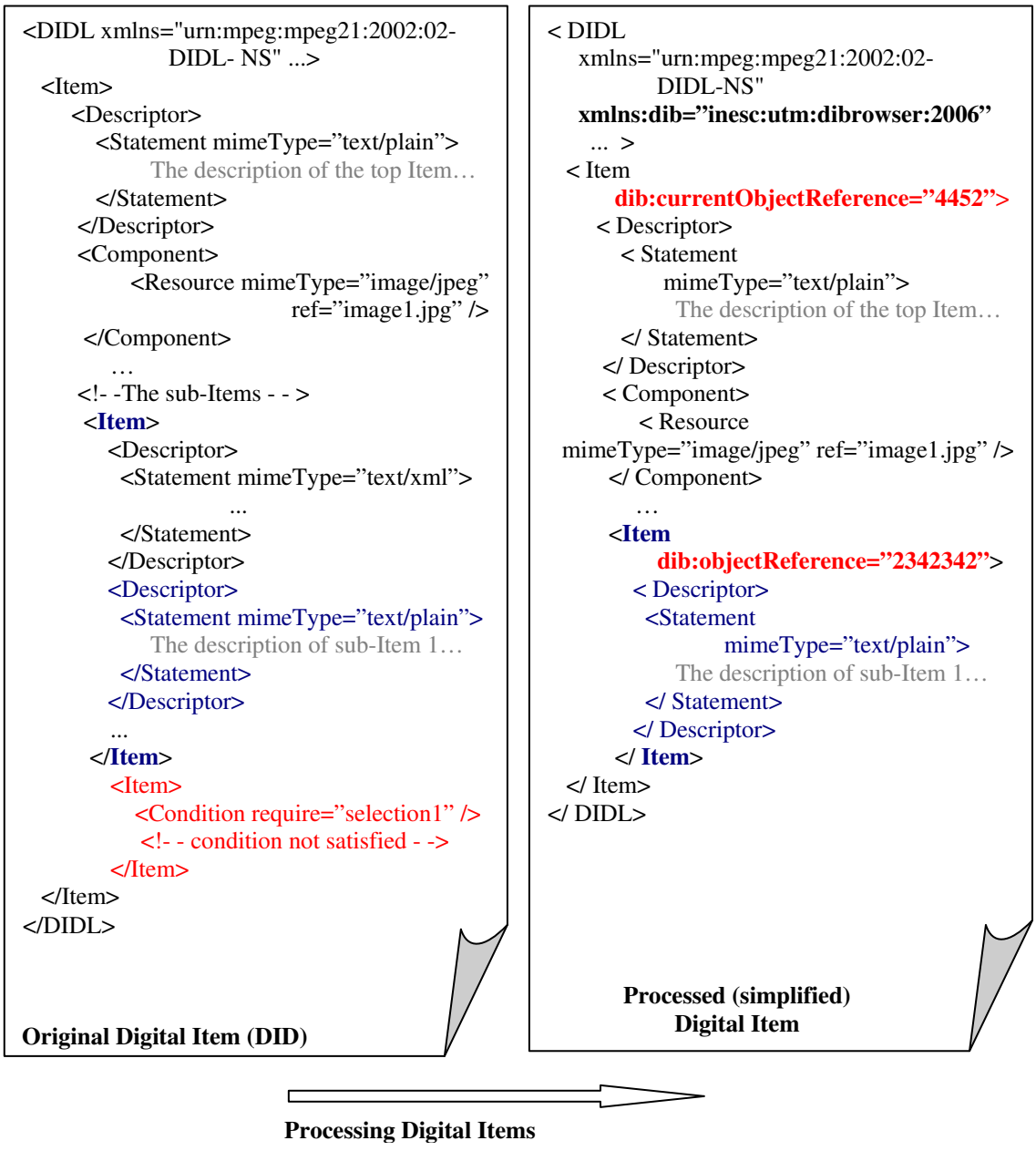
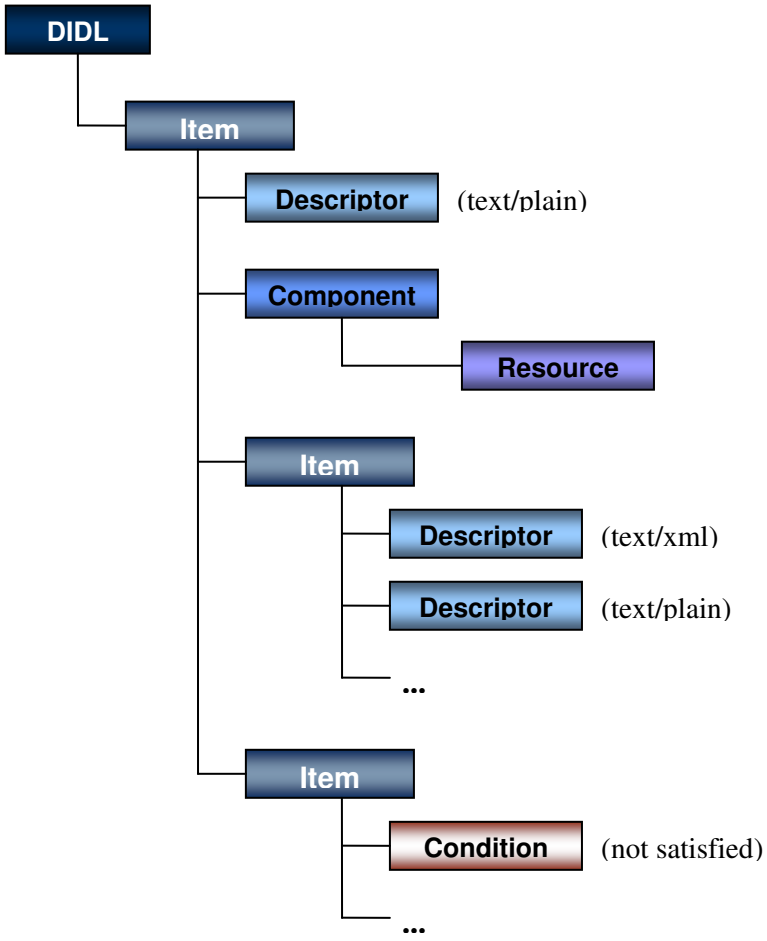
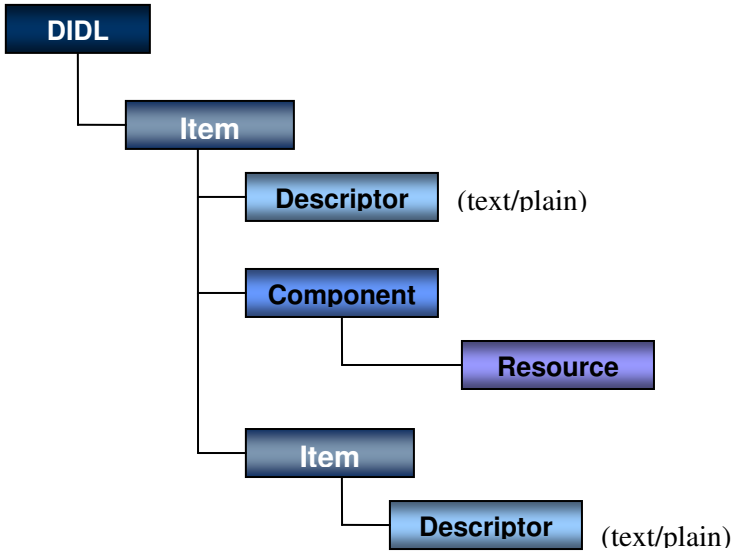



**Figure 27 Example of processing an Item element; the sub-Item that passes the filtering is marked in blue and the other sub-Item is discarded**




**a) The original element**




**b) The element after processing**

**Figure 28 Example of processing the base elements**

For example, during the processing, the elements that do not meet some conditions will be discarded; the sub-elements will be filtered and only their first textual descriptions and the

corresponding identifiers will be kept; the comments are not relevant for the common Users and may also be removed; and some additional information specific for the MPEG21 DI Browser may be included for management purpose of the correspondence between the base elements (see the *currentObjectReference* and *objectReference* attributes in Figure 27). Figure 28 illustrates the manner of simplifying the base elements during the processing.

### 5.2.3. Web philosophy

The presentation of the Digital Items contents to Users will be made in a Web pages style, following a Web philosophy.

The information will be presented progressively in more than a single step, except when the Digital Item has a basic DID structure. The decision to browse by pieces (which are base elements) has emerged from the difficulty to process at once the huge amount of information from a complex DID structure. In this case the visualization of the content is also affected, becoming slower than usually or limited, especially on "thin" devices such as PDAs and mobile phones. As the system proposed in this thesis is intended to be accessible from any type of devices, it is important to provide to them the support for performing processing and visualization of Digital Items. The solution is to enable an "element by element" processing and visualization.

At each step of the navigation, a base element (contained in the DID structure of the Digital Item) will be displayed to the User. As the *Item* is normatively defined as "the lowest level of granularity transacted by Users within the MPEG-21 framework" [9], it will be considered a base element for the Digital Items consumed by the MPEG-21 DI Browser system. A *Container* groups a set of *Items* and therefore it can also be seen as a base element.

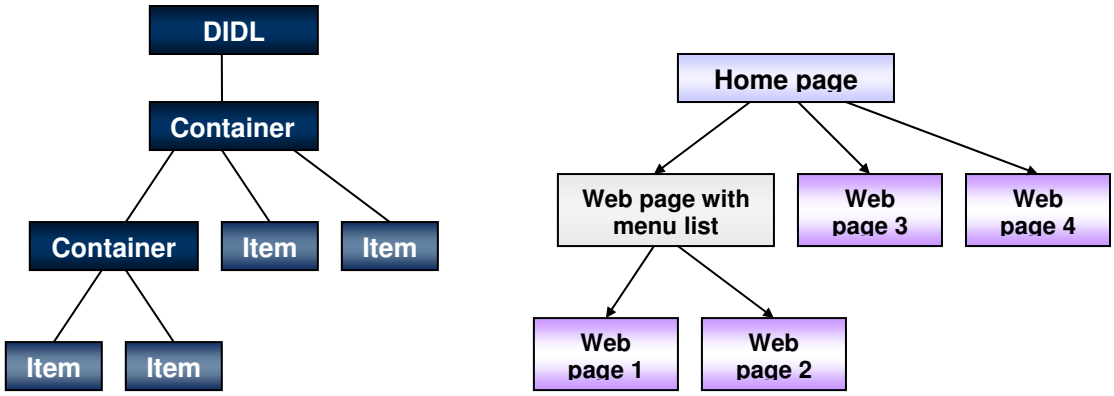



**Figure 29 Similarity between a DID representation (on the left) and a website (on the right)**

As it can be noticed in Figure 29, there is some similarity between the DID representation of a Digital Item and the manner of navigating through a website. Both are

based on a tree structure. This will facilitate the adoption of the Web philosophy when developing the mechanism for the navigation between the base elements of a Digital Item.

During the browsing through a Digital Item, the User will be presented at each step with the Web page corresponding to a base element. The hierarchical relationship between the base elements will be defined by hyperlinks.

When an element is visualized, the Web links pointing to its sub-elements will be displayed. In any moment the User will be able to navigate backwards between the elements.

A base element of type *Item* may contain information that is not made available to the User unless he performs some selections. During navigation, he is asked to express his preferences related to some parts of the content. In order to allow the Users to visualize the restricted information, an intermediary Web page is necessary for making those selections. In the intermediary page the values for the selections will be represented as HTML radio buttons. When this step finishes, the preferences will be processed and the User will see the parts of the base element that meet the selections made. For example, a selection may be set for choosing the resolution (size) of an image in order to present the desired version of that specific resource at the User terminal.

The advantage of applying this Web philosophy is that the Users are accustomed to Web pages navigation and it will be intuitive for them to consume the Digital Items using the MPEG21 DI Browser.

## 5.3. System analysis

### 5.3.1. Identification of the System Functional Architecture

In the following an analysis will be made for the proposed architecture for the MPEG21 DI Browser.

Three main entities can be involved in the browsing and consumption of Digital Items:

- A terminal device to connect the User to the Digital Item browsing system (MPEG21 DI Browser);
- The system (the MPEG21 DI Browser Peer) where the Digital Item processing necessary for the visualization and User interaction with Digital Items is performed;
- A remote repository where Digital Items are stored.

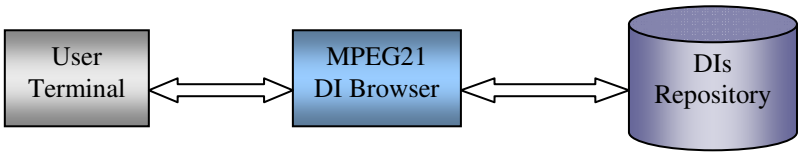



**Figure 30 MPEG21 DI Browser components**

A clear separation of the graphical interface from the business logic is essential when developing the application. With this separation and well defined APIs it is possible to provide the connectivity from one module to another permitting also the use of different graphical interfaces without the necessity of altering the business logic. If necessary, a middleware layer can be added to facilitate the support of different GUIs as it is shown in Figure 31.

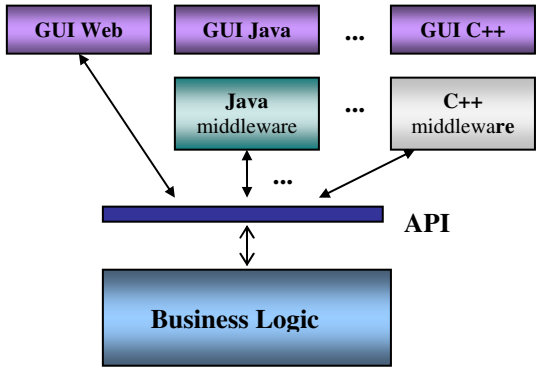



**Figure 31 Separating the GUI from the processing part**

The proposed system will use the concept of client-server architecture, where the client application is responsible for the content presentation and user interactions with the Digital Items, while the server is used for processing the respective Digital Items, registering User's preferences by processing his selections, verifying the conditions within the Digital Items and also sending the results of the User requests.

The system flexibility will be reflected in the possibility of reusing the business logic functionalities in other client applications permitting the use of different interfaces and different environments (as Web pages and/or client applications developed in different programming language: Java, C++, .NET, etc.).

The client application will be named WDI Browser (Web Digital Item Browser) and the server sub-system name will be IDI Browser (INESC Digital Item Browser). The MPEG21 DI Browser functionality will be based on the mentioned server application which will have a total control over navigation and the User's interactions with Digital Items.

Web Services represent a new distributed architecture for Web applications that ensures maximum interoperability and transfer of data in an efficient manner. Therefore, it was decided that the API of the IDI Browser sub-system of the MPEG21 DI Browser should provide as Web Services the methods defined in the interface between client applications and the internal processing part of the main server application. The client may be any application running on a specific device that can communicate through Web services or can include a plugin capable of using the API of the server. In this thesis the client application used for demonstration will be implemented in a Web pages style.

A Web Service can be seen as an application without graphical interface that provides on the Web a set of methods or operations that are published in a specific format, in order to be available for other applications acting as remote clients benefiting from that Web Service.

Figure 32 shows the communication (XML based) between a client application and a Web Service running on a server.


**Figure 32 The Web Service concept [56]**

On the server side, the Web Service provides operations implementing for example, for the IDI Browser, the functions for accessing Digital Items and processing parts thereof. On the client side, the application uses a proxy mechanism to reference the Web Service, which facilitates the call of any Web Service operation. The communication between client application and Web service is made by using messages in an XML format. The client application sends a message indicating the method that it needs to execute on the Web server and optionally some parameters. The server receives the message, executes the specified operation and sends to the client the results of the executed operation.

The types of exchanged messages between the client and server applications are depicted in Figure 33. The interactions between the User and the MPEG21 DI Browser sub-systems are:

(1) – the User accesses the Web client using a Web browser (e.g., Internet Explorer, Netscape, Firefox, etc.) and requests a Digital Item or selects an operation;

(2) – the client application requests the execution of the respective operation;

(3) – after the operation is executed, the Web Service sends the results to the client;

(4) – the User visualizes the results in the Web browser.

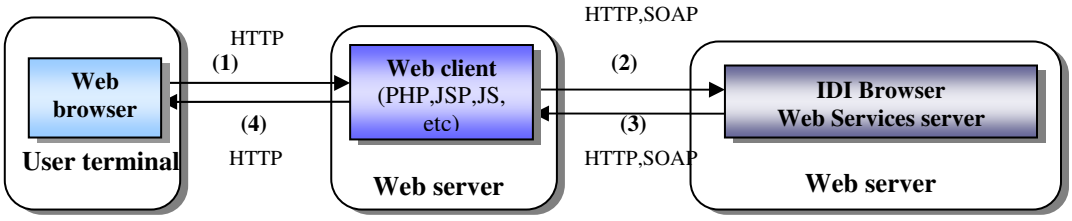



**Figure 33 The communication between User terminal and MPEG21 DI Browser components**

### 5.3.2. Architecture of the MPEG21 DI Browser

The general architecture of the proposed system for browsing and interacting with Digital Items is presented in Figure 34.

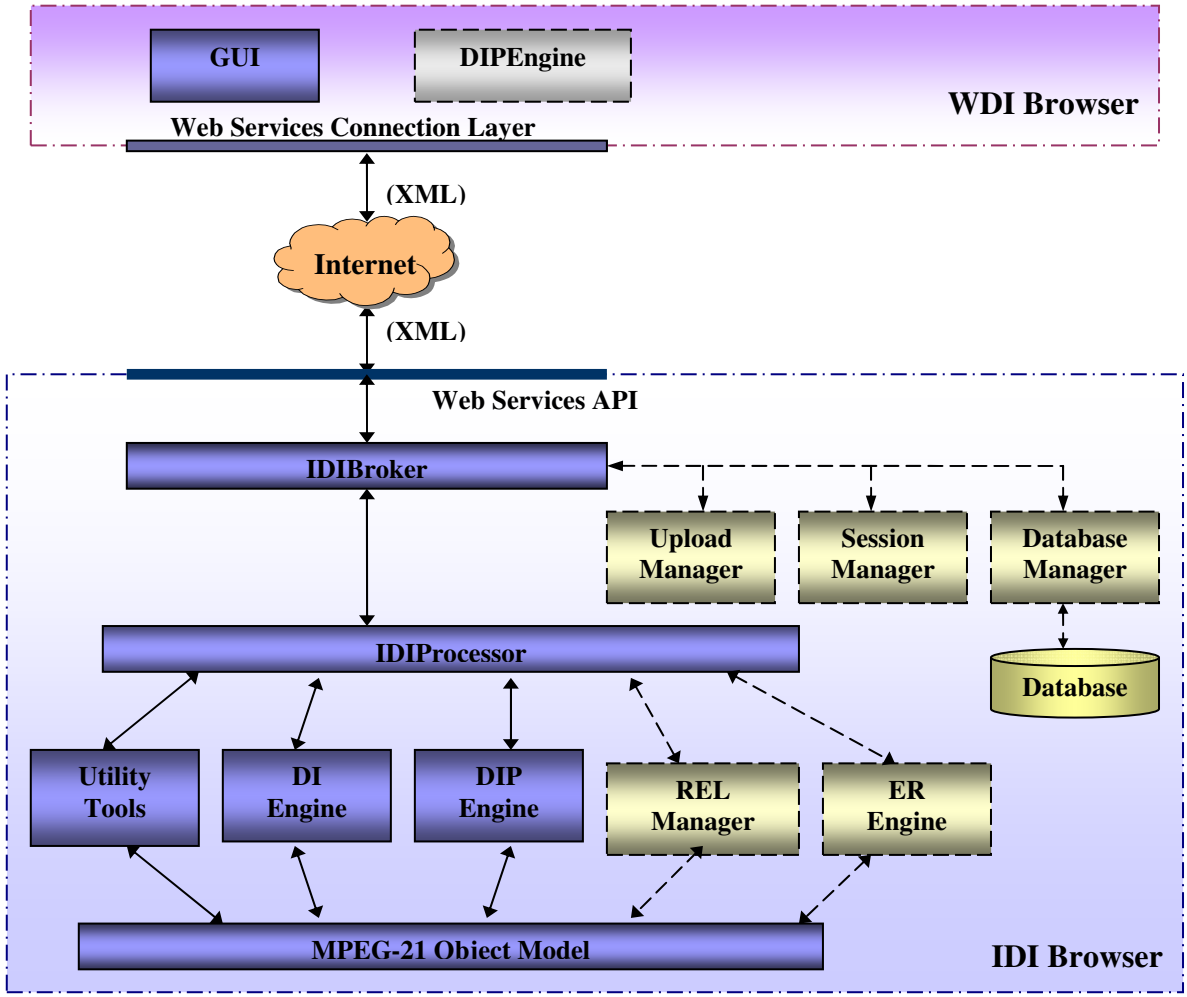



**Figure 34 The general architecture of the MPEG21 DI Browser**

The components of the application are:

- **WDI Browser –** it is the client application that generates the graphical Web interface provided to User to permit his interaction with the MPEG21 DI Browser system; the essentials sub-modules of the WDI Browser are:
    - **GUI** – it generates the graphical interface as web pages, visible in the User's Web browser;
    - **Web Services Connection Layer** – it is responsible for connecting with the server and for requesting the necessary operations for the presentation of Digital Items;
- **IDI Browser –** it is the server, the central unit of the MPEG21 DI Browser system, ensuring the processing of Digital Items and providing the operations of the Web Service that any client application needs for presenting Digital Items to Users. The component modules of the IDI Browser are:
    - **Web Services API** – it represents the interface between the User actions and the internal processing part of the application; it defines the Web Services operations available to the WDI Browser;
    - **IDIBroker** – it is an intermediary layer that assures the communication between the various modules of the system;
    - **IDIProcessor** – it is the central module, implementing the general processing part of the MPEG21 DI Browser;
    - **DIEngine** – it is dedicated for processing Digital Items at DID level, working with the corresponding objects and simplifies the DID contents for a faster and more intuitive browsing, enabling the "step by step" navigation;
    - **DIPEngine** – it is essential for realizing the dynamic interaction of the User with the Digital Items; it implements Digital Item Basic Operations (DIBOs), which are operations that define the methods (DIMs) available for a Digital Item; the DIPEngine executes these DIMs. Optionally, the DIPEngine or some of its functionalities may be developed at client side;
    - **Utility Tools** – it contains tools needed for MPEG-21 processing and also for other specific operations, such as downloading external Digital Items to the IDIBrowser server in order to be processed and then sent to the client application for being presented to the User;

- **MPEG-21 Object Model** – it is the low level module that realizes the mapping of the MPEG-21 elements (DID, DII, etc.) into objects.

