Multimedia Terminal Architecture: An Inter-Operable Approach

Beilu Shao[1], Marco Mattavelli[1], Maria T. Andrade[2], Samuel Keller[1], Giorgiana Ciobanu[2], Pedro Carvalho[2]
[1]*Swiss Federal Institute of Technology (EPFL), Switzerland*
[2]*INESC Porto, Portugal*
beilu.shao@epfl.ch, marco.mattavelli@epfl.ch, maria.andrade@inescporto.pt, samuel.keller@epfl.ch, giorgiana.ciobanu@ inescporto.pt, pedro.carvalho@inescporto.pt

Abstract

This paper addresses the inter-operability problem of multimedia terminal in media content delivery over heterogeneous networks and devices. We design and implement a terminal which includes a content browser providing presentation, navigation and interaction with MPEG-21 Digital Item Declaration (MPEG-21 DID) in order to support universal media access (UMA). We optimized the architecture in a client-server distributed approach with Web Service support. This terminal enables the MPEG-21 standard compliant content accessible on different terminal devices via common Web browsers. Such a design strategy illustrates a next-generation multimedia terminal supporting inter-operability in multimedia content adaptation over a heterogeneous delivery chain.

1. Introduction

The latest development of multimedia computing and great success of the Internet have drawn tremendous attention and contribution from both academia and industry. Many different techniques have been developed to exploit the efficiency and the effectiveness of multimedia delivery, from video coding with outstanding rate-distortion performance to reliable and adaptive video communication. Different kinds of transmission and terminal devices have been deployed, from PC, Set-Top Box to mobile devices like PDA. The capabilities and working scenarios of these devices usually vary widely in terms of network bandwidth, computing power, screen display, decoding capabilities and user interaction functionalities. At the same time, a variety of new media content formats, standard-compliant or proprietary, have emerged resulting in a diversity of media itself. In summary, multimedia content delivery has been growing from single format video transmission over the Internet to complex multimedia adaptive delivery across heterogeneous networks, terminals, and users [1].

Along this trend, new standards have emerged. MPEG-21[2] has been developed to provide an open multimedia framework to guarantee inter-operability focusing on how the entities of multimedia application infrastructure services should relate, integrate, and interact. Of particular importance towards the fulfillment of universal media access is the availability of an open object data model. MPEG-21 has delivered one specification addressing precisely this aspect: the Digital Item Declaration (DID) [3].

However, current multimedia terminal design rarely takes into consideration the requirements to support the MPEG-21 data model for enhanced value-added services and thus, current terminals do not offer the necessary functionality. The overall heterogeneity clearly imposes new demands to the terminal design, increasing its complexity to be able to provide efficient and more complex processing and presentation of audio-visual content rather than simple video decoding.

This paper attempts to investigate the multimedia terminal design in support of the inter-operability requirements for content delivery by using the MPEG-21 Digital Item Model, a Client-Server architecture and a Web-oriented strategy. The distributed approach is achieved through the use of Web Service technologies, allowing the deployment of functionalities in a distributed fashion. This has the major benefit of freeing the client (the end user terminal) of the power-consuming processing tasks. Together with the Web-oriented strategy, it enables virtually any commercially available user device with standard Web browsers installed, to receive and consume complex multimedia objects. The proposed terminal is based on two major components, the terminal middleware and the MPEG-21 Digital Item Browser. Both components are implemented in a distributed approach, interacting among them to present in the best possible format the complex multimedia objects to the end user.

This paper is organized as follows. Section 2 presents a short overview of the related prior work. In Section 3, we present the Distributed Digital Item Browser, the key component of the terminal to support MPEG-21 Digital Item Declaration for inter-operability. We also highlight the flexible architecture design achieved with the Client-Server and Web Service approaches. Section 4 represents the multimedia terminal middleware design and the function coordination with the Distributed Digital Item Browser. In Section 5, we describe the implementation details of the terminal. Section 6 draws the conclusion.

2. Overview of Related Prior Work

Universal Multimedia Access (UMA) represents the concepts of the access to multimedia information by any terminal through any network [1], [4]. The objective of UMA technology is to make available different presentations of the same multimedia content, more or less complex, e.g., in terms of media formats or transmission bandwidth, suiting different content creation, communication networks and consuming terminals.