There are also some additional modules (marked with light-green in Figure 34) that can be created to meet the objectives of the extended requirements (see sub-section 5.1.2):

- **REL Manager –** it verifies the existence of the licenses included in the Digital Item Declaration, sends a notification to the User to purchase a license and gives authorization for the consumption of the Digital Item;
- **ER Engine –** it implements the functionality of Part 15 – Event Reporting of MPEG-21;
- **Upload Manager** module – it is part of the Web application and is useful for uploading Digital Items from the User machine to the repository and also for downloading a Digital Item to the User terminal;
- **Session Manager –** for the identification of Users and for controlling the User sessions;
- **Database –** it contains the descriptions of *Items* in Digital Items, information about User licenses (e.g., usage limit) and information about User sessions;
- **Database Manager –** a transparent and flexible interface for accessing the database.

IDIBroker is the module that intermediates the communication of these additional modules and the other internal processing modules, except for the REL Manager and ER Engine which use IDIProcessor for the interconnection with the other modules.

The messages exchanged between the sub-systems (client and server) of the MPEG21 DI Browser will be based in the DID representations for the parts (base elements as were defined in 5.2.2 and 5.2.3) of the current browsed Digital Item, as a consequence of the "step by step" processing and visualization (see 5.2.3). At each step of navigation the client application requests a base element and the server sends as response the processed (simplified) version of this element, in an XML format, as normative DIDs. Such an XML message may be named "mini-DID", as it encapsulates only the simplified element. In other words, the communication between WDI Browser and IDI Browser is based on these "mini-DIDs".

In order to make functional the proposed MPEG21 DI Browser system, obviously, it is required to have connections to the Internet on the machines where WDI Browser and IDI Browser are running. Also the User's terminal device has to be connected to Internet for accessing the system (see Figure 35).

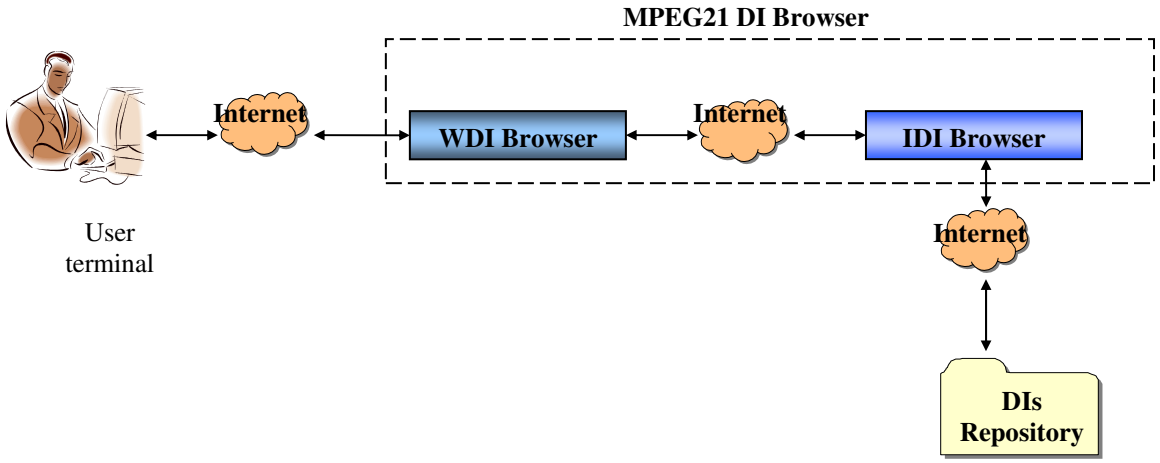



**Figure 35 MPEG21 DI Browser, a Web accessible system**

## 5.4. UML specification of the MPEG21 DI Browser

For defining the MPEG21 DI Browser system the Unified Modeling Language (UML) was used, since it enables comprehensible specifications, utile for the development process. The following sub-sections illustrate the UML diagrams identifying the User's interactions with the system, and the MPEG21 DI Browser functionalities meeting the requirements enunciated in 5.1.1.

### 5.4.1. Use cases diagram

This type of UML diagrams are usually created when it is necessary to illustrate the utility of a system or application. They represent the use cases for the respective system. A use case is a technique for capturing the functional requirements of the system being under specification. Each use case provides one or more scenarios that convey how the system should interact with the users, called actors, to achieve a specific goal or function. Use case actors may be end-users or other systems (e.g., client applications).

The User (Digital Item Consumer) is the actor interacting with the MPEG21 DI Browser in order to browse and consume Digital Items. The User accesses the application through a Web graphical interface in a common Web browser. Referring to the User interactions with the system, they can be defined through the use cases shown in Figure 36.

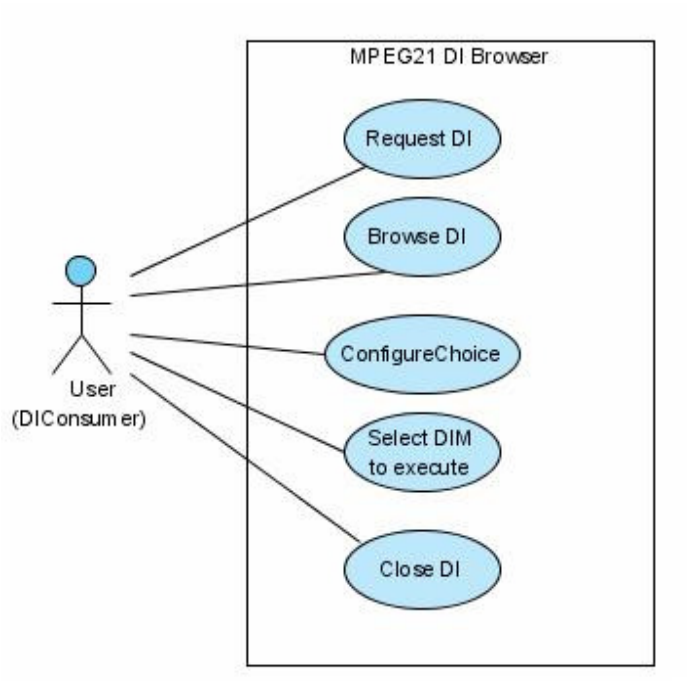



**Figure 36 Use cases diagram for User (DIConsumer) interaction with the system**

Description of the use cases for Digital Item Consumer:

- Request DI

  When requesting a Digital Item, the User will provide to the MPEG21 DI Browser the path to the corresponding Digital Item stored on a remote repository. As response, the MPEG21 DI Browser will present the top element from the DID (*Item* or *Container*) to the User.

- Browse DI

  While browsing the DID content, the User will be faced with Web pages corresponding to DID elements (*Items* or *Containers*), including links that point to their sub-elements content.

- Configure *Choices*

  To select a resource for consumption, the User has to make some choices expressing his preferences. The most suitable ones will be chosen, by fulfilling some conditions related to the User terminal characteristics, to the purchased license or to other aspects considered to be necessary for the Digital Item presentation.

- Select DIM

  When the Digital Item contains DIM methods, providing the DIP functionality, a list with the available DIMs will be displayed to the User. He may then select a DIM method to be executed by the DIPEngine.

- Close DI

  When the User wants to abandon Digital Item browsing, he may request to close the respective Digital Item. This command will clear the Digital Item from the processing system.

Considering as an actor the client application – WDI Browser, the use cases will be focused on the operations provided by the server application – IDI Browser, as it can be seen in Figure 37. The WDI Browser will present to the User the Digital Item content and will facilitate the navigation, but to succeed in that, it needs to call the operations from the server that do the necessary processing.

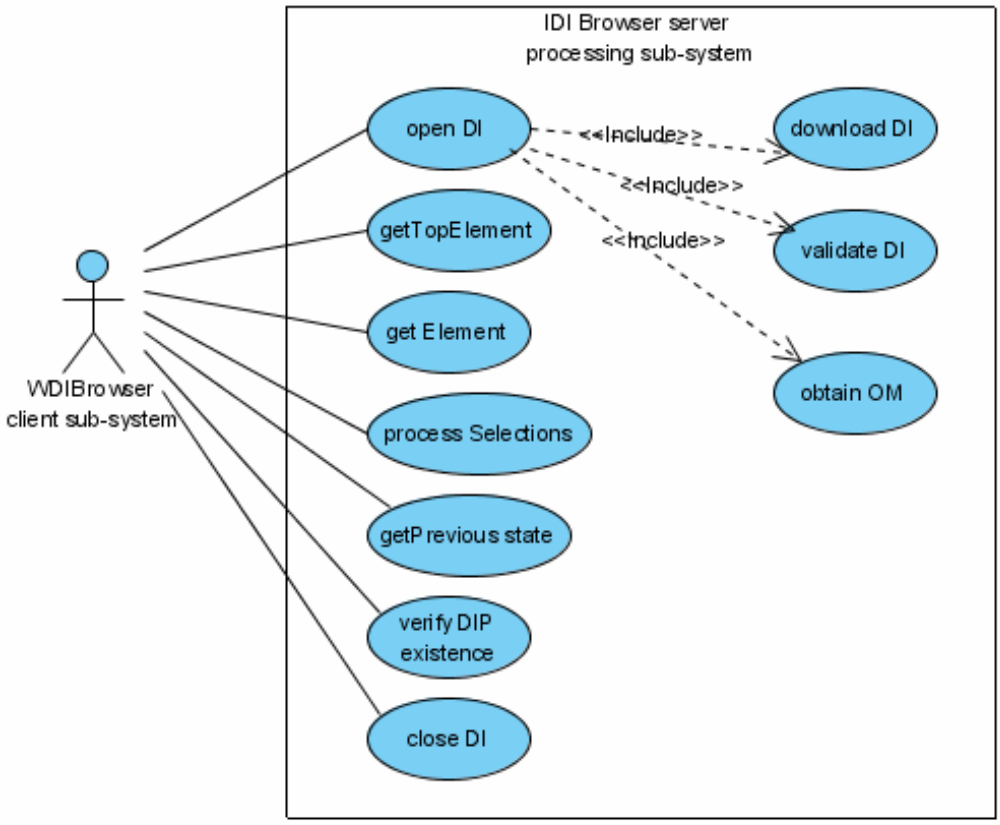



**Figure 37 Use cases diagram for WDIBrowser (client) interaction with the IDIBrowser (server)**

The identified use cases are:

- Open DI

  When the User is requesting a Digital Item, he sends its URL to the WDI Browser and this is then passed to the IDI Browser. On the server side this request is processed and includes three phases: the downloading of the Digital Item from an external Web repository to the IDI Browser server, the

validation of the downloaded Digital Item and then, if it is valid, the corresponding DID representation is mapped to the Object Model. As response the WDI Browser is notified if the Digital Item was successfully opened or not at the server side.

- Get Top Element

  After the Digital Item is opened on the server side, the WDI Browser can request the first (top) element contained by the DID. This element may be an *Item* or a *Container*. The top element will be presented to the User by the WDI Browser.

- Get Element

  If the current displayed element (*Item*/*Container*) contains other sub-elements, upon the User request for one of the sub-elements the WDI Browser will call the *getElement* service operation of the IDI Browser. As response it will receive the XML string of the respective element.

- Process Selections

  When the User sets his preferences by configuring the *Choice* elements contained in the current displayed *Item*, the selections set will be processed on IDI Browser in order to sent the filtered *Item* to the WDI Browser that will present it to User.

- Get Previous State

  In any moment the User may request to return to the previous element (*Item*/*Container*) and in this case the WDI Browser will request the IDI Browser to execute the *processBack* operation to return to the previous state. In this manner a better control will be possible over the content that is displayed by the WDI Browser to User.

- Verify DIP existence

  When the requested Digital Item contains DIMs that can be executed by the DIPEngine, it is useful to show to the User a list with the available DIMs, but for doing this it is necessary first to ask the server that is processing the Digital Item if it contains DIP information (DIMs).

- Close DI

  When the User wants to quit browsing of Digital Item he will notify the WDI Browser GUI, and the client application will then pass the notification

to the IDI Browser server where the Digital Item is cleared and the processing ends.

### 5.4.2. Activity diagram

The activity diagrams of the UML notation are useful to represent the activities flux of all use cases for the Digital Item consumer (the User). This type of UML diagrams describe the available actions that can be carried out by the actors (in this case: User or WDI Browser client application).

In Figure 38 and Figure 39, the activity UML diagrams corresponding to both situations are represented: interaction User – MPEG21 DI Browser and interaction WDI Browser – IDI Browser.

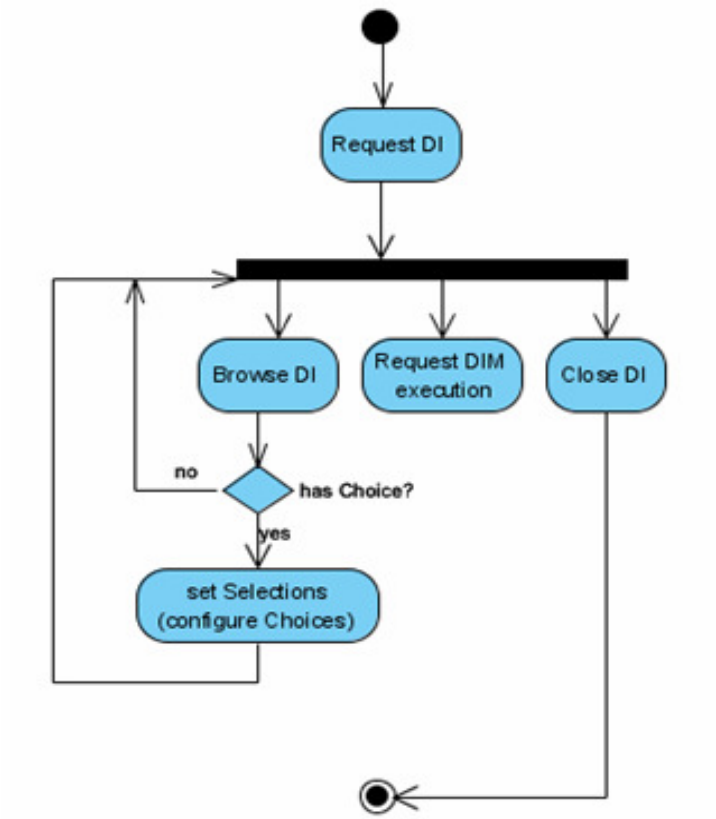



**Figure 38 Activity diagram of User interaction with MPEG21 DI Browser**

Initially, the User provides the location of a certain Digital Item and as response receives the top element containing a list with the available sub-elements. From this point, the User can browse through the Digital Item, can request the execution of DIMs, if it is the case, or can abandon the navigation.

The browsing consists in the navigation through the DID structure of the Digital Item allowing the User to visualize the contained resources and the corresponding descriptions.

During the browsing, the User has to set some *Selections* that allow him to express his preferences, when *Choice* elements are present in the DID representation of the Digital Item.

If the visualized Digital Item contains DIMs, the User may request to execute them and to dynamically interact with the content.

In any moment the User may abandon the browsing by "closing" the Digital Item.

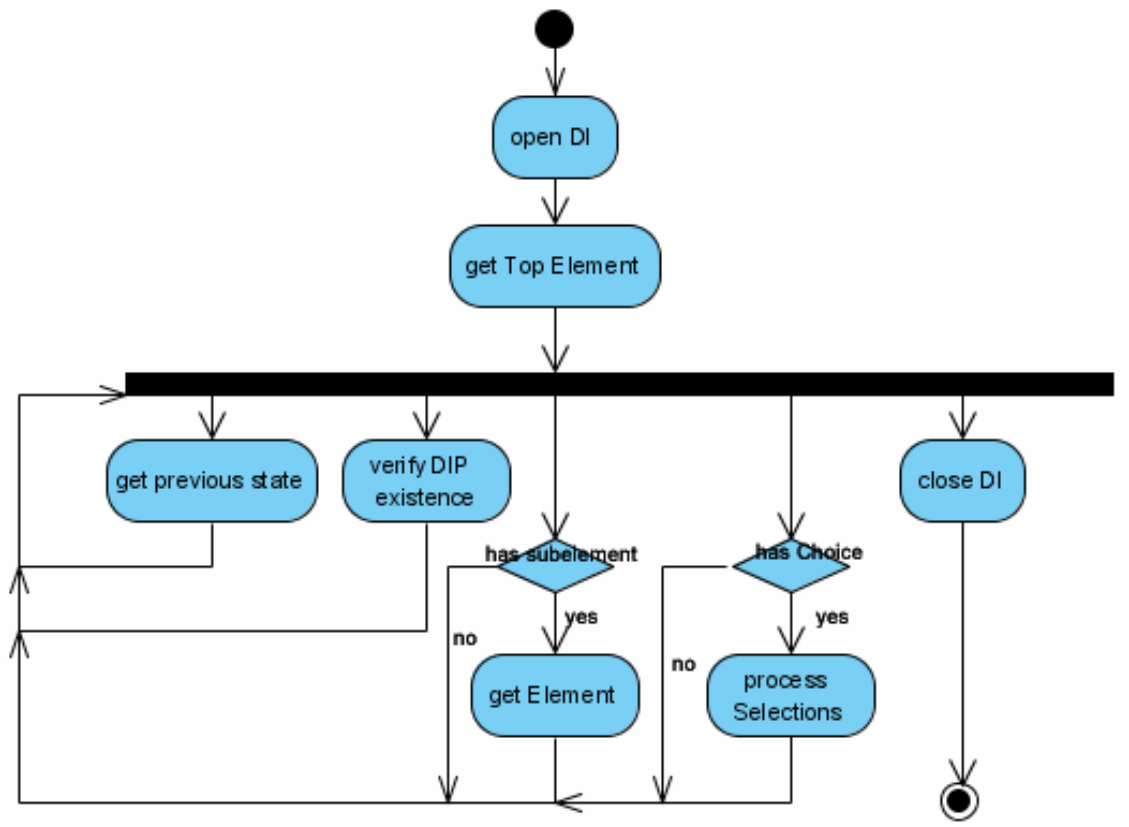



**Figure 39 Activity diagram of WDI Browser interaction with IDI Browser**

In Figure 39, the actions are similar with the use cases defined in 5.4.1 (see Figure 37).

### 5.4.3. Sequence diagram

This type of UML diagrams is frequently used to represent the behavior and the interactions between system entities. For the MPEG21 DI Browser system these entities are: the User terminal, the WDI Browser client application and the IDI Browser server.

The following figure illustrates the sequence of interactions that can exist between the mentioned entities as a consequence of User actions.

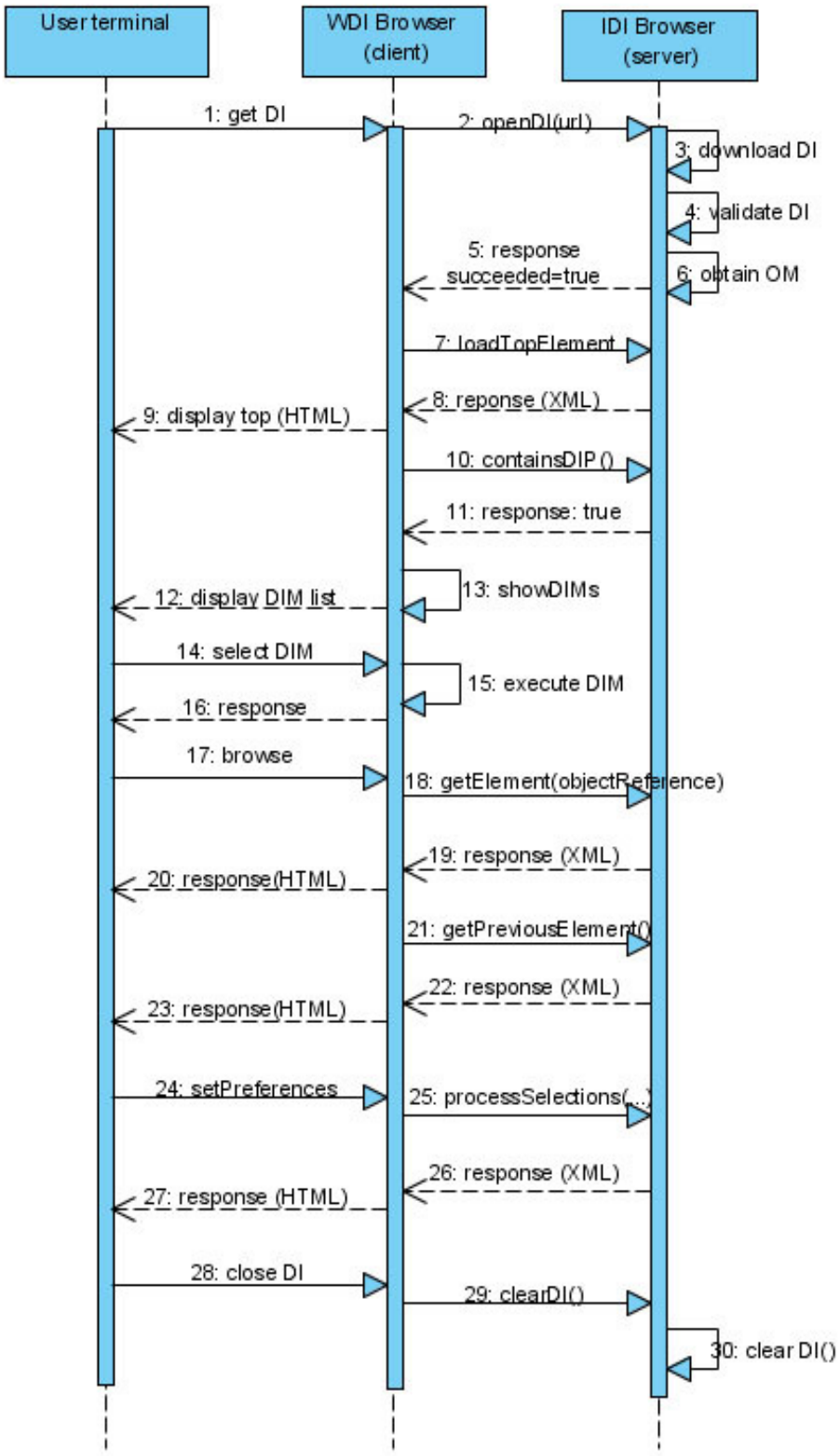



**Figure 40 Sequence diagram for MPEG21 DI Browser**

## 5.5. Usage scenario for the MPEG21 DI Browser system

How can the MPEG21 DI Browser be usefull? To answer this question a usage scenario will be presented next.

Imagine that John Smith, working at “Discover, Travel and Live - Tourism Agency”, is producing a new presentation about Portugal in order to publish it on the agency’s website to promote tourism in this country. While traveling in various regions of Portugal, he has collected many media resources: photos, videos, traditional music, and information about folklore, the Portuguese people way of live, about their culture, and general information useful for tourists. John is using an authoring application for MPEG-21 Digital Items to create his presentation. He combines all the media resources and information in one Digital Item and then uploads it to a specific Web (remote) repository, reporting its location to his agency but also to his family and friends that are curious about Portugal.

The agency’s website has its own client application (e.g., developed with .ASP) for the IDI Browser Web Services (the core of MPEG21 DI Browser system). The agency’s customers can use this client application of the agency’s website to access the respective Digital Item and to view the presentation about Portugal.

At the same time, John’s family and friends may access the WDI Browser for opening and navigating through the Digital Item and for enriching their knowledge about Portugal.

The exemplified usage scenario demonstrates that the MPEG21 DI Browser may be useful for the Users that want to consume Digital Items, no matter if only the IDI Browser server or the whole the system functionality (client and server) are used.

# Chapter 6

## 6. Implementing the MPEG21 DI Browser System

This chapter presents the development of the client-server architecture of the MPEG21 DI Browser according to the basic requirements and the specifications enunciated in Chapter 5. It describes the implementation of the main concepts on which the proposed system is based: the object model for the relevant MPEG-21 parts (DID, DII and DIP); the processing of the Digital Items provided by the IDI Browser server; and the presentation of the information, following the Web philosophy, by the WDI Browser client application.

### 6.1. Object Model

The definition and importance of the object model representation of the MPEG-21 elements were detailed in 5.2.1. The mapping of the MPEG-21 elements into object instantiations and vice-versa is made with a specific tool that needs of a set of rules – normatively defined in XML schema files – in order to reflect the relationships between these elements.

In the case of INESC DIDBrowser (presented in section 4.4), the used tool – JiBX – generates the objects at runtime but it had not the support to map also the mentioned rules. Therefore, the mapping rules were created by the implementors and only then the object model was built (see Figure 18). As MPEG-21 is still evolving and the schema files are changing for some MPEG-21 parts, it is quite difficult to alter the mapping rules and to re-generate the corresponding object model each time a modification occurs. For this reason, another and more flexible object model is needed.

In this thesis, a new object model was created and manipulated using a specific tool from the Apache XMLBeans v.2.2.0 toolkit [62]. It permits at runtime the automatic creation of the mapping rules and the generation of the Java classes corresponding to the MPEG-21 elements, as it is shown in Figure 41. This allows the easy and fast re-creation of the object model when it is necessary.

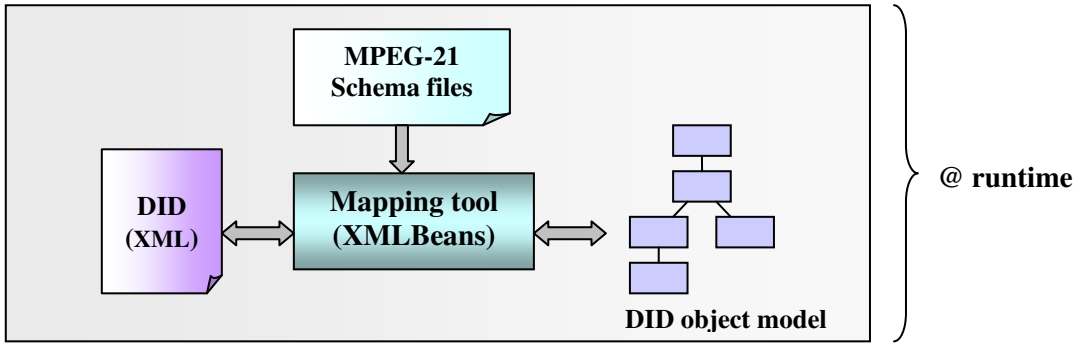



**Figure 41 DID mapping into object model at MPEG21 DI Browser**

Finally, the resulting classes were compiled in a Java library for later use in the system. The Javadoc [63] tool was used to generate the API documentation files corresponding to each class and included also in the obtained library.

The operations mentioned above have to be executed for each schema file or set of schema files corresponding to one single part of MPEG-21. At the end, several Java libraries should be ready and they constitute the sub-components of the Object Model module (this module is part of the system, see Figure 34).

At runtime the IDI Browser uses the API provided by the XMLBeans to map the contents of a new opened DID into one instance of the object model. Once created the instance, its elements can be accessed, interrogated, modified and processed. In the present implementation, when a new "Open Digital Item" command is given to the IDI Browser, a new object model instance is created and the old one is discarded.

The Object Model module is rather more specific to MPEG-21 processing than to IDI Browser; therefore the created Object Model can also be used in other MPEG-21 applications.

The present Object Model includes the implementations for DID, DII and DIP. In future work, the object model for the other parts of MPEG-21 can be created.

## 6.2. IDI Browser server sub-system

The IDI Browser server application is the core of the MPEG-21 DI Browser system. It provides all the processing needed for the system functionalities. Its main role is to centralize the MPEG-21 specific processing and to control all the interactions with Digital Items.

The IDI Browser is structured on modules (see Figure 42), where the lowest-level module is the Object Model containing the generated objects, corresponding to MPEG-21 elements, which was described in the previous section.

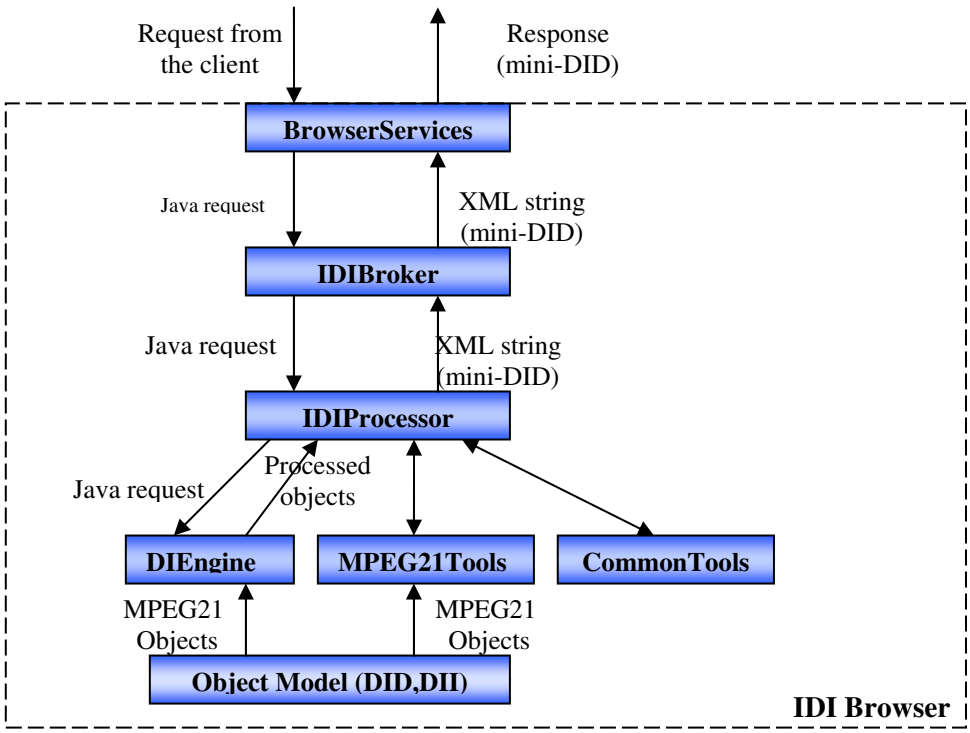



**Figure 42 Architecture of IDI Browser**

Each module of the IDI Browser was developed in Java due to the fact that the implementation of most of the existent MPEG-21 reference software and utility software is in Java, bringing the major advantages of being portable to many platforms and free. The top level module, Browser Services, encapsulates all the libraries corresponding to the other modules and was archived as a .war (Web archive) for later deploying as a Web service in the Tomcat container.

The messages that can be exchanged between the internal modules of the IDI Browser are shown in Figure 42. The top module receives the requests for parts of the Digital Item from the client application and translates the respective messages into Java requests that are passed to the lower level modules, until they meet the IDIProcessor. This last module analyzes the requests and asks the DIEngine to answer them by processing the parts of the Digital Item that are needed to be displayed at the client application. When other operations are necessary, for example downloading a Digital Item, the IDIProcessor will ask the CommonTools to execute them. If it is necessary to validate the DID representation of a Digital Item, then the MPEG21Tools module is responsible for this operation. The Object Module will provide to the DIEngine and MPEG21Tools the object instantiations for the MPEG-21 elements that are processed. When the DIEngine finishes preparing for the browsing the requested parts of the Digital Item, it sends the corresponding processed objects to the IDIProcessor which will transform them into an XML string and will send them to the

upper module. These responses ("mini-DIDs") will be passed up to the BrowserServices and then to the client application.

When considering the structure of the MPEG21 DI Browser system specified in 5.3.2, it is expected that the DIPEngine module, ensuring the DIP functionality, is integrated at the IDI Browser server application. But a problem appears and a question derives from it: how to develop the DIBO [23] operations such as play, print, execute, etc. on the server side and how to send the results to the client application after executing these DIBOs on the IDI Browser? The "step by step" browsing philosophy has to be reflected in the manner of developing the DIP; this means that a new implementation for the functionality of part 10 of MPEG-21 is needed, as the present reference software does not provide the support for the Web philosophy adopted in this thesis. The interfaces for DIBOs are normatively defined but their implementation is up to the developer and therefore a solution for this problem may be found allowing the remote execution of DIBOs. The current version of the MPEG21 DI Browser is not focusing on finding a solution in this sense but rather in exercising the integration of DIP concepts and using their advantages. Therefore the decision was to include in the WDI Browser client the MPEG-21 reference software provided for DIP. A factor that greatly influenced this decision was the insufficient time for developing a complex MPEG-21 Peer system during this dissertation.

The following sub-sections present the implemented modules within the IDI Browser server sub-system of the MPEG21 DI Browser.

#### 1) DIEngine

The DIEngine is responsible for processing Digital Items at the level of the DID Object Model. The processing consists in constructing the simplified DID base elements (*Item* or *Container*) as specified in 5.2.2. This module has a mechanism for caching the original elements before the processing, for caching the processed (simplified) elements and for storing the selections set by User up to the moment. This caching is necessary for providing the previous state in the step by step navigation through the Digital Item (the previous element and the corresponding selections set up to that respective point) in any moment when the User requests to browse backwards.

In each step the DIEngine is responsible for preparing a "clone" of the current visualized element (*Item* or *Container*) and this includes a filtering process where the sub-elements that do not meet the *Conditions* are discarded. The information that is not relevant for Users (e.g., the comments) is also removed. Then, for each sub-element (*Item* or *Container*) only the first textual *Descriptor* is kept, since they are elements to be fully-viewed later on in the next

steps. And finally, some extra information is included, that is MPEG21 DI Browser application specific, such as the generated identifiers for the sub-elements (see the *objectReference* attribute in Figure 27). These identifiers serve for associating the internal elements with the ones requested from the client side upon the User clicks.

#### DIEngine components

The DIEngine module is composed by several classes contributing to the processing of elements (*Items* and/or *Containers*) of the Digital Item. In the following figure the component sub-modules (classes) and the dependence relation between them are presented.

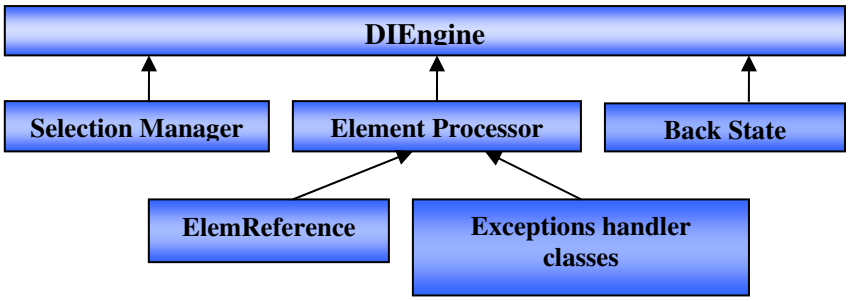



**Figure 43 DIEngine component classes**

Next, a brief description is made for each class within the DIEngine module.

#### Exception handler

The exception classes provide a better handling and fixing of the errors that may occur. They may give additional information that makes the identification and localization of the problems easier to debug.

#### ElemReference

The ElemReference class is a container used to store the association between an element (*Item* or *Container*) and a reference generated for the identification of the element.

#### ElementProcessor

ElementProcessor is the main responsible for the processing of *Items*/*Containers* (base elements) within a DID, at the level of the Object Model. The results from this processing are then provided to the DIEngine class that manages the obtained simplified elements that will be sent to the client application.

The ElementProcessor implements among other methods those for: obtaining the first textual *Descriptor* object of a given element; returning the list of *Conditions* within a specified element; testing a set of *Conditions* of a certain element to see if it is to be presented

or not; filtering by *Conditions* the sub-elements; removing the comments information; and generating an integer reference for a given element.

The essential processing methods with their implemented functionalities of this class are illustrated below.

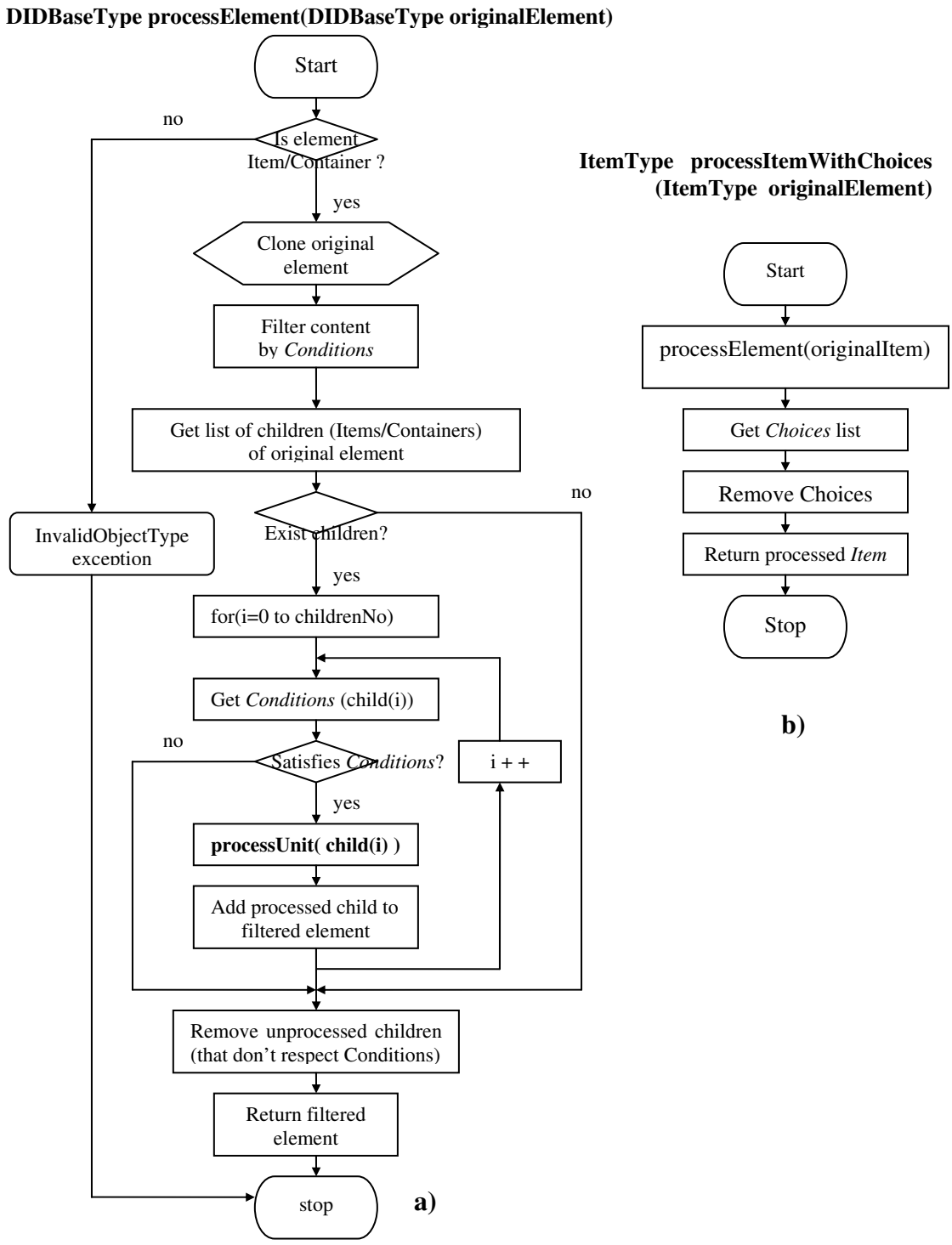



**Figure 44 Methods for processing the element of ElementProcessor class: a) processElement(…) is used for general elements; b) processItemWithChoices(…) is used when some Selections were previously set**

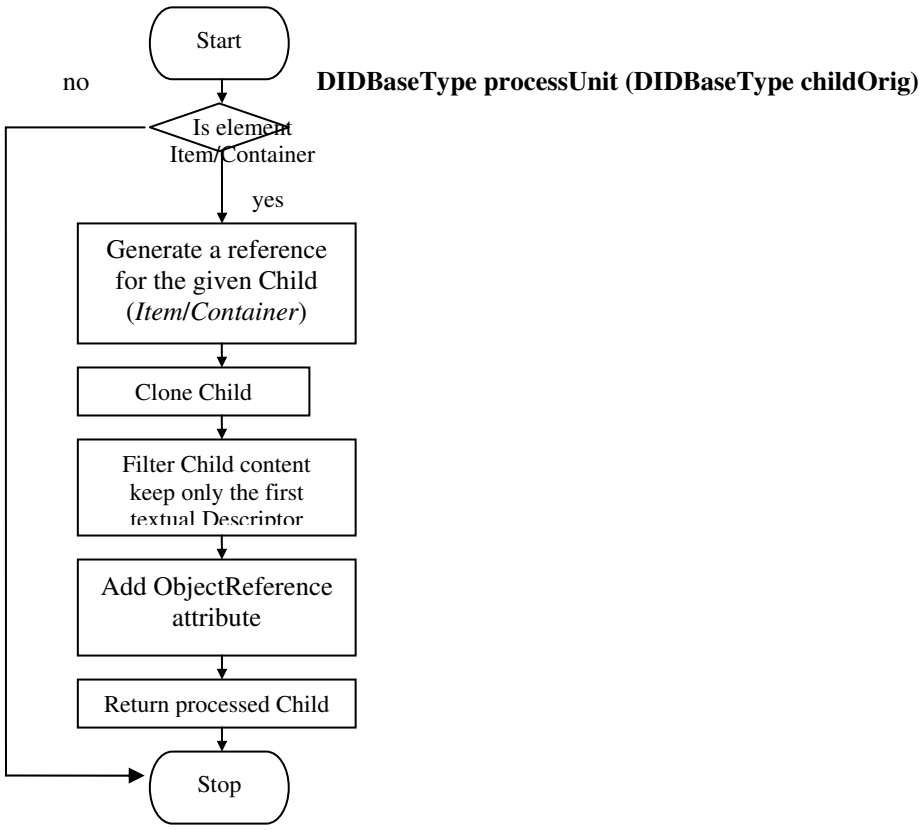



**Figure 45 Method for processing the sub-elements (Items/Containers)**

#### SelectionManager

This class is used for registering *Selections* elements. For each new *Selection*, it stores information such as: the *Selection* identifier, the boolean value representing the state of that respective *Selection* and the identifier of the parent *Choice* element.

The methods of the SelectionsManager class are dedicated for testing the existence of *Selections* and their values, and also for testing the *Conditions* restricting an element; for adding/removing *Selections* values, etc.