To achieve the goal of UMA, i.e. inter-operability during delivery chain, content compliant with MPEG-21 Digital Item Declaration (MPEG-21 DID) has been widely used [3]. The MPEG-21 DID is an XML structured data type that describes the structure and the relation among the various components of Digital Item (DI). Typically it cannot be simply presented to average users through a normal XML editing tool.

The DID specification does not specify how DIs should be presented to the user. Accordingly, a variety of different implementations can be realised, providing different views and ways of presenting and enabling the interaction with DIs. Usually, applications are developed according to the usage scenario in view or to the device they are supposed to be installed in, thus providing non-interoperable solutions. Invariably, the focus of the work is in ensuring accessibility, flexibility and performance of the Peers on dedicated terminal devices.

Existing research work has been developed in [5]-[9]. However, most of these solutions for browsing MPEG-21 DIs have portability problems, being often restricted to one type of terminal device or type of application. The Distributed DI Browser, coordinated under a terminal middleware with other functional components intends to solve the portability problem and proposes a modality of processing and visualizing complex DIs on the majority of terminal devices.

3. MPEG-21 Digital Item Browser

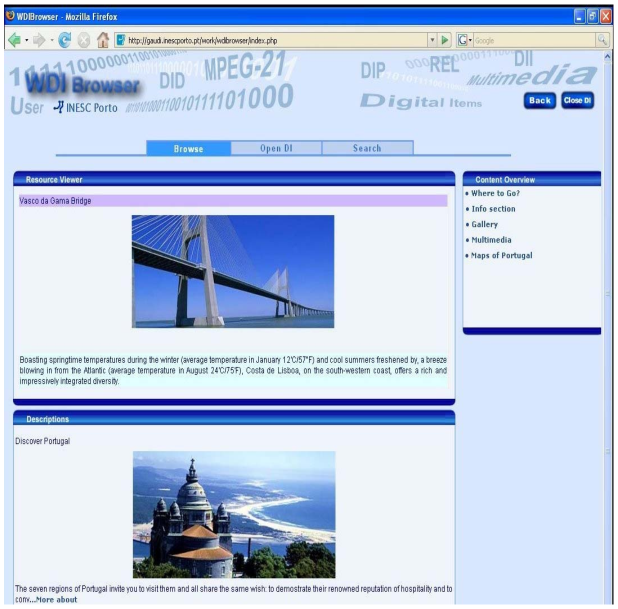
Considering that one of the goals to achieve was to obtain a terminal as interoperable as possible, it was decided that the content should be represented also in an interoperable format, more precisely according to the MPEG-21 Digital Item Declaration (DID) specification. This has imposed the need to develop a module for the terminal that would be able to process the MPEG-21 DIDs and presenting them in a friendly manner to the end-user. The Distributed Digital Browser (Distributed DI Browser) is this module and is thus responsible for rendering and presenting the content to end-users, allowing the user to interact with the content.

3.1. MPEG-21 DID Support and Web-oriented approach for Delivery Inter-operability

Transparent access to content is not always simple or possible due to the great heterogeneity of the information that can be found as well to the diversity of multimedia-enabled end-user equipment. The solution proposed in this paper to the effective access to varied multimedia information and enabling simple interaction with complex objects regardless of the terminal device, relies on the use of the MPEG-21 specifications, notably the MPEG-21 DIDL and the use of the DDI Browser implemented using a Web-oriented approach.

On one side, the MPEG-21 DIDL defines a standard XML-based model to express the structure of complex digital objects and to describe its components (media resources or additional descriptions). These media components do not need to have a specific encoding format, nor do the descriptions need to follow a given metadata model. There is complete freedom to include in the Digital Item any type of component and all included components are described in the DID using a standard XML-based language. This obviously motivates interoperability, as currently all Web-enabled devices understand XML.