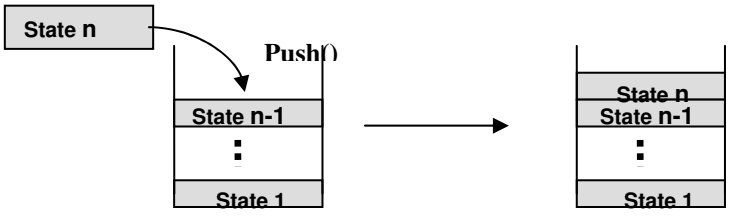
#### State

The DIEngine keeps track of several past states of navigation through the Digital Item at any given moment, and particularly, when the Back command is issued, it restores the previous state. The contained State class implements such a state. A state is composed by the reference to the previous element presented to User and the selections made until that moment.

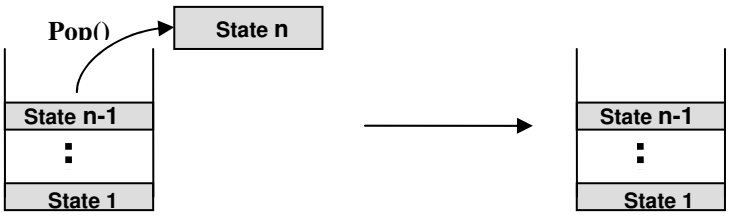
#### DIEngine

The DIEngine class (has the same name as the module) handles the requests for Digital Items, processing parts thereof, in order to ensure a proper navigation and interaction with them. The processed *Item*/*Container* elements are enclosed in DIDL objects and form "mini-DIDs" (defined in 5.3.2) sent to the next upper level module which is the IDIProcessor.

This class also ensures the management of the previous states of navigation for the current Digital Item. It uses a stack object to store the State instantiations containing the associated previous states. The push operation serves for inserting the states instantiations into the stack object and the pop operation for obtaining the last managed state. The stack provides a "natural" mechanism for rolling back to previous states at a given moment, as illustrated in Figure 46.




Adding a new State instantiation




Get the previous state (State instantiation)

**Figure 46 The management of the previous states of navigation**

The main methods implemented here are for: opening the Digital Item by mapping it into the DID Object Model; loading the top element of the DID; getting a certain element; processing the Selections set by User; processing the back command - returning the previous element; verifying if the DID contains DIMs (DIP methods) to be executed; and closing the browsed Digital Item by cleaning up all used temporary data.

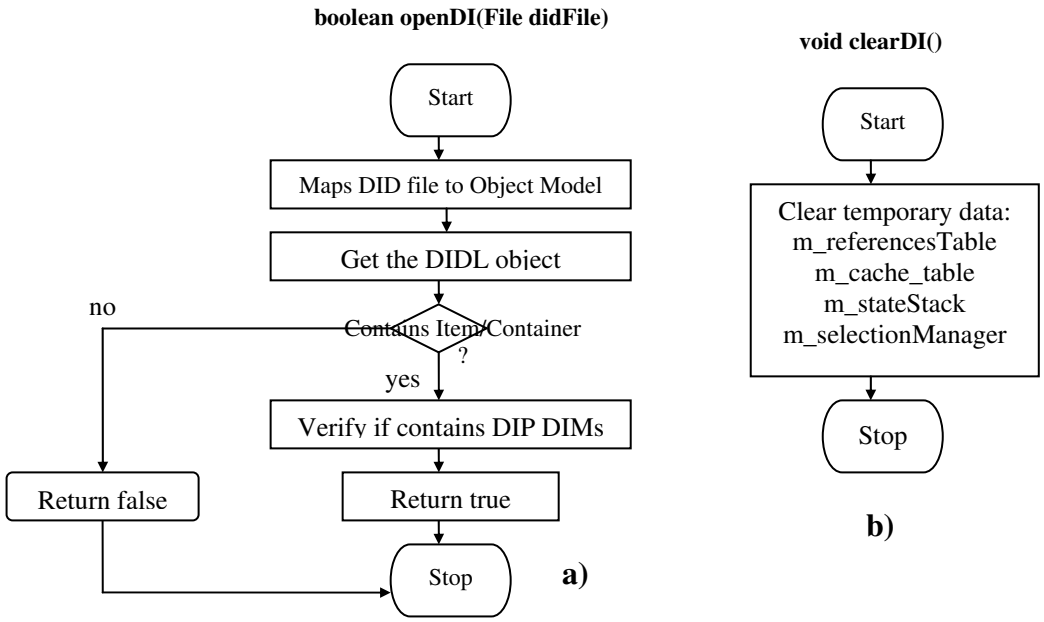



**Figure 47 DIEngine methods for: a) opening the Digital Item; b) clearing the Digital Item**

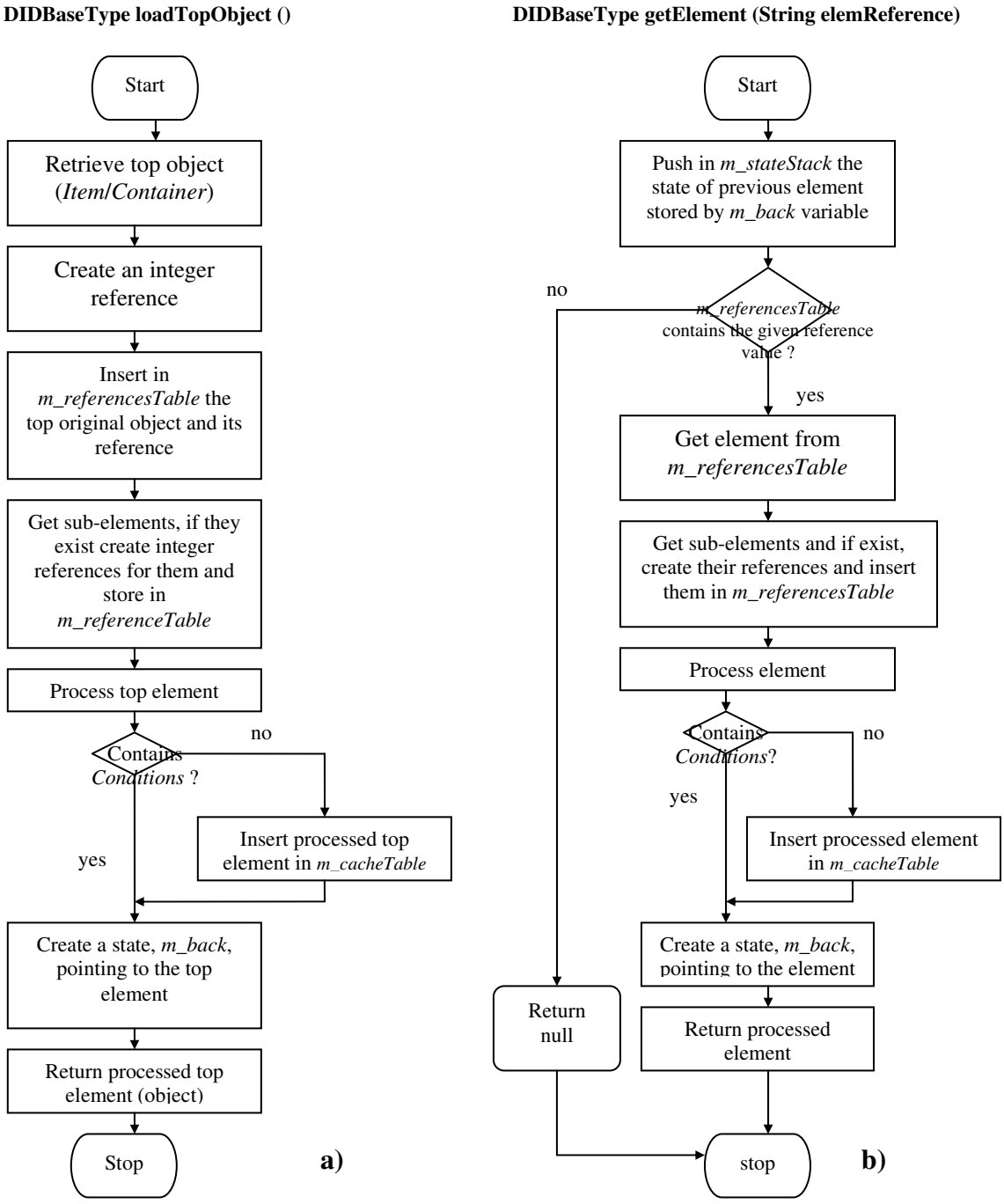



**Figure 48 DIEngine methods for: a) loading top element; b) geting the processed element identified by the given reference value.**

As it can be noticed in Figure 48, the mechanism used for caching and reusing the previous processed elements contributes to a faster response that increases the performance of the IDI Browser sub-system. Therefore the User navigates more rapidly into the requested Digital Item content when using the WDI Browser Web graphical interface.

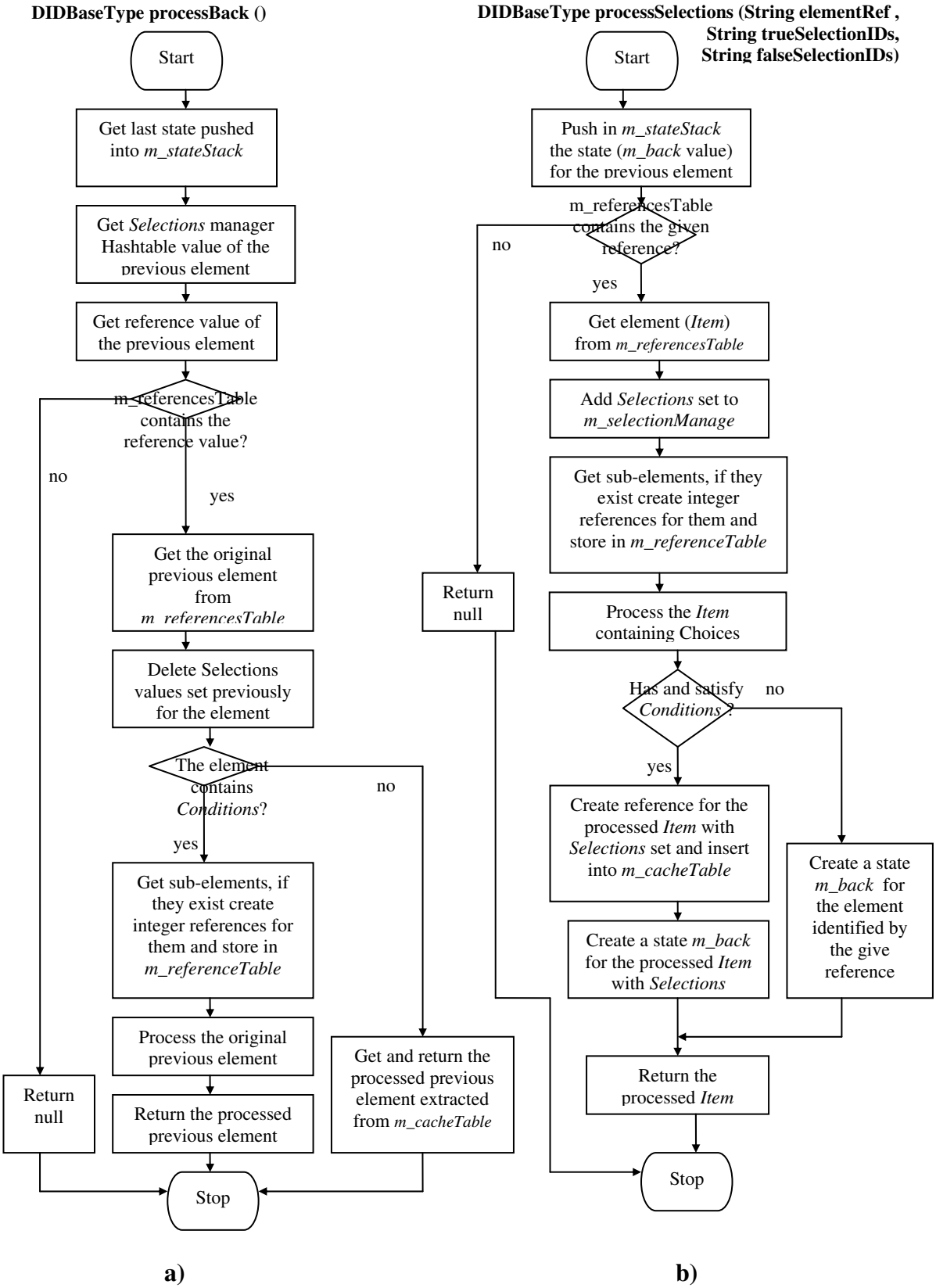



**Figure 49 DIEngine methods for: a) processing the Back command; b) processing the Selections set within the given Item**

### 2) Utility Tools

Utility Tools is the general name given to the MPEG21Tools and CommonTools modules (see Figure 42).

The MPEG21Tools module is intended for developing general MPEG-21 functionalities and that are not specific for the IDI Bowser and for its manner of processing the Digital Items. MPEG21Tools contains a sub-module named XmlUtils that provides among other XML specific processing facilities a class that implement the validation of an XML file by using its schema file.

The CommonTools module is presently composed only of a sub-module named FileReceiver that is responsible for downloading a file from the given location. More exactly, it is used to transfer a remote Digital Item to a local repository on the IDI Browser server application. The supported transfer protocols are HTTP and FTP.

### 3) IDIProcessor

The IDIProcessor module interconnects all the modules implementing the Digital Items processing that is specific for parts of MPEG-21. Currently only the DIEngine is implemented. As future work other modules will be created for processing REL, DIA and ER information from a Digital Item.

This module contains a class with the same name and its public methods are:

- *openDID* – receives as parameter the remote location of the Digital Item and returns a boolean value indicating if the DID file was successfully opened or not; this method downloads the Digital Item from the given location (in the case of a remote file) to a local repository on the Web server machine where the IDI Browser sub-system is installed, then opens and validates the respective DID file, and finally calls the *openDID* methods of the DIEngine for mapping the DID XML file into the corresponding Object Model;
- *loadTopElement* – returns the top element as a specific XML content; it calls the *loadTopObject* method of the DIEngine and receives the top element which is then included in a *DIDL* object and build a “mini-DID”;
- *getElement* – returns a requested element that is identified by the reference value given as parameter; it calls the *getElement* method of the DIEngine;
- *processBack* – moves back to the previous navigation stage and returns an XML string containing the anterior processed element obtained from the *processBack* function of the DIEngine;

- *processSelections* – returns the element that resulted from the processing of the *Selections* made by User within a certain *Item*;
- *clearDID* – calls the DIEngine *clearDID* method;
- *hasDIP* – also calls the corresponding method of the DIEngine and returns a boolean value: true if the DID contains DIP Label information and false if the DID does not contain such a tag;
- and *getDIDLocation* – returns as a string the Digital Item real location (on the local server or the remote path).

#### 4) IDIBroker

The IDIBroker's role is to intermediate the communication between the internal MPEG-21 processing module (i.e., the IDIProcessor) and other modules that may be developed as interfaces with databases storing Digital Items, information about Users and their sessions, about their licenses, etc., or with modules that may process or manage information other than MPEG-21. For the current version of the IDIBrowser sub-system, the IDIBroker module provides only the necessary methods for processing Digital Items while browsing them. These methods call the ones developed in the lower level module IDIProcessor, e.g., *openDID*, *loadTopElement*, *getElement*, *processBack*, etc.

#### 5) BrowserServices

The Web Services technology was used to implement this module due to the flexibility and interoperability across heterogeneous platforms that they offer to the MPEG21 DI Browser system.

Regarding the Web Service operations that the IDI Browser has to provide to the WDI Browser client application (or other client used for Digital Items consumption within the MPEG-21 framework), a wrapper Java class was created on top of the processing modules (also Java classes) in order to provide the Web services. These Web Services call some public methods of the Java classes used in Digital Items and other MPEG-21 processing. The technology that was used to develop the IDI Browser Web Services was the Sun Java Web Service Developer Pack 2.0 (JWSDP 2.0) [57]. The advantages of using JWSDP 2.0 are: it includes the support for the implementation of a distributed system accessible through common Web technologies, and has everything that is necessary for using SOAP and WSDL in connection with Java. It provides a complete environment for developing and publishing Web Services in Java, having implementations of the Java APIs for XML, and some tools and

platforms to easy develop and deploy Web Services, such as the Tomcat 5 [58] and Apache Ant [59].

The BrowserServices module of the IDI Browser implements the Web Services operations needed by the WDI Browser client application that ensures the presentation and User interactions with Digital Items and parts thereof.

The implemented Web services operations for the IDIBrowser are:

- *openDID*
- *loadTopElement*
- *getElement*
- *processBack*
- *processSelections*
- *clearDID*
- *hasDIP*
- and *getDIDLocation*.

The BrowserServices module also contains the ConfigManager class which is responsible for the management of the configuration file used to easy set some of the server parameters. The configuration file provides the location on the Web server machine of the schema files used for validating the DID opened at the IDI Browser, and also for pointing to the location of the configuration file for the log system.

The IDIBrowser Web archive contains all the Java libraries and the compiled internal processing modules that are needed for running the IDI Browser Web services.

## 6.3. WDI Browser client sub-system

The most important role of the Web client is to translate the User's actions, to request Web Services from the server and to send the results to the User's Web browser.

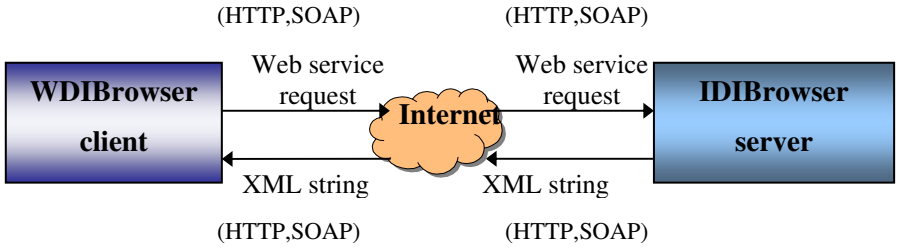



**Figure 50 Communication between MPEG21 DI Browser sub-systems (WDI Browser and IDI Browser)**

The purpose of the WDI Browser sub-system of the MPEG21 DI Browser is to be a demo client for the Web Services provided by the IDI Browser server. Using the results sent

by the server application responsible for Digital Item processing, the WDI Browser presents to the User the content and permits him to interact with it.

The Web client had to be implemented using a technology that supported communication with Web Services via HTTP, with the exchanged messages being encoded as SOAP, and provided a Web graphical interface (HTML pages) to User.

Therefore the WDI Browser client sub-system was created in a website style using PHP 5 [60] and running on the Apache 2 [61] HTTP server. PHP is a well known technology, flexible and currently used for Web development worldwide. It has a high-level API that substantially contributes to saving time on the development phase. PHP was designed for rapidly and stable development of cross-platform Web applications that do not depend on the browser technologies and also allows to easily create the Web Services client. PHP 5 has an extension that can be used to create SOAP servers and clients.

The WDI Browser was created to be user-friendly and flexible, for supporting various Web graphical interfaces adequate to different terminal device. At present only the interface to Web access from PC terminal devices has been implemented.

The modular architecture contributes to the mentioned flexibility. The component modules are organized as shown in Figure 51. The modules called by the top level GUI module are classified by their provided functionality into two types: the ones used for implementing a minimal processing and management of the received parts of the Digital Items (as ElementRender) and other used only for generating the Web interface and for the presentation of Digital Items (as ContentGenerator).

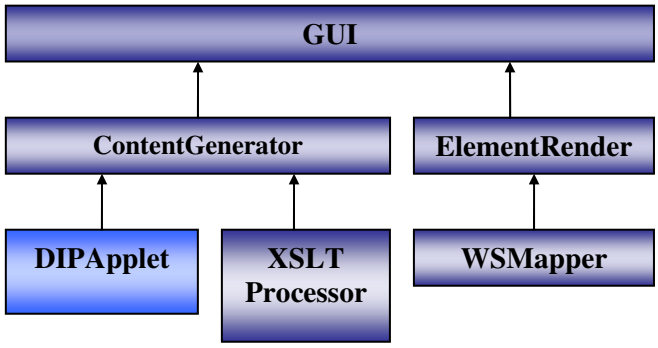



**Figure 51 Architecture of WDI Browser client sub-system**

Excepting the DIPApplet, all the modules were developed in PHP 5. XSLTProcessor, ElementRenderer and WSMapper were developed in an object oriented programming manner. The following sub-sections describe the component modules of the WDI Browser client application.

### 1) WSMapper

WSMapper is the lower module of the WDI Browser that directly accesses the published WSDL description of the IDI Browser's Web services.

This module is based on the WSConnection class building the SOAP client for the Web Services provided by the IDI Browser server. The methods of the available Web Services return the processed elements in an XML string format.

### 2) ElementRender

ElementRender intermediates the Web Services requests between the GUI and the WSMapper by calling the methods of the WSConnection class for obtaining the elements ("mini-DIDs" containing the requested processed *Items* or *Containers*) that will be presented by the GUI module. ElementRenderer also contains other methods for managing XML that are required during the elements presentation. Such methods are for verifying if the current element to be displayed contains sub-element or testing the existence of *Choices* elements in order to create a new browser window for configuring these *Choices*.

### 3) XSLTProcessor

This module is essential in the presentation of Digital Items due to its method which transforms an XML string, received from the IDI Browser, into HTML, displaying only the relevant content in a Web page style.

Stylesheet templates (.xsl files) were created in order to generate the HTML views for the content of the *Descriptors*, *Components* and *Choices* with the corresponding *Selections,* and respectively for building the right menu of the sub-elements hyperlinks (containing references to the available sub-*Items* or sub-*Containers*).

For *Descriptors*, the textual information contained in a *Statement* element of a "text/plain" *mimeType,* and media resources (such as images), that may exist in the *Descriptors* can be visualized. But it may also display, in the future, XML information such as TV-Anytime and/or MPEG-7 descriptions.

The media resources that presently can be displayed are: audio files, images, videos and ordinary text.

When an *Item* contains more *Choices* they are all displayed in a separate browser window, where the *Selections* are represented as HTML radio buttons. After setting the values for *Selections* (Yes or No; where Yes corresponds to true and No to false) and pressing the submit button OK, the *processSelections* Web Service operation is requested.

The first textual descriptions of the available sub-elements are visualized in a Web menu that is placed on the right side of the graphical interface, where each description is a hyperlink containing the reference of the corresponding sub-element. When clicking on the sub-element's description, the respective sub-element is requested from the IDI Browser.

### 4) ContentGenerator

The ContentGenerator module is composed by several .php files which create the HTML views of the essential parts of the WDI Browser interface when showing the *Descriptors* information, displaying the *Resources*, or containing the HTML forms corresponding to the Back and Close buttons, or the *Choices* window, etc. These files use the method provided by the XSLTProcessor for transforming the XML elements into HTML.

### 5) GUI

The GUI is the top level module of the WDI Browser and acts as an interface between the User actions and the Digital Items browsing processing.

It contains several .php files which compose the HTML skeleton to which are added the "views" provided by the ContentGenerator module to build the visual Web aspect of the WDI Browser client application.

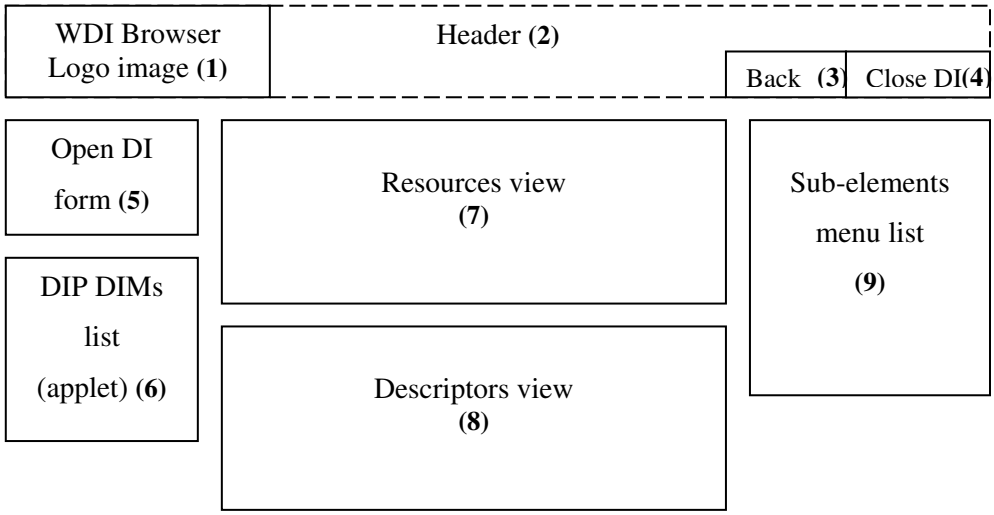



**Figure 52 Layout of the WDI Browser graphic interface (version for PC)**

The skeleton of the graphical interface is composed by the HTML views as illustrated in Figure 52. (1) and (2) are simple images, (3) and (4) are the views for the Back and Close buttons that when pressed initiate the requests for, respectively, *processBack* and *clearDID* Web Service operations. (5) contains the HTML form with an input text element for introducing the location (URL) of the Digital Items and an "Open" button for processing the

request for the *openDID* Web Service operation. The view (6) is an applet providing the MPEG-21 DIP functionality and ensuring the dynamic interactions of the Users with the Digital Items. (7) and (8) display the information and data existent/referenced in/by *Descriptors* and *Component*/*Resources* within the current element. When the visualized element does not contain any *Component*, in the *Resource*'s view, a Flash advertising movie will be shown. The last view, (9), is used to list, as a Web menu, the hyperlinks to the available sub-elements and only the first textual description for each such sub-element is displayed. When a sub-element description is clicked, the request for *getElement* Web Service operation available on IDI Browser server sub-system is initiated.

### 6) DIPApplet

Including the DIP functionality in the MPEG21 DI Browser makes it possible for Users to interact with the Digital Items, allowing them to alter the content at the DID level and to request the execution of methods that add more interactivity to the consumption process.

Although it was decided to implement the DIPEngine at the client side, the problem of how to execute the DIBOs remains because the WDI Browser is also running on a Web server. The DIP reference software developed in Java was conceived to be part of a desktop Peer installed on the User's machine. In order to make it suitable for the WDI Browser, the reference software was transformed into an applet Web application which can easily be included in a Web page. When a Java technology-enabled browser is used to view a page that contains an applet, the binary code of the applet is transferred to the User terminal and executed by the local Java Virtual Machine (JVM). The applet can be visualized on different operating systems supporting JVMs.

The DIPEngine provided by the DIP reference software required minimal changes to be included into the applet. The interfaces have not been altered and they are still in conformance with the standard specifications. Two new methods were created: one for executing the DIM methods selected at the applet's interface and the other to fill the list, of the applet GUI, with the existing DIMs. The last method also executes the DIMs set to autorun; this means that when a DIM, with the *autorun* attribute set to "true", is found, the respective DIM is executed automatically. Immediately after that, the *autorun* is set to "false" to avoid a new execution when the Web page containing the applet is reloaded after a refresh command.

The DIPApplet is the Java class that creates the applet. Its objective is to provide an interface between the User actions (select a DIM, request to execute a DIM, specify the argument objects, etc) and the DIP functionality implemented by the corresponding reference

software. The applet receives two parameters: the URL of the Digital Item location and the autorun value indicating if there exist DIMs to be run automatically.

The view for the DIPApplet is created in the WDI Browser interface only if the requested Digital Item contains DIMs or JDIXOs to be executed by the DIPEngine. This is specified by the DIP Label tag within a DID document [3].

## 6.4. Results

The User can access the MPEG21 DI Browser system by using most Web browsers (e.g., Internet Explorer, Mozilla Firefox, Netscape Navigator, Opera, etc) but will be conditioned by the Internet connection of his terminal device. He will be presented with the WDI Browser Web interface. The User will browse the Digital Items in a manner similar to websites navigation. WDI Browser is a cross-platform application: it was tested under Windows and Linux operating systems.

The next sub-sections describe how the Users can consume the Digital Items through the WDI Browser interface.

### 1) Open Digital Items

The User requests a Digital Item by writing its URL in the corresponding input text field of the “Open DI” view, placed on the left of the WDI Browser interface.


**Figure 53 Requesting a Digital Item by introducing its location**

Immediately after pressing the Open button, the User will see the content of the top element of the Digital Item and may start to browse it and its sub-elements. But at the WDI Browser two steps were transparently done: first - the address was tested to find out if it was correctly given and then the *openDID* Web Service of the IDI Browser was requested; and second – if the open was successfully, the *loadTopElement* service operation was then called.

### 2) Requesting parts (elements) of a Digital Item

An element is requested from the IDI Browser server by selecting its description on the right menu of the WDI Browser, named Content Overview. The right menu is displayed to the User only if the current element contains sub-elements (*Item*/*Containers*) which obey to the existing conditions.

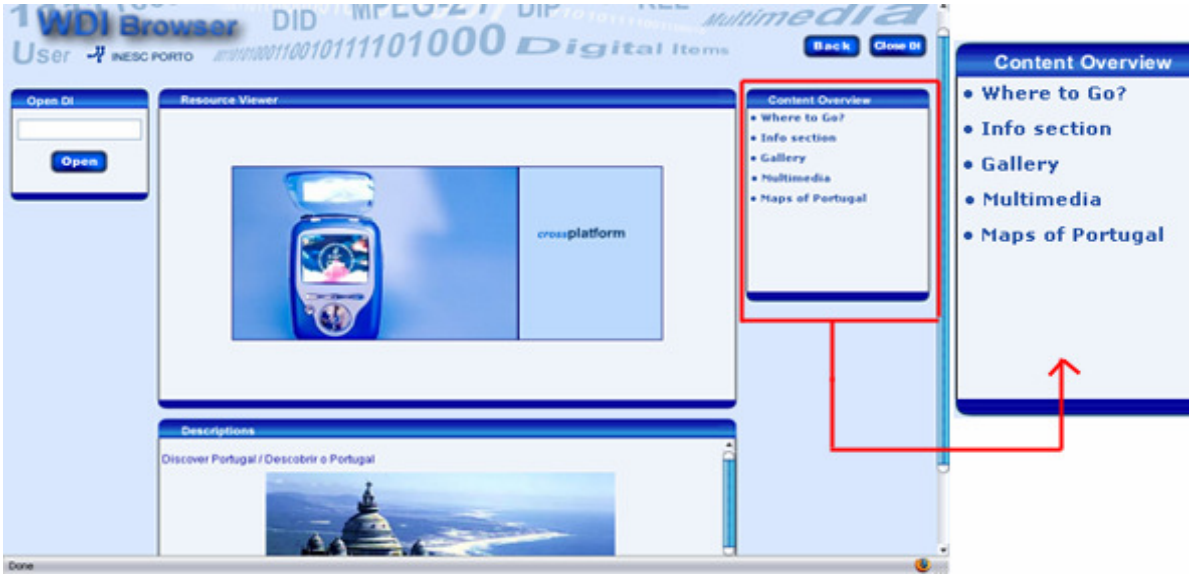



**Figure 54 The menu with available sub-elements, on the right of the WDI Browser interface**

The User can visualize various types of resources (images, video, audio, text) in the corresponding Resource Viewer, where the content of Components elements with their Resources is displayed. When Component elements contain long descriptions, these will be displayed in a shorter version. For viewing the entirely descriptions text, the User has to open a new browser window by clicking the "More about" hyperlink (see Figure 57, on the right). The same is done for the long textual descriptions shown in the Descriptors View.

The video and audio resources are played in the media player available on the User machine (e.g., Windows Media Player, QuickTime Player). The referenced Web pages or HTML text are shown in HTML frames displayed in the Resource Viewer.

When the current element does not contain Components with Resources, a Flash movie for publicity is displayed (see Figure 53 and Figure 54).


**Figure 55 Examples of contents displayed for an element (Item or Container)**

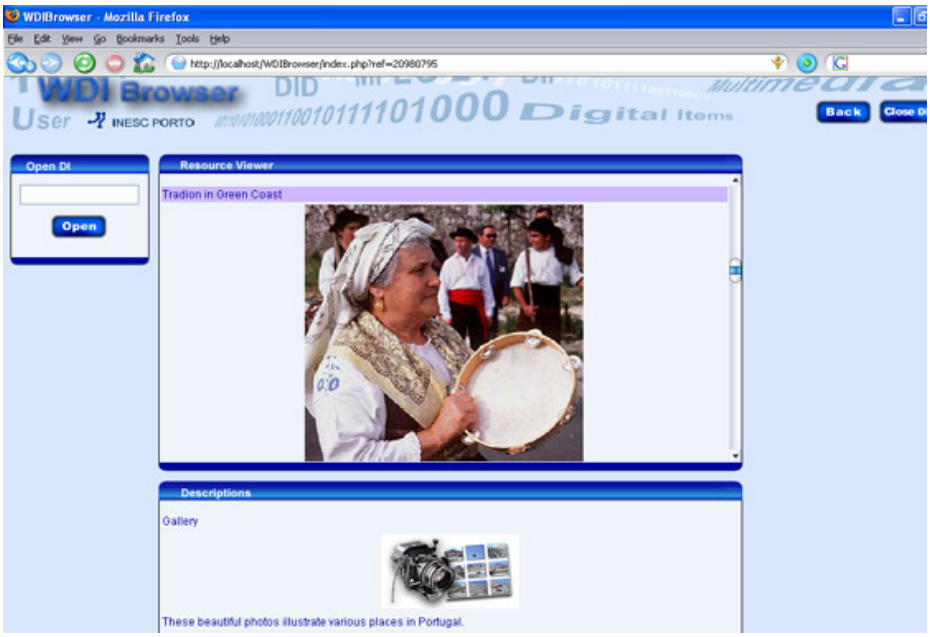



**Figure 56 Example of elements containing images**

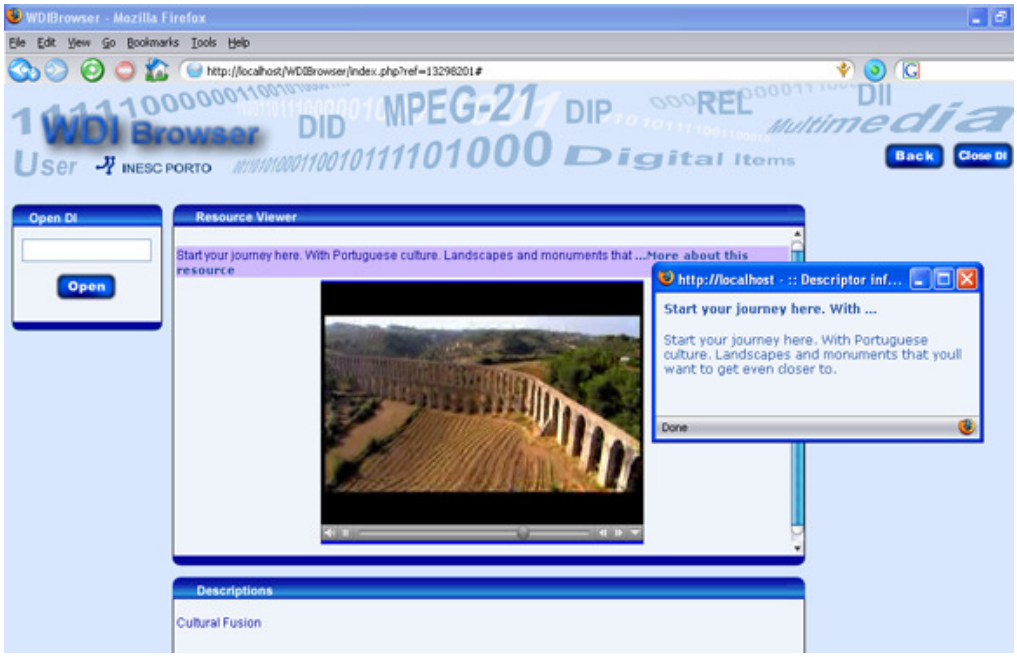



**Figure 57 Example of elements containing video resources**

### 3) Configuring Choices

If the current requested element contains *Choices*, at the moment it is displayed a new browser window appears with the *Selections* to be specified by User for visualizing a specific part of the element. If the Cancel button is pressed, the *Choice* window will disappear and the User will see only the content that is not conditioned by the respective *Selections*. If the User sets the values for some *Selections* and then changes his mind and wants to set them differently, he can press the Reset button and specify the new values. The *Selections* values true and false are labeled with “Yes” and “No” respectively.

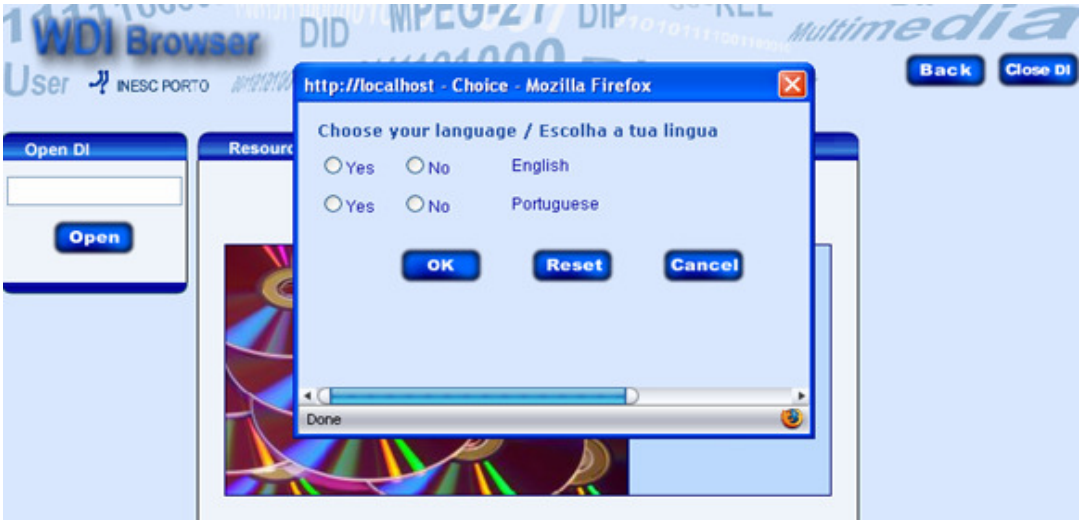



**Figure 58 Example of a Choice window**

After configuring the *Choices,* the User may submit them by pressing the OK button and, immediately after that, the *processSelections* service operation is requested from the IDI Browser server. The received response is the same element that was previously displayed but this time containing also the metadata satisfying the *Conditions* influenced by the *Selections* set.

#### 4) Back button

The Back button placed on the top right contributes to the Web style of navigation in the Digital Item content. When pressed, the *processBack* service operation is called. The result consists in the previous element displayed to User. For a correct browsing it is recommended that the WDI Browser's Back button is used instead of the common Web browser Back command due to the *Selections* and/or *Conditions* that may exist and may restrict the display of parts of the previous element.

If the previous element contains *Choices*, although the *Selections* have been set to specific values, when the previous element is requested, all the *Selections* made in the anterior step will be cleared and the *Conditions* depending on them are no longer satisfied. The *Choice* window will appear again in order to allow Users to perform new selections.

#### 5) Close DI button

The Close DI button is next to the Back button, on the right of the Web interface, and is useful for clearing the Digital Item at the IDI Browser server in order to free it from the data managed during the processing. After pressing the Close DI button the views for *Descriptors*, for Content Overview (sub-element's menu on the right), and the resources are cleared, and only the Open DI and the Resource Viewer will remain visible but without any content (see Figure 53).

#### 6) Dynamical interaction User - Digital Item

The dynamic interaction between Users and Digital Items is ensured by the DIPApplet as it was said in the previous section. If the browsed Digital Item contains DIP DIM methods that may be executed, the DIPApplet will be loaded to the User's terminal.

A list with the available DIMs is presented to the User (see Figure 59), from which he may select one or more for execution.

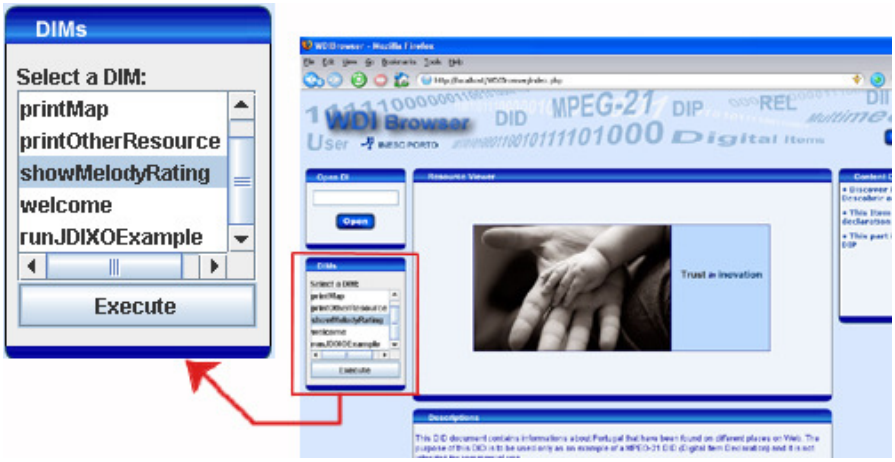



**Figure 59 Displaying DIMs list**

After pressing the Execute button, a list with the arguments (MPEG-21 objects) applicable to the selected DIM (e.g. *showMelodyRating* in Figure 59) will appear, as illustrated in Figure 60.

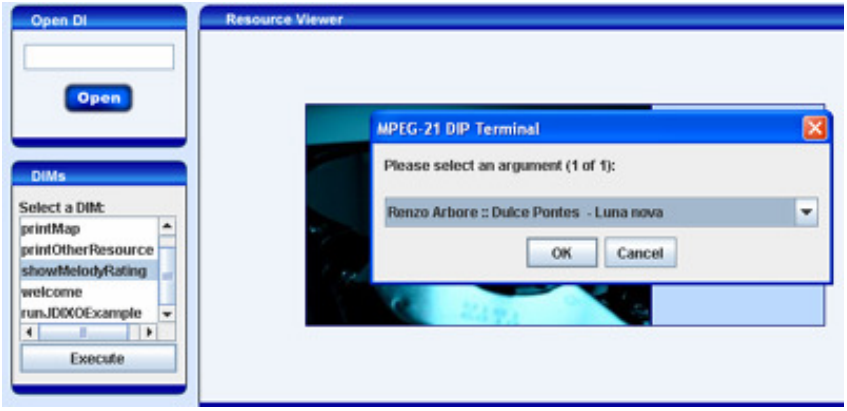



**Figure 60 Selecting the arguments for DIMs**

The User may choose one or more arguments from the selection list and, after pressing OK, the DIM is executed. For example, in the case of executing the *welcome* DIM (which has no arguments but containing a DIBO alert operation with the message "Hello and welcome to Portugal DID!"), a popup window is generated by the DIPApplet and displays the message, as it can be seen in Figure 61.


**Figure 61 The result of executing the welcome DIM**

# Chapter 7

## 7. Tests and System Evaluation

Chapter 7 presents the tests that were made for the evaluation of the developed system. The response time for the processing and presentation of the information was analyzed in the next sections.