On the other side, presenting XML directly to the end-user is not the friendliest possible approach. Although XML is very convenient because it is both machine and human readable, the average user is not used to navigate in complex XML trees. The explosion of popularity of the World Wide Web and of Web applications, suggests that a Web-oriented implementation would cover the requirements of a considerable number of application scenarios. To date, HTML Web pages are one of the friendliest ways of presenting complex multimedia content, to which the average user is very familiar with. In addition it would provide a solution for the portability problem, as MPEG-21 DIDs could thus be presented using any normal Web browser, which all multimedia-enabled devices already are equipped with. Accordingly, it was decided to adopt a Web-oriented approach for the development of the MPEG-21 Distributed DI Browser [10].



Vasco da Gama Bridge

Boasting springtime temperatures during the winter (average temperature in January 12'C/57"F) and cool summers freshened by, a breeze blowing in from the Atlantic (average temperature in August 24'C/75'F), Costa de Lisboa, on the south-western coast, offers a rich and impressively integrated diversity.

Discover Portugal

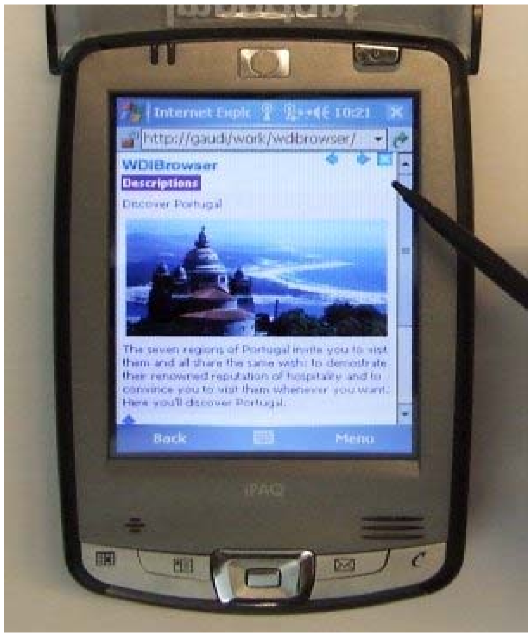
The seven regions of Portugal invite you to visit them and all share the same wish: to demostrate their renowned reputation of hospitality and to conv...More about

Fig.1 Distributed DI Browser on PC

The adoption of a Web-oriented approach, allows the Distributed DI Browser and thus the MPEG-21 DIDs, to be accessible on different terminal devices through the use of commons Web browsers (e.g. IE, Mozilla Firefox, Netscape, Opera, IE Mobile, MiniOpera, native Symbian browser, etc.) Fig. 1 and Fig.2 provide screenshots of the Distributed DI Browser used with different Web browsers in two distinct terminal devices.

3.2. Distributed Architecture and Web Service Approach for System Inter-operability

As referred above, one major goal to achieve was interoperability across formats and systems, allowing the end-user to access any kind of content using any type of terminal. Because the presentation of MPEG-21 DIDs to the end user involved a significantly complex processing step to produce a friendly graphical presentation to the user, it was decided to adopt a distributed approach to de-couple the processing tasks (time and power consuming) from the presentation and rendering tasks. It was thus decided to implement two distinct modules, client and server, communicating via a well defined Web Services interface using SOAP (Simple Object Access Protocol). In rich, powerful devices, both modules can be installed there, whereas for thin clients with limited processing capability, the server side can be installed in a remote machine, or even both the client and the server. All communication to and from the Distributed DI Browser modules is done using SOAP over HTTP or simply HTTP.



Discover Portugal

The seven regions of Portugal invite you to visit them and all share the same wish: to demostrate their renowned reputation of hospitality and to convince you to visit them whenever you want. Here you'll discover Portugal.

Fig. 2 Distributed DI Browser on PDA

Accordingly, the Distributed DI Browser was designed and implemented adopting client-server architecture and using Web services technology to establish the communication. Web Services concepts were applied for increasing the interoperability and portability to heterogeneous terminal devices. This architecture delegates on the server sub-system the responsibility of DI processing, leaving the client sub-system with the content presentation task. It thus provides a clear separation between the Digital Item processing tasks and the operations related to its presentation to the user. The server application of Distributed DI Browser is named *Digital Item Processing Server* (DIP Server) and the client sub-system is generically designated as *Graphic Digital Item Renderer* (GDI Renderer). This distributed architecture is represented in Fig. 3.