### 7.1. Processing Digital Items

In order to test and evaluate the MPEG21 DI Browser system's performance it is more relevant to study the part ensuring the processing of Digital Items, because the time until the presentation on the WDI Browser mostly depends on it. Due to the "step by step" manner of presenting the Digital Items, it is important to test the response time of the methods of the IDI Browser server application that implement the necessary processing of the displayed elements. The processing methods to be tested were: *openDID*, *loadTopObject*, *getElement* and *processSelections*.

In order to accomplish these tests, a set of Digital Items samples were created. A subset of these samples is based on simple *Items* containing only *Descriptors* and *Components* with *Resources*, while the other subset is based on more complex *Items,* having *Choices*, *Conditions*, *Components* with *Resources*, *Annotations* and other metadata that is not DID specific (e.g., TV-Anytime).

Within each subset, the number of *Items* was considered another measure for the Digital Items complexity. For analyzing the system behavior, in the two subsets of Digital Items samples, the number of contained *Items* of the associated DIDs was varied by multiplying one single *Item* element. From IDI Browser application's point of view, all these *Items* are seen as different entities although they are identical. Consequently no caching mechanism was applied during their processing and this approach did not affect the results of the tests. Further more, by increasing the number of *Items* using the same element, a coherence of the tests would be achieved.

The tests were made on a PC with 512 MB of RAM and a Pentium 4 CPU at 3 GHz. The WDI Browser and IDI Browser applications were running on the same mentioned PC.

The results of the tests are included in Table 1 and Table 2.

**Table 1 Response time for processing Digital Items based on simple Items**

| **No. of Items** | **Time for *openDID*** (msecs) | **Time for loading top** (msecs) | **Time for *getElement*** (msecs) |
|---|---|---|---|
| **1** | 14 | 1 | 0 |
| **50** | 19 | 20 | 0 |
| **100** | 21 | 41 | 0 |
| **150** | 31 | 82 | 0 |
| **200** | 38 | 128 | 0 |
| **250** | 46 | 180 | 0 |
| **300** | 60 | 225 | 0 |
| **350** | 62 | 265 | 0 |
| **400** | 77 | 320 | 0 |
| **450** | 89 | 400 | 0 |
| **500** | 93 | 475 | 0 |
| **1000** | 101 | 756 | 0 |

**Table 2 Response time for processing Digital Items based on complex Items**

| **No. of Items** | **Time for *openDI*** (msecs) | **Time for loading top** (msecs) | **Time for *getElement*** (msecs) | **Time for *processSelections*** (msecs) |
|---|---|---|---|---|
| **1** | 20 | 2 | 1 | 1 |
| **50** | 22 | 23 | 1 | 1 |
| **100** | 39 | 57 | 1 | 1 |
| **150** | 67 | 91 | 1 | 1 |
| **200** | 85 | 130 | 1 | 1 |
| **250** | 117 | 183 | 1 | 1 |
| **300** | 128 | 285 | 1 | 1 |
| **350** | 158 | 353 | 1 | 1 |
| **400** | 176 | 431 | 1 | 1 |
| **450** | 234 | 520 | 1 | 1 |
| **500** | 409 | 615 | 1 | 1 |

| | | | | |
|---|---|---|---|---|
| **1000** | 605 | 2200 | 1 | 1 |

The results obtained for the *openDID* and *loadTopObject* methods are depicted in the charts of Figure 62 and Figure 63. In these charts it can be noticed that the response time for processing a Digital Item and its parts depends on the number of contained *Items*. When testing the samples based on simple *Items*, the *processSelections* method was not analyzed because the *Items* do not contain *Choices* elements.

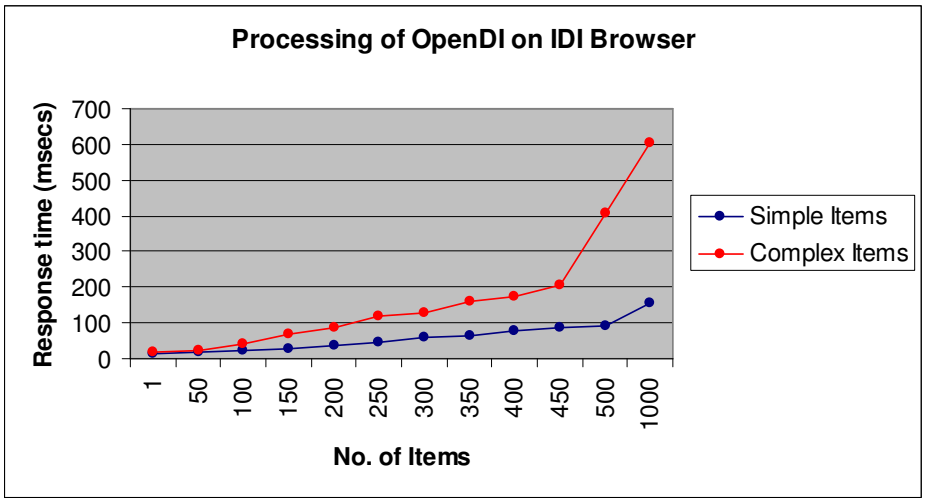



**Figure 62 Variation of time for loading the Digital Item into DID objects and validating it vs the number and complexity of the Items**

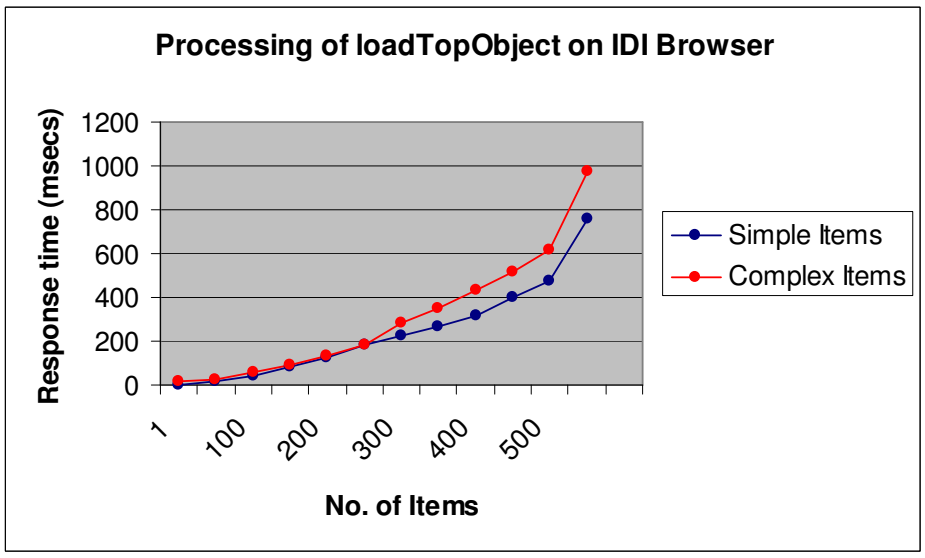



**Figure 63 Variation of time for loading the top element of the Digital Item vs the number and complexity of the Items**

The conclusions that can be drawn from the previous figures are related with the influence of the number of *Items* and their complexity on the response time for processing those Digital Items.

As it can be seen, when the number of *Items* increases, the processing time is affected. When opening a Digital Item on the IDI Browser, the validation of the DID file, representing the Digital Item, and its mapping into the object model require more processing time for more *Items* (see Figure 62). Similar results were obtained when loading the top element. As the Items were identical the necessary time for executing the *getElement* method is always the same and it is nearly nil for simple *Items,* and of 1 millisecond for more complex *Items*. For the same reason the response time for *processSelections* has the value of 1 millisecond for all the Digital Items samples based on complex *Items*.

When comparing the results as a function of the *Items* complexity, the conclusion is that more time is needed for processing complex *Items* containing various type of metadata. The complex *Items* need a consistent processing in order to filter them due to the content that do not respect some conditions and the unnecessary information, and to obtain more simplified elements to be sent to the WDI Browser.

The complexity of a Digital Item is given not only by the number of *Items* but also by the complexity of the *Items*. The structure and content of a Digital Item affect the response time of the processing.

Still the values for the processing time are about a few milliseconds, except when loading the top element of a Digital Item having a number of 1000 of complex *Items* (the response time in this case is almost of 2 seconds), and they seem more than reasonable also for such large and complex Digital Items.

The response time of the *processBack* method was not illustrated here because the values are closed to 0 for all the cases, due to the caching mechanism of the previous processed elements obtained in the anterior steps of navigation.

Due to the way of processing Digital Items by pieces at the IDI Browser level, after they are transformed into DID objects, the response time is considerable reduced. If the Digital Items were browsed as a whole at once, it would be more difficult to obtain the same values for the response time as the ones reached by the current version of IDI Browser.

## 7.2. Presenting Digital Items

The presentation of Digital Items is generated using XSLT Stylesheets for creating the HTML views of the content. The time necessary for generating the presentation is negligible. An exception could exist for thin devices (PDAs, mobile phones, tablet PC, etc), where the presentation time may depend on the terminal capabilities which are poorer than for a common PC.

As the visualization of Digital Items is made on the WDI Browser client application, the time until the content is displayed to the User depends not only on the time needed for processing it on the IDI Browser but is also affected by the transmission time of the information, and therefore it depends on the Internet connection type and on the network capacity. The best response times during the browsing are obtained with a good network connection between the User's terminal and the Web server machine where the WDI Browser is installed, but also between the Web servers running the WDI Browser and the IDI Browser sub-system. In favorable conditions, the total time, from the User's request up to the presentation of the received elements on the client sub-system, may have very good values.

In the particular case of having the WDI Browser and the IDI Browser running on the same machine, as was the case when the above tests for the processing response time were carried out, the total time until the first element (the top) is loaded is almost nil. The exception occurred when more than 500 *Items* are handled; the total time in the case of the simple elements was 2 seconds and for the complex ones was 3 seconds. When testing a Digital Item composed by 1000 simple *Items,* the total time (processing and presentation) was 4.5 seconds, while for the sample with 1000 complex *Items* the total time was of almost 5.6 seconds. These values are satisfactory considering the huge amount of information contained in the respective Digital Items.

## 7.3. Remarks

The above tests have demonstrated that the MPEG21 DI Browser has the capacity of browsing simple and large Digital Items, ensuring a fast navigation between the elements (*Items* or *Containers*). The MPEG21 DI Browser can be considered as having a good performance when consuming Digital Items.

In the last decade the Internet has gained more and more popularity and networks are still evolving in a promising direction. In the near future the problem of the transmissions delays of the data through the networks will be minimized and this will facilitate the use of complex Web applications. Also the MPEG21 DI Browser systems will become more rapid and will ensure that the loading and browsing of Digital Items is made in real time.

The technologies for mobile devices (PDAs, mobile phones, etc.) are also evolving. Soon the presentation time of the complex information on such devices will be reduced and they will use all the advantages that systems such as the MPEG21 DI Browser offer when consuming Digital Items.

# Chapter 8

## 8. Conclusions and Future Work

This chapter concludes this dissertation with some final remarks about the MPEG-21 framework and also briefly comments the work developed and other features that may be implemented in the future.

### 8.1. MPEG-21

MPEG-21 is one of the newest of a series of standards produced by the Moving Picture Experts Group and is coming into life in a moment of fast evolution of the Internet infrastructure and of digital multimedia communications. At this moment several questions are being asked regarding the manner of delivering multimedia content over heterogeneous networks and to various terminal devices such as how to support the desirable universal and transparent access. Most important aspects are about how to represent and reference the content transmitted on the delivery channel, how to treat the consumers and producers in order to put an end to illicit file sharing, how to adapt the content to the terminal environments, how to interact with the multimedia content, how to get more control of the actions over this content, how to stream the media resources and what formats are more appropriate for the content in order to achieve these aspects.

The MPEG-21 framework intends to answers all these questions. Its goal is to provide a common language to the multimedia community for the consumption of the contents. The Digital Items were introduced for representing in a structured and generic form the content and for acting as the fundamental units of distribution and transaction. The DIDs build the structure of Digital Items. The REL provides the means of sharing digital rights/permissions/restrictions for digital content from content creator to content consumer in a "ubiquitous, unambiguous and secure" manner. DIA provides the tools for presenting the User with the most adequate multimedia content, depending on the terminal, network, User preferences and the natural environment where it will be consumed. DIP enables Users to

specify how Digital Items should be rendered or processed assuring a dynamic interaction with the content. ER helps to monitor the usage of Digital Items and to define the reportable events to be specified, detected and acted upon. MPEG-21 standard also defines a file format for the storage and distribution of Digital Items. Recently, MPEG-21 has specified the tools for the streaming of Digital Items. Each part of MPEG-21 is independent of the others and it may be used as stand-alone, as it was not rigorously defined the way of using or integrating them, and this demonstrates flexibility of the multimedia framework. Due to the complexity of the MPEG-21 specifications, it is difficult for the multimedia content industry to adopt the whole standard. Therefore the MPEG-21 implementations are restricted to particular usages and used for specific fields of application, depending on the producer's needs.

The MPEG-21 standard still does not specify the means for using and integrating various technologies for developing the applications of the multimedia framework. This allows flexibility and freedom for the implementors, but at the same time, incompatibilities may occur between the applications provided by different developers (producers), reducing the interoperability. How can MPEG-21 solve this problem and will it succeed in defining a "common language" for the "communication" between all the applications of the MPEG-21 multimedia framework?

## 8.2. The MPEG21 DI Browser system

The MPEG21 DI Browser, specified and developed for this thesis, consists of a MPEG-21 Peer used for the consumption of Digital Items. It can be considered as being a "player" of Digital Items, where the *Item* and *Container* elements are the "played" content.

The goal was to implement a distributed and flexible Web architecture accessible from any device, enabling interoperability and universal and transparent access over a variety of networks. The proposed system has a client-server architecture based on Web Services. The server, named IDI Browser, represents the central point of the MPEG21 DI Browser and it concentrates all the processing, controlling the Digital Items consumption for each type of client application that is able to communicate with the Web Service server. The WDI Browser is the demo client of the developed system and currently provides the Web interface only for PC Users.

The WDI Browser sub-system provides a friendly User Web interface due to the intuitive manner of navigating between elements, for its accessibility by the majority of Web browsers and for allowing Users to interact, statically and dynamically, with the Digital Items.

An advantage of the WDI Browser is that it can display any type of Digital Items: a music album, a container of learning objects, a multimedia presentation, a digital library, etc, while other client applications may be used only for a specific type of Digital Item.

The MPEG21 DI Browser enables the navigation and processing of Digital Items constructed in conformance with the MPEG-21 specifications, independently of the restrictions imposed by its producer.

The current version of the implemented system provides a relevant insight into the concepts and complexity of the MPEG-21 technology, especially for the parts 2 – DID and 10 – DIP. The MPEG21 DI Browser is for the moment a "player" responsible for the browsing and interaction with Digital Items.

## 8.3. Future Work

Future work on the MPEG21 DI Browser is related with the improvement of the system with additional functionalities for increasing its usability, with the development of new MPEG-21 features and with other aspects covered by the extended requirements enunciated in Chapter 5.

### 1) Improvement of DIP implementation

For the current version of the system a temporary solution was chosen for the implementation of the DIP functionality based on the associated reference software. More exactly, the DIP reference software, provided by MPEG-21, was integrated at the client side as an applet application. In the reference software the processing for this part is done for the entire Digital Item and this is not fully compatible with the implemented system when considering the philosophy of "step by step" navigation through *Items* and *Containers*. Therefore, the most important future work is to pass this functionality to the server side and to alter the code of the reference software to work in conformance with the new and particular processing philosophy. In this way all the specific MPEG-21 processing will be centralized on the same server and great consistency will be conferred to the system.

The current reference software does not provide all the necessary support for executing the external operations (DIXOs, [23]) and this can also be handled in future implementation.

### 2) Increasing the usability

It is important to develop the mechanism that enables the Digital Items browsing on different devices, particularly mobile ones. The WDI Browser client should be completed

with a new module for generating the Web interface for mobile devices (e.g., PDAs). The system should have a mechanism for detecting the type of terminal device and for deciding then the proper and more suitable graphical interface for it. An alternative can be to include the HTML information in the *Descriptor*/*Statement* of the Digital Items, which will be used for defining the graphical interface.

The results of this thesis will be integrated in the second phase of the ENTHRONE project. Content search is another feature to be developed for the next prototype of the MPEG21 DI Browser. The search for Digital Items and parts thereof will take into account the User preferences, the terminal capabilities and networks characteristics, in conformance with the requirements defined in ENTHRONE.

Another aspect to be treated will be the browsing of Digital Items located in Users terminals.

### 3) Additional MPEG-21 features

The functionalities for other parts of MPEG-21 will be implemented and constantly updated in order to be conformant with the evolution of the standard.

The REL part of MPEG-21 is the digital rights language for trusted content and services. The MPEG21 DI Browser can be enriched with new functionalities for processing REL, for example purchasing licenses for contents and trading with them. A license can be associated with an entire DID and/or with parts of it.

The adaptation of Digital Items and subsequently of media resources is other essential aspect to be implemented with the aid of the DIA description tools defined in part 7 of the standard.

### 4) Integration of external modules and tools

HTTP is a stateless protocol, i.e. Web servers respond to client requests without linking them to each other. Applying a state mechanism scheme allows a user's multiple requests to be associated with each other across a session. Being able to separate and recognize users' actions as specific sessions is critical for security. The information about Users and their sessions should be stored in a specific database.

The Digital Items are presently stored in a folder repository on a Web server. In future implementation, the repository containing the Digital Items could be a database.

New modules are required to create the interface for accessing, managing and querying the information from databases in order to provide a flexible and transparent connection with them.

Due to the modularized structure of IDI Browser server application, the MPEG21 DI Browser system has great flexibility and allows the integration of other external modules or tools in order to add new functionalities.

## Annex A – Example of DID (Informative)

This annex contains an example of a Digital Item and shows how it can be represented as a DID which is in fact an XML representation.

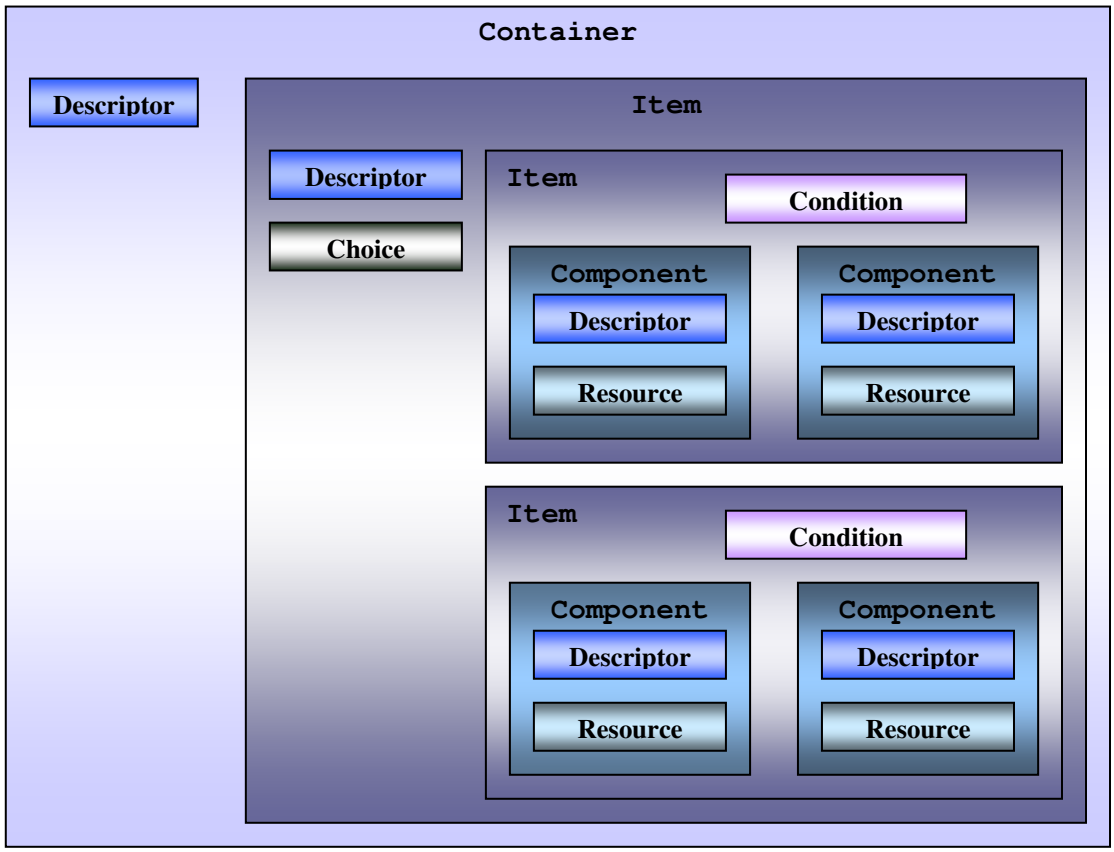



**Figure 64 Example of Digital Item**

Below is presented the XML structure of the DID based on the Digital Item illustrated in the figure above.

```
<?xml version="1.0" encoding="UTF-8"?>
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
            xmlns:didmodel="urn:mpeg:mpeg21:2002:02-DIDMODEL-NS"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<Container>
 <Descriptor>
  <Statement mimeType="text/plain">The Engeneering Faculty of Porto
University
   </Statement>
 </Descriptor>
 <Item>
```

```
    <Descriptor>
      <Statement mimeType="text/plain">Photo Album of FEUP</Statement>
    </Descriptor>
    <Choice choice_id="CHOICE_2" minSelections="1" maxSelections="2">
       <Descriptor>
            <Statement mimeType="text/plain">You can choose to view in the
small and/or in the large format.</Statement>
      </Descriptor>
      <Selection select_id="MINI_PHOTO">
            <Descriptor id="MINI_PHOTO_DESC">
               <Statement mimeType="text/plain">Mini Photo Album of FEUP
(in small format).</Statement>
            </Descriptor>
      </Selection>
      <Selection select_id="LARGE_PHOTO">
            <Descriptor id="LARGE_PHOTO_DESC">
               <Statement mimeType="text/plain">Photo Album of FEUP (in
large format).</Statement>
            </Descriptor>
      </Selection>
    </Choice>
    <Item id="MINI_ALBUM">
        <Condition require="MINI_PHOTO"/>
        <Component>
            <Descriptor>
                  <Statement mimeType="text/plain">Buildings of the
Departaments II</Statement>
             </Descriptor>
            <Resource mimeType="image/jpeg" ref="http://www.fe.up.pt/si/
album_geral.fotografia?f=Buildins.jpg"/>
        </Component>
        <Component>
            <Descriptor>
                  <Statement mimeType="text/plain">Inside of Metalurgy
Building</Statement>
            </Descriptor>
            <Resource mimeType="image/jpeg" ref="http://www.fe.up.pt/si/
album_geral.fotografia?f=Interior.jpg"/>
        </Component>
      </Item>
      <Item id="MAX_ALBUM">
          <Condition require="LARGE_PHOTO"/>
          <Component>
```

```
            <Descriptor>
                  <Statement mimeType="text/plain">South side of the
Departaments, Photo taken by: DEMM, 2005</Statement>
            </Descriptor>
            <Resource mimeType="image/jpeg" ref="http://www.fe.up.pt/si/
album_geral.fotografia?f=Buildins_big.jpg"/>
         </Component>
         <Component>
            <Descriptor>
                  <Statement mimeType="text/plain">Inside of Metalurgy
Building, Photo taken by: DEMM,2005</Statement>
            </Descriptor>
            <Resource mimeType="image/jpeg" ref="http://www.fe.up.pt/si/
album_geral.fotografia?f=Interior_big.jpg"/>
         </Component>
       </Item>
     </Item>
  </Container>
</DIDL>
```

## Annex B – Rights Expression Language

For understanding the concepts of this normative part, the means for the following terms are given:

- A *Grant* is an XML structure that expresses an assertion that some Principal may exercise some Right against some *Resource*, subject, possibly, to some *Condition*.
- *GrantGroups* provides additional expressive power not otherwise found in Grants.
- *Condition* – indicates a condition to be satisfied with respect to a given authorization request and authorization story.
- *License Rights Expression* – expression that is created by Principals to Conditionally or Unconditionally permit the same or other Principals to have Rights upon Resources. Conceptually, a License is a container of Grants or GrantGroups, each one of which conveys to a particular Principal to exercise some identified Rights on an identified Resource, possibly needing some Condition to be fulfilled. A License may be issued by a party which authorizes certain Grants or GrantGroups.
- *Principal* – an encapsulation of the identification of an entity involved in the granting or exercising of rights; more exactly, it represent the "subject" that is permitted to carry out the action involved in exercising the Right.
- *Resource* represents the object to which a Principal may be granted rights. A resource can be an e-book, an audio or video file, an image, an email service, or a piece of information (name, email address) that can be owned by a Principal.
- *Right* – means a privilege that a person may claim, which makes her entitled to create copies of, distribute, or perform all or parts of a published or recorded work for a certain period of time. A Right specifies an action or a set of actions that a Principal may perform on or using the associated *Resource*, under some *Condition*.

This part of MPEG-21 defines an authorization model to specify whether the semantics of a set of *Rights Expressions* permit a given *Principal* (can be a user) to perform a given

*Right* upon a given optional *Resource* during a given time interval based on a given authorization context and a given trust root.

The authorization model makes use of an authorization request, an authorization context, an authorization story, and an authorizer. These are used to create authorization proofs which help to define the semantics of *Licenses* with respect to authorization requests.

An authorization request contains the following members:

- a optionally member that identifies the *Principal* for which permission is requested;
- a second member identifying the *Right* the performance of which is requested to be permitted;
- a third member identifying the *Resource* upon which permission is requested;
- a member that indicates the interval in which the requested performance is to occur;
- an authorization context with statements that are to be considered true when establishing the requested permission;
- a set of *License* elements that may be consulted;
- and a set of *Grant* elements that do not require an authorizer.

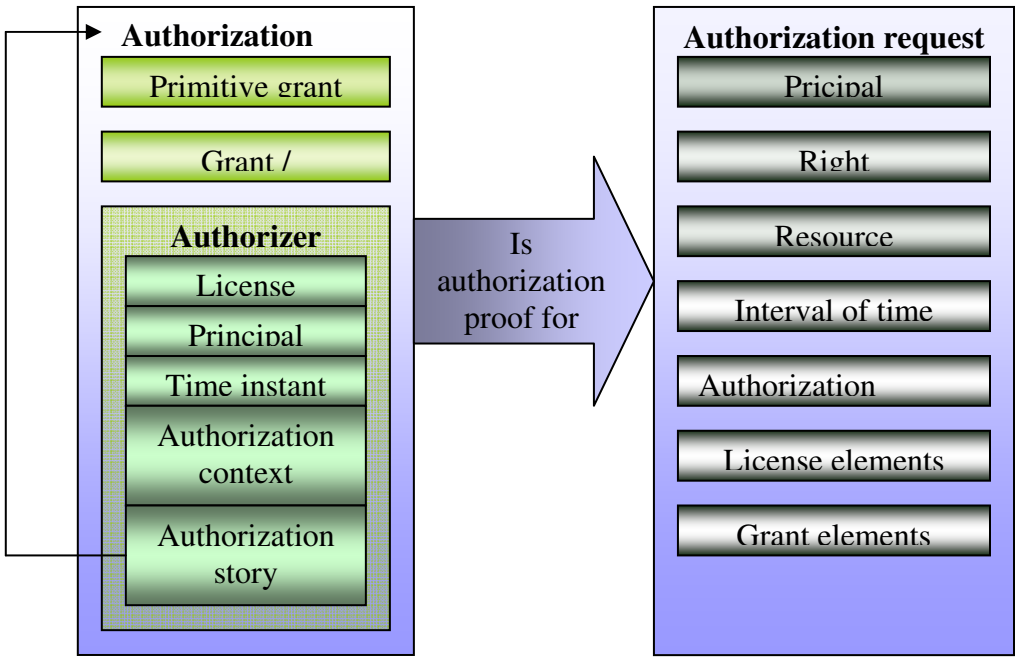



**Figure 65 Authorization model for REL**

An authorization context is a set of properties, each representing a statement and having one unique name (consisting in one Qualified Name and zero or more parameters) and one value.

An authorization story is composed by a primitive grant, a *Grant* or *GrantGroup* and optionally an authorizer. The authorization story is the authorization proof of an authorization request.

The authorizer consists in a *License*, a *Principal*, a time instant, an authorization context and an authorization story, [16].

# Annex C – Definitions for DIA description tools

The description tools specified by this part are organised in eight major categories [18]:

- **Usage Environment Description – UED**: involves User characteristics, Terminal capabilities, Network characteristics and Natural environment characteristics and enables the description of these various dimensions of the usage environment.
- **Bitstream Syntax Description Link – BSDLink**: allows the creation of different adaptation architectures based on the tools specified in the DIA, DID and MPEG-7.
- **Bitstream Syntax Description – BSD**: describes the syntax, for a high level structure, of the coding format of a binary media resource.
- **Terminal and Network QoS – AdapationQoS**: used to permit the generation of metadata describing the relationships between QoS constraints, adaptation operations that satisfy these constraints and the expected result. QoS constraints are related to network characteristics and terminal capabilities.
- **Universal Constraints Description – UCD**: enables the description of restrictions and optimisations on adaptation operations.
- **Session Mobility – SM**: specifies description tools intended to store information about the consumption of a Digital Item, i.e., about the interaction of a User with a Digital Item.
- **Metadata Adaptability – MA**: provides information that can be used to assist the operations of adapting metadata.
- **DIA Configuration – DIAC**: provides information to configure Digital Item Adaptation Engines. It identifies the DIA descriptors suggested for a specific adaptation and the location where the adaptation should take place (e.g., at the receiver side, sender side). It also identifies how choices/selections (see ISO/IEC 21000-2 [10]) should be processed.

# Annex D – Digital Item Processing

This section is a short introduction to the DIP concepts and shows how to embed DIP information into Digital Items. Some examples were created in order to understand the manner of using DIP metadata within Digital Items.

## D.1 Digital Item Methods

DIP functionality is included in Digital Items by using Digital Item Methods - DIMs. The following code is an example of a DIM:

```
function DIM_01( photo )
{
  var objectMap = DIP.getObjectMap( didDocument );
  var obj = objectMap.getObjectsOfType("urn:foo:Photo" );
  var cmp = obj[0].getElementsByTagName("Component").item(0);
  DIP.alert('This DIM method will print the given object of
             type urn:foo:Photo, included in a Resource
             Component',MSG_INFO);
  DIP.print(cmp);
}
```

**Example A Simple Digital Item Method**

In the above example the DIBOs used in the DIM are: DIP.*getObjectMap*, *getObjectsOfType*, *getElementsByTagName*, *DIP.alert* and *DIP.print*. DIBOs can also be operations related to other parts of MPEG-21 (e.g., DID: *areConditionsSatisfied*, *configureChoice*, *setSelection*; DII:*getElementsByIdentifier*, *getElementsByRelatedIdentifier*, *getElementsByType*; DIA: *adapt*; REL: *getLicenseInformation*, *queryLicenseAuthorization*; and DIP: *alert*, *execute*, *play*, *print*, *release*, *runDIM*, etc).

The exemplified DIM, when executed, will first display to the User an "alert" popup window with the given message, and then prints the object of the "*urn:foo:Photo*" type.

### Digital Item Method declaration

A DIM declaration is included within a DIDL *Component* element and it is composed by [3]:

- A DIM information descriptor – which is a DIP ***MethodInfo*** element contained in a DIDL *Descriptor-Statement* element. The content of this element is a sequence of ***Argument*** elements corresponding to the sequence of arguments of the DIM. If the DIM has no arguments then the ***MethodInfo*** element need not to be present. A DIM may be automatically executed if its *autorun* attribute value is true.
- A flag indicating the DIM declaration that is to be processed by the DIP Engine. This flag is represented by the DIP ***Label*** element, also contained in a DIDL *Descriptor-Statement* element. The content of this element is an Uniform Resource Identifier (URI) with the value: `urn:mpeg:mpeg21:2005:01-DIP-NS:DIM`.
- And a DIM definition that can be referenced or embedded in a *Resource* element of a DIDL *Component*. The DIM is referenced when its definition is contained by other Digital Item.

An example of a DIM declaration with external reference to the DIM definition and without arguments is presented next:

```
<Item>
  <Descriptor>
    <Statement mimeType="text/plain">Referenced DIM</Statement>
  </Descriptor>
  <Component id="dim_1">
    <Descriptor>
      <Statement mimeType="text/xml">
        <dip:MethodInfo /> <!-This MethodInfo can be absent -->
      </Statement>
    </Descriptor>
    <Descriptor>
      <Statement mimeType="text/xml">
       <dip:Label>urn:mpeg:mpeg21:2005:01-DIP-NS:DIM</dip:Label>
      </Statement>
    </Descriptor>
    <Resource mimeType=" application/mp21-method"
              ref="www.idibrowser.com/DIM#dim_1" />
  </Component>
</Item>
```

**Example B Referenced DIM definition**

Other examples of DIMs declarations are provided in Example C. The first DIM (dim_2) has two arguments and its definition is embedded using a CDATA section. The second DIM (dim_3) is embedded with base64 encoding.

```
<Item>
  <Component id="dim_2">
    <Descriptor>
      <Statement mimeType="text/plain">Embedded DIM with
CDATA</Statement>
    </Descriptor>
    <Descriptor>
      <Statement mimeType="text/xml">
        <dip:MethodInfo>
          <dip:Argument>urn:foo:Type1</dip:Argument>
          <dip:Argument>urn:foo:Type2</dip:Argument>
        </dip:MethodInfo>
      </Statement>
    </Descriptor>
    <Descriptor>
      <Statement mimeType="text/xml">
       <dip:Label>urn:mpeg:mpeg21:2005:01-DIP-NS:DIM</dip:Label>
      </Statement>
    </Descriptor>
    <Resource mimeType=" application/mp21-method"><![CDATA[
      function dim_2 (argument_1, argument_2)
      {
        /* DIM definition */
      }
      ]]>
    </Resource>
  </Component>
  <Component id="dim_3">
    <Descriptor>
      <Statement mimeType="text/plain">Embedded DIM with base
64</Statement>
    </Descriptor>
    <Descriptor>
      <Statement mimeType="text/xml">
       <dip:Label>urn:mpeg:mpeg21:2005:01-DIP-NS:DIM</dip:Label>
      </Statement>
    </Descriptor>
    <Resource mimeType=" application/mp21-method"
              encoding="base64">
      cjZVllYXI+MTk5MTwvUHJvZElzc3VlWWVhcj48QWJzdHJhY3RQcmVzZW50W
U4+TjwvQWJzdHJhY3RQ …
    </Resource>
  </Component>
</Item>
```

**Example C Embedded DIM with arguments**

## D.2 DIXOs (J-DIXOs) in Digital Items

DIXOs are operations defined in a specified DIXO Language (it is chosen by the implementer) used to extend the functionality provided by DIBOs and that are called by a specified DIBO (runMyDIXO – e.g., *runJDIXO* in the case of Java implementation). DIXOs also include bindings to DIBO in DIXO Language.

J-DIXOs (DIXOs implemented in Java) can be a Java class, a jar file containing Java classes, or a base directory that contains Java classes. It can be incorporated into a Digital Item in two steps: J-DIXO declaration and J-DIXO definition.

J-DIXO declaration shall include [3]:

- A list of J-DIXOs defined in the associated resource within the DIDL *Component* element. This list is placed within a DIDL *Descriptor-Statement*, in ***JDIXOClasses*** elements (which are not necessary present when the associated resource does not define any J-DIXO). The content of ***JDIXOClasses*** is a sequence of Class sub-elements indicating the fully qualified Java class name of a Java J-DIXO class that is intended to be invoked as a J-DIXO.
- A flag similar to the DIM declaration that is represented by a ***dip:Label*** element in a DIDL *Descriptor-Statement* and that indicates the J-DIXO to be processes by the DIP Engine. The value of this flag is: `urn:mpeg:mpeg21:2005:01-DIP-NS-DIXO:Java`.
- And J-DIXO definitions referenced or embedded (as base64-encoded bytes of binary data) into the *Resource* child of the *Component*.

`Example D` shows how to reference a J-DIXO contained in a jar file.

```
<Item>
 <Descriptor>
   <Statement mimeType="text/plain">Referenced J-DIXO</Statement>
 </Descriptor>
 <Component id="jdixo_1">
  <Descriptor>
    <Statement mimeType="text/xml">
     <dip:Label>urn:mpeg:mpeg21:2005:01-DIP-NS-
           DIXO:Java</dip:Label>
    </Statement>
  </Descriptor>
  <Descriptor>
    <Statement mimeType="text/xml">
     <dip:JDIXOClasses>
       <dip:Class>
         idibrowser.idiprocessor.jdixo_1
       </dip:Class>
     </dip:JDIXOClasses>
    </Statement>
  </Descriptor>
  <Resource mimeType=" application/java-archive"
             ref="jdixos.jar" />
 </Component>
</Item>
```

**Example D A referenced J-DIXO**

## D.3 Object Map and dip:ObjectType

The Object Map is a map consisting of DID Objects and a list of DIMs and describes the relationships between them.

In a Digital Item each Object is associated with an Object Type using a dip:ObjectType element (contained in a DIDL *Descriptor-Statement*). An Object Type is associated with an argument of a DIM and this argument shall correspond only to a single Object Type [23].

```
<Item>
  <Descriptor>
   <Statement mimeType="text/plain">Photo</Statement>
  </Descriptor>
  <Descriptor>
   <Statement mimeType="text/xml">
      <dip:ObjectType>urn:foo:Photo</dip:ObjectType>
   </Statement>
  </Descriptor>
  <Component>
    <Resource mimeType="image/jpeg" ref="photo1.jpg" />
  </Component>
</Item>
<Item>
 <Descriptor>
   <Statement mimeType="text/plain">DIM</Statement>
 </Descriptor>
 <Component id="DIM_01">
  <Descriptor>
    <Statement mimeType="text/xml">
     <dip:Label>urn:mpeg:mpeg21:2005:01-DIP-NS-DIM</dip:Label>
    </Statement>
  </Descriptor>
  <Descriptor>
    <Statement mimeType="text/xml">
     <dip:MethodInfo>
        <dip:Argument>urn:foo:Photo</dip:Argument>
     </dip:MethodInfo>
    </Statement>
  </Descriptor>
  <Resource mimeType=" application/mp21-method"><![CDATA[
      function DIM_01( photo )
      {
        var objectMap = DIP.getObjectMap( didDocument );
        var obj = objectMap.getObjectsOfType("urn:foo:Photo" );
        var cmp = obj[0].getElementsByTagName("Component").item(0);
        DIP.alert('Print the photo',MSG_INFO);
        DIP.print(cmp);
      }]]>
  </Resource>
 </Component>
</Item>
```

**Example E Using dip:ObjectType in Digita 1**

In the previous example, an object (the first Item) is defined as being of type *urn:foo:Photo*. The *Item* contains an image in its *Component-Resource*. The second Item incorporates the DIM that accepts as argument only an object of type *urn:foo:Photo* . In other words, the argument of DIM_01 will be the first *Item*. This DIM will print the content of the *Resource* contained in the *Component* element of the given *Item* (the first).

## D.3 DIP Engine

In order to implement the part 10 of the MPEG-21, a DIP Engine tool is required. This interprets and executes the User's interactions with the Digital Items and parts thereof. The DIP functionalities assure the availability to the User, when he receives a Digital Item, of a list of DIMs that can be applied to it. This means that the DIP functionality is defined by DIMs (for example methods as: print, save, adapt the content of Digital Item, insert User comments, etc.) that call Digital Item Basic Operations (DIBOs) and Digital Item Extended Operations (DIXOs). These operations are implemented on the DIP Engine and their role is to provide the basic functionality for DIMs.

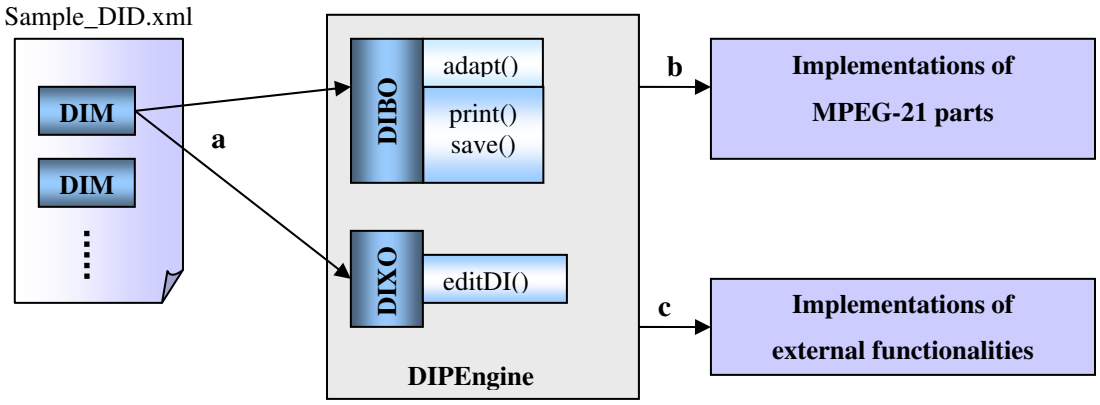



**Figure 66 DIP functionalities and the role of DIPEngine**

Figure 66 illustrates the way that DIMs from a DID (after the User has selected a DIM) can be executed by the DIP Engine, calling the specific implemented operations:

- **a** represents the calls from the DIM definition to DIBOs and DIXOs;
- **b** are the calls from DIBO to the tools implementing the functionalities of the MPEG-21 parts;
- **c** indicates the calls from DIBOs and DIXOs to external tools implementing other functionalities than MPEG-21 specific.

Including DIP functionality into multimedia framework enables the dynamic interactivity of the Users with the Digital Items.

# Annex E – API of the IDIProcessor module of IDI Browser

This annex provides the API of the implemented IDIProcessor Java class which is the base of the IDIProcessor module.

## E.1 IDIProcessor class

```
  public class IDIProcessor
  { …
     /** Opens a Digital Item Description.
        * @param digitalItemURL the URL of the DID
        *  @return  boolean  value  indicating  if  the  DID  was  successfully
opened.
        */
       public boolean openDID(String digitalItemURL) {…}
/*-------------------------------------------------------------------*/
     /**
        * Clears a Digital Item Description.
        */
       public void clearDID(){…}
/*-------------------------------------------------------------------*/
     /**
        * Loads the top DID element of the Digital Item Description.
        * @return the top element as a specific XML string
        */
       public String loadTopElement(){…}
/*-------------------------------------------------------------------*/
     /** Gets a DID element by reference.
        * @param elementReference the reference of the element
        * @return the element as a specific XML string
        */
       public String getElement(String elementReference) {…}
/*-------------------------------------------------------------------*/
     /**
        * Moves back to the previous navigation stage.
        * @return the DID element to be visualized, as a specific XML string.
        */
       public String processBack();
```

```
/*-------------------------------------------------------------------*/

     /**
        * Processes the selections made by the client.
        * @param elementReference the reference of the DID element
        *                         whose Choices sub-elements
        *                         contain the Selections
        * @param trueSelectionIDs the selections set to true
        * @param falseSelectionIDs the selections set to false
        *@return the DID element refered by elementReference if not
        *              null as a specific XML content.
        */
       public String processSelections(String elementReference,
                                       String trueSelectionIDs,
                                       String falseSelectionIDs){…}
/*-------------------------------------------------------------------*/

     /**
        * Gets the DID
        * @return the DID as a string
        */
       public String getDID(){…}
/*-------------------------------------------------------------------*/

     /**
        * Gets the DID location
        * @return the string representing the DID location
        */
       public String getDIDLocation(){…}
/*-------------------------------------------------------------------*/
     /**
        * Verifies if DID contains or not DIP Label elements
        * @return the boolean value indicating if there are DIP Label in the
        *         DID
        */
       public boolean hasDIP()    {…}
```

# Annex F – API of the DIEngine module of IDI Browser

The next sections describe the methods implemented in the Java classes of the DIEngine module.