Client and server sub-systems communicate via a well defined Web Services interface. This distributed approach has the advantage of enabling the browsing of complex Digital Items also on "thin" devices (such PDAs and mobile phones) as it moves all the processing heavy tasks to the server side. The same processing core can be provided to various terminal devices, from PC platforms up to mobile phones and PDAs devices.

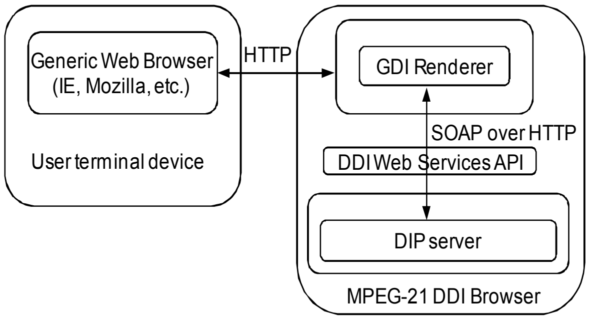


Fig. 3 Distributed DI Browser Architecture

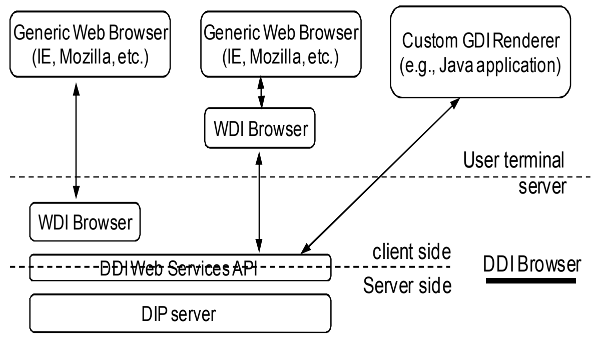


Fig. 4 Possible Alternatives for GDI Architecture

The clear separation that was done between GUI generation and the processing, thus allows using the same processing module (DIP server) with different GDI Renderers to suit the requirements of various terminal devices. Nevertheless, it is considered that the Web-oriented approach adopted for the GDI Renderer, by which the graphical representation of the DID is a set of HTML pages, is already able to meet the requirements of any type of terminal, as long as that terminal has a Web browser installed. The implementation of the GDI Renderer as a Web application, designated as Web DI Browser (WDI Browser), is illustrated in Fig. 4 for deployment alternatives and other possible solutions.

The WDI Browser can be deployed on the same Web server as the DIP Server sub-system or on a separate Web server. Furthermore, the GDI Renderer may be developed either as an Applet or as a plug-in to a generic Web browser. Other possible implementations of the GDI Renderer module could be for example a custom Java client application installed on the user terminal, using the services of the DIP Server module installed at the server side via the Web Services API. These possible different configurations are shown in Fig. 4.

As a summary, the Distributed Digital Item Browser provides the following functionalist:

- Browsing of MPEG-21 Digital Items stored on a remote repository – the user is able to search for, access and consume complex DIs, possibly composed of heterogeneous media resources, stored on a remote repository. The browsing of the contents of the DIs is made on a Web pages style.
- Download of Digital Items – the requested Digital Item is transparently downloaded from a remote repository in order to be processed and browsed;
- Validation of Digital Items – before being presented to the user, the Digital Items must be validated to ensure that they are conformant with Part 2 (DID) of the MPEG-21 standard;
- Choices processing – the system offers static User interaction with the content of the DI by supporting the corresponding mechanism specified in part 2 of MPEG-21. This mechanism, which exists in the *Choice* element of the DID, addresses User's selections;

Presentation of audiovisual resources - image, video and audio resources referenced in the DID, are reproduced on the terminal within the Web browser using embedded players plug-ins invoked by the DDI Browser (e.g. Windows Media Player, QuickTime Player.

4. Multimedia Terminal Middleware

Distributed Digital Item Browser needs to be coordinated under a terminal middleware together with other modules to achieve fruitful user experiences. These modules include but not limit to a media decoder to decode compressed audio-visual