## F.1 IDIEngine interface

```
public interface IDIEngine
{
    /** Opens a Digital Item Description.
       * @param digitalItemURL the URL of the DID (from a local repository)
       * @return boolean value indicating if the DID was successfully
       *                opened.
       */
      public boolean openDID(String digitalItemURL)
            throws UnmarshallFailureException;
/*-------------------------------------------------------------------*/
    /**  Clears a Digital Item Description. */
      public void clearDID();
/*-------------------------------------------------------------------*/
    /** Returns the CONTAINER or ITEM that is at the top of a DIDL
       *         document. Information to be displayed is filtered.
       * @return a DIDL element (CONTAINER or ITEM)
       */
      public DIDBaseType loadTopObject();
/*-------------------------------------------------------------------*/
    /**
       * Gets the element having the given reference as a specific object.
       * Only CONTAINERS and Items can be returned. Information to be
       * displayed is filtered.
       *
       * @param elemReference the elemenent reference
       * @return the object representing the element.
       * @throws InvalidObjectReference If the reference does not correspond
       *          to a CONTAINER or ITEM object
       */
      public DIDBaseType getElement(String elemReference)
            throws InvalidObjectReference;
```

```
/*-----------------------------------------------------------------*/
     /**
        * Returns the previous ITEM or CONTAINER while respecting any
conditions that it may have.
        * @return DIDLBaseType the parent ITEM or CONTAINER.
        */
       public DIDBaseType processBack();
/*-----------------------------------------------------------------*/
     /**
        * Registers a set of selections pertencing to the Choices of the
        * given Element. The strings trueSelectionIDs and falseSelectionIDs
        * contain the Selections identifiers such as:
        * "selectionID selectionID ... selectionID"
        *
        * @param elementRef the reference to the Item containing the Choices
        * @param trueSelectionIDs the identifiers of the Selection(s) that
        *                         were set to TRUE
        * @param falseSelectionIDs the identifiers of the Selection(s) that
             * were set to FALSE
        * @return the Item containing the Choices with the Selections
             * processed.
        */
       public DIDBaseType processSelections(String elementRef,
                                            String trueSelectionIDs,
                                            String falseSelectionIDs);
/*-----------------------------------------------------------------*/
     /**
        * Gets the Object that represents the entire DID
        * @return the DID Object
        */
       public DIDLType getDID();
/*-----------------------------------------------------------------*/
       /**
        * Gets the DID as a DOM Node
        * @return the DIDL element
        */
       public Node getDIDNode();
/*-----------------------------------------------------------------*/
```

```
    /**
       * Gets the DIDL document corresponding to the DID
       * @return the DIDLDocument
       */
      public DIDLDocument getDIDLDocument();

/*-----------------------------------------------------------------*/

      /**
       * The processed element (Item or Container) is included in a DIDL
       * parent element, having the same atributes (namespaces) as the
       * initial DID file that is consumed
       * @param processedElement is the resulted DIDBaseType element after
       *                       the processing
       * @return a node containing a DIDL that inlcudes the processed
       *         element within
       */
      public String setDIDLparent(DIDBaseType processedElement);

/*-----------------------------------------------------------------*/

      /**
       * Get the value for public variable m_hasDIP indicating if the DID
       * contains or not DIP Label elements
       * @return the boolean value for m_hasDIP
       */
      public boolean hasDIP();
}
```

## F.2 ElementProcessor class

```
 public class IDIProcessor {
    /** Filters a CONTAINER or ITEM removing information that is not to be
       * presented.
       * @param originalElement is the DIDBaseType element to be processed
       * @return DIDBaseTypeElement is the resulted element to be presented
       *     or return null if an associated condition is not satisfied
       */
      public DIDBaseType processElement(DIDBaseType originalElement)
            throws InvalidObjectType {…}
/*-----------------------------------------------------------------*/
```

```
/*-----------------------------------------------------------------*/
     /** filter the Item by Choice subelements in order to NOT display the
        * respectives Choices after processing the User Selections
        * @param element is the Item to be filtered
        * @return the filtered Item
        * @throws InvalidObjectType if the DIDBaseType element is not an Item
        */
       public DIDBaseType processItemWithChoices(DIDBaseType element)
       throws InvalidObjectType   {…}
/*-----------------------------------------------------------------*/
     /** Gets the first description belonging to the given element
        * @param element the DIDBaseType element (Item or Container)
        * @return a Descriptor object or null if no Descriptor exists
        */
       public DescriptorType getFirstDescription(DIDBaseType element){…}
/*-----------------------------------------------------------------*/
     /** Returns the LIST of CONDITIONS of the Element and null otherwise
        * @param element the DIDBaseType element
        * @return a List of Conditions available for the element
        */
       public List getConditions(DIDBaseType element) {…}
/*-----------------------------------------------------------------*/
     /** Filter the subelements that are not respecting the conditions
        * within the current Element (Item or Container)
        * @param element is an Item or a Container
        * @return DIDBaseType representing the filtered element
        */
     public DIDBaseType filterSubelementsByConditions(DIDBaseType element)
                 {…}
/*-----------------------------------------------------------------*/
     /** Tests a set of conditions that afect a given element
        * @param conditions the list of Conditions for the Element
        * @return a boolean TRUE value indicating that Conditions are
        *            satisfied or FALSE if Conditions are not satisfied
        */
       public boolean testConditions(List conditions) {…}
/*-----------------------------------------------------------------*/
```

```
     /** Delete recursively all the Comments from the current element and
from all its subelements
        * @param e is the DIDBaseType element to be filtered by Comments
        */
       public void deleteCommentElem(Node elem){…}
/*-------------------------------------------------------------------*/
     /**
        * Get the children of the given element and generate a reference
        * for each child
        *
        * @param element is the Item/Container to be processed
        * @return a list containing the child elements and their references
        */
public ArrayList<ElemReference> getChildrenAndReferences(DIDBaseType
                                                              element){…}
…
}
```

## F.3 SelectionManager class

```
     /*This class manages a table which contains IDs of selections
      *and their respective values. */

public class SelectionManager {
     /** Get the HashTable of the selections ID and their value     */
       public Hashtable<String, Boolean> getSelectionsTable() {…}
/*-------------------------------------------------------------------*/
     /** Set the HashTable containing the selections ID and their value */
public void setSelectionsTable(Hashtable<String, Boolean> selectionTable) {…}
/*-------------------------------------------------------------------*/
     /** Clears all selections */
       public void clear() {…}
/*-------------------------------------------------------------------*/
     /** Adds new selections. Selections are received as a sequence of
        * identifiers(selectionID) separeted by spaces<br>
        *   <p>"first_ID second_ID third_ID".
        * @param trueSelections selections set to TRUE
        * @param falseSelections selections set to FALSE
        */
public void addSelections(String trueSelections, String falseSelections){…}
/*-------------------------------------------------------------------*/
```

```
    /** Adds selections as being set to TRUE. Selections are received as a
       * sequence of identifiers separeted by spaces<br>
       *  <p>"first_ID second_ID third_ID".
       * @param selections selections to store as being TRUE
       */
      public void addTrueSelections(String selections) {…}
/*-------------------------------------------------------------------*/
    /** Adds selections as being set to FALSE. Selections are received as a
      * sequence of identifiers separeted by spaces<br>
       * <p>"first_ID second_ID third_ID".
       * @param selections selections to store as being TRUE
       */
      public void addFalseSelections(String selections) {…}
/*-------------------------------------------------------------------*/
    /** Removes the specified selections
      *@param selections array with the identifiers of the selections to
       * remove
       */
      public void deleteSelections(String[] selections) {…}
/*-------------------------------------------------------------------*/
    /** Checks the status of a specific selection
       * @param selection the selection to be tested
       * @return a Boolean Object containing the status of the selection or
       * null if the selection is not registered
       */
      public Boolean checkSelection(String selection) {…}
/*-------------------------------------------------------------------*/
    /** Verifies if a lists of selections evaluates to TRUE. The selections
      * are passed as a list separeted by spaces<br>
       * and the test is performed as an AND, i.e., if one the parameters is
       *  evaluated to FALSE the condition is not <br> satisfied.
       * @param selections the list of selections to evaluate
       * @return a TRUE Boolean specifying if the given selections are set
       *                  to true and contained in hash table
       *                 or FALSE Boolean value if one or more selections
       *                  are false or not contained in hash table
       */
      public boolean checkConditionsForTrue(String selections) {…}
/*-------------------------------------------------------------------*/
```

```
     /** Verifies if a lists of selections evaluates to TRUE. The selections
        * are passed as a list separeted by spaces<br>
        * and the test is performed as an AND, i.e., if one the parameters is
        * evaluated to FALSE the condition is not <br> satisfied.
        * @param selections the list of selections to evaluate
        * @return a TRUE Boolean specifying if the given selections are set
        *             to true and contained in hash table
        *           or FALSE Boolean value if one or more selections are
        *            false or not contained in hash table
        */
       public boolean checkConditionsForTrue(List selections) {…}
/*-------------------------------------------------------------------*/
     /** Verifies if a lists os selections evaluates to FALSE. The selections
        * are passed as a list separeted by spaces<br>
        * and the test is performed as an AND, i.e., if one the parameters is
        * evaluated to TRUE the condition is not <br> satisfied.
        * @param selections the list of selections to evaluate
        * @return TRUE if all selections are false (and contained in hash
        *        table) or FALSE if at least one of the selections is not
        *        registered or is true
        */
       public boolean checkConditionsForFalse(String selections) {…}
/*-------------------------------------------------------------------*/
     /** Verifies if a lists os selections evaluates to FALSE. The selections
        * are passed as a list separeted by spaces<br>
        * and the test is performed as an AND, i.e., if one the parameters is
        * evaluated to TRUE the condition is not <br> satisfied.
        * @param selections the list of selections to evaluate
        * @return TRUE if all selections are false (and contained in hash
        *    table) or FALSE if at least one of the selections is not
        *        registered or is true
        */
       public boolean checkConditionsForFalse(List selections) {…}
...
}
```

## F.4 ElemReference class

```
     /** This class creates an object containing an element of DIDBaseType
      * (Item or Container) and its corresponding reference (as Integer)
      */
public class ElemReference implements Cloneable {

     /** Set the  element
        * @param elem is a Item or Container
        */
       public void setElem(DIDBaseType elem) {…}
/*-------------------------------------------------------------------*/
     /** Set the reference
        * @param reference is an integer value for
        *                  the reference atribute of the element
        */
       public void setReference(Integer reference) {…}
/*-------------------------------------------------------------------*/
     /** Get the Element
        * @return the element (Item/Container)
        */
       public DIDBaseType getElem() {…}
/*-------------------------------------------------------------------*/
     /** Get the reference of the element
        * @return the reference of the element
        */
       public Integer getReference() {…}
/*-------------------------------------------------------------------*/
     /** Create a copy of the DIDBaseType element and of its reference
        * @return the ElemReference  object containing the copy of the
        *         element and its reference
        */

     public ElemReference clone() {…}
     …
}
```

## F.5 State class

```
     /** BackState class will define the states of a Digital Item during the
      * browsing.
      * A state is defined by the reference of the element sent to User
      * and the selections made until that moment.
      *  */
public class State {

     /** Set selections state from SelectionManager
         * @param selections are the selection made until the moment*/
         public void setSelectionsTable(SelectionManager selections) {…}
/*-----------------------------------------------------------------*/
     /**  Set the element (Item/Container) reference
          * @param elemID is the value for element reference*/
         public void setReference(String elemID) {…}
/*-----------------------------------------------------------------*/
     /**   Get the selections made until the moment
          * @return m_selectionManager is the table containing the selections
          *        references and their value */
         public SelectionManager getSelectionsTable() {…}
/*-----------------------------------------------------------------*/
     /** Get the reference of the element (Item/Container)
          * @return elementReference is the identifier of the element
          * */
         public String getReference(){…}

...
}
```

# References

[1] http://www.enthrone.org/, August 2006

[2] Information Technology – Multimedia Framework (MPEG-21) – Part 1: Vision, Technologies and Strategy, ISO/IEC 21000-1:2002, 2002

[3] G. Drury, Rik Van de Walle, I. Burnett, M Kim, V. Swaminathan, „Study text of ISO/IEC 21000-10 FCD DIP“, ISO/IEC N6943, January 2005, Hong Kong, China

[4] http://w3.inescn.pt/internet/, September 2006

[5] Information Technology – Coding of moving pictures and associated audio for digital storage at up to about 1,5 Mbit/s, ISO/IEC 11172, 1993

[6] Information Technology – Generic coding of moving pictures and associated audio information, ISO/IEC 13818, 2000

[7] Information Technology – Coding of audio-visual objects, ISO/IEC 14496, 2001

[8] Information Technology – Multimedia content description interface, ISO/IEC 15938, 2002

[9] Information Technology – Multimedia Framework (MPEG-21) – Part 2: Digital Item Declaration, ISO/IEC 21000-2:2003, March 2003

[10] Van de Walle, I. Burnett, and G. Drury, “ISO/IEC 21000-2 DID 2nd Edition” MPEG document N6409, Munich March 2004

[11] Information Technology – Multimedia Framework (MPEG-21) – Part 2: Digital Item Declaration, ISO/IEC FDIS 21000-2:2005(E), April 2005

[12] T. Berners-Lee, R. Fielding, U.C. Irvine, L. Masinter, “Uniform Resource Identifiers (URI): Generic Syntax”, IETF RFC 2396, August 1998

[13] Information Technology – Multimedia Framework (MPEG-21) – Part 3: Digital Item Identification, ISO/IEC 21000-3:2003, March 2003

[14] Pedro Carvalho, “Multimedia Content Adaptation For Universal Access”, MSc thesis, Faculty of Engineering of the University of Porto, September 2004.

[15] Information Technology - Multimedia Framework - Part 4: Intellectual Property Management and Protection Components, ISO/IEC 21000-4 FCD IPMP Components, April 2005

[16] Information Technology – Multimedia Framework (MPEG-21) – Part 5: Rights Expression language, ISO/IEC 21000-5:2004, 2004

[17] Information Technology – Multimedia Framework (MPEG-21) – Part 6: Rights Data Dictionary, ISO/IEC 21000-5:2004, 2004

[18] Information Technology – Multimedia Framework (MPEG-21) – Part 7: Digital Item Adaptation, ISO/IEC FDIS 21000-7:2004, March 2004

[19] F. Keukelaere, G. Drury, C. Timmerer, X. Wang, "Text of ISO/IEC 21000-8 CD MPEG-21 Reference Software", MPEG Integration Group, ISO/MPEG N6470, March 2004

[20] I. Burnett, G. Drury, C. Timmerer, R. VanDeWalle, X. Wang, J. C. Dufourd, " MPEG-21 Reference Software WD4.0", MPEG MDS Group, ISO/MPEG N6254, December 2003

[21] http://www.enikos.com, August 2006

[22] D. Singer, "ISO/IEC CD 21000-9", MPEG Systems Group, ISO/MPEG N6331, April 2004

[23] R. Walle, I. Burnett, G. Drury, M. Kim, V. Swaminathan, "ISO/IEC 21000-10 CD – Part 10: Digital Item Processing", MPEG MDS Group, ISO/MPEG N6173, December 2003

[24] http://www.ecma-international.org/publications/standards/Ecma-262.htm, August 2006

[25] "Text of DTR of ISO/IEC 21000-11", MPEG Requirements Group, ISO/MPEG N6392, March 2004

[26] Information Technology - Multimedia Framework – Part 11: Evaluation Tools for Persistent Association Technologies, ISO/IEC WD 21000-11:2004, July 2004

[27] "Text of ISO/IEC 21000-12/PDTR Test Bed for MPEG-21 Resource Delivery", MPEG Integration Group, ISO/MPEG N6255, December 2003

[28] Chun-Jen Tsai, and Chung-Neng Wang, "Study of ISO/IEC PDTR 21000-12", MPEG document N6471, Munich, March 2004

[29] Information Technology - Multimedia Framework - Part 14: Conformance Testing, ISO/IEC CD 21000-14, Busan, April 2005, Korea

[30] "Current Vision on Event Reporting in MPEG-21", MPEG Requirements Group, ISO/MPEG N5871, July 2003

[31] F. Nuttall, Y. Song, K. Ji, A. Tokmakoff, N. Rump, "MPEG-21 Event Reporting WD (v1.0)", MPEG-21, MPEG MDS Group, ISO/MPEG N6419, March 2004

[32] F. Nuttall, Kyunghee Ji, A. Tokmakoff, N. Rump, Th. Kummer-Hardt, "Study of ISO/IEC 21000 Event Reporting Final Committee Draft", ISO/IEC N7735, October 2005, Nice, France

[33] Text of ISO/IEC 21000-16 FDIS", System group, ISO/IEC MPEG2005/N7247, April 2005, Busan

[34] Information Technology - Multimedia Framework - Part 17: Fragment Identification for MPEG Resources, "Study of ISO/IEC 21000-17 CD MPEG-21 FID (Fragment Identification of MPEG Resources)", Busan, April 2005, Korea

[35] MDS, "Request to make ISO/IEC 21000-18 publicly available", ISO/IEC N6951, Hong-Kong, January 2005, China

[36] Information Technology - Multimedia Framework - Part 18: Digital Item Streaming, "Text of ISO/IEC 21000-18 FCD Digital Item Streaming", Montreux, Switzerland, April 2006

[37] Zvi Lifshitz (Optibase Ltd.) and members of the MUFFINS project, "A simple framework that exercises MPEG-21 technologies", ISO/IEC MPEG2002/8850, October 2002

[38] Multimedia Framework For INteroperability in Secure (MPEG-21) Environments, December 2003

[39] H. Hellwagner, Klagenfurt University-Austria, DANAE, QoS Concertation Meeting (CG-2), Nice, December 2004

[40] http://danae.rd.francetelecom.com, November 2005

[41] ENTHRONE Consortium, „IMS Proof-of-Concept Presentation", Deliverable D06, January 2005

[42] http://multimedialab.elis.ugent.be/demo.asp, July 2006

[43] Robbie De Sutter, Sam Lerouge, Wesley De Neve, Peter Lambert, Rik Van de Walle – Ghent Univesity, "Advanced mobile multimedia applications: using MPEG-21 and Time-Dependent Metadata" , 2003

[44] Robbie De Sutter, Sam Lerouge, Jeroen Bekaert and Rik Van de Walle – Multimedia Lab, "Dynamic Adaptation of Streaming MPEG-4 Video for Mobile Applications" , 2003

[45] http://mpeg-21.itec.uni-klu.ac.at/cocoon/mpeg21/_mpeg21Demo.HTML, August 2006

[46] http://ro.uow.edu.au/, August 2006

[47] Shane Lauf, Ian Burnett – University of Wollongong, Australia, "Implementation of a Mobile MPEG-21 Peer", November 2005

[48] Filip Hendrickx – IMEC, Tom Beckers - Ghent University – IBBT, Nico Oorts – VRT, Rik Van De Walle - Ghent University – IBBT; "An Integrated Approach for Device Independent Publication of Complex Multimedia Documents", Internet and Multimedia Systems, and Applications Conference - Honolulu, Hawaii, USA, 2005

[49] Information Technology - Multimedia Framework - Part 15: Event Reporting, "Text of FDIS ISO/IEC 21000-15 MPEG-21 ER", Bangkok, Thailand, January 2006

[50] http://mpeg-21.itec.uni-klu.ac.at /cocoon/mpeg21/_mpeg21Demo.xml, August 2006

[51] www.AXMEDIS.org, August 2006

[52] P. Bellini, P. Nesi, D. Rogai, A. Vallotti, "AXMEDIS Tool Core for MPEG-21 Authoring/Playing", Proceedings of the First International Conference on Automated Production of Cross-Media Content for Multi-Channel Distribution (AXMEDIS'05) , 2005

[53] http://jibx.sourceforge.net/, August 2006

[54] P. Carvalho, M. Andrade, G. Ciobanu, L. Ciobanu, "An Application for Browsing and Interacting with MPEG-21 Digital Items", submitted in July, 2006

[55] Institute of Electrical and Electronics Engineers – IEEE, Standard Computer Dictionary: "A Compilation of IEEE Standard Computer Glossaries", New York, 1990

[56] José Joaquim Magalhães Moreira, "wsQL – Uma arquitectura de software baseada em Web Services", MSc thesis, Faculty of Engineering of the University of Porto, December 2005

[57] http://java.sun.com, August 2006

[58] http://jakarta.apache.org/tomcat, August 2006

[59] http://jakarta.apache.org/ant, August 2006

[60] www.php.net, August 2006

[61] http://httpd.apache.org/, August 2006

[62] http://xmlbeans.apache.org/, August 2006

[63] http://java.sun.com/j2se/javadoc/, August 2006

[64] http://java.sun.com/applets/, May 2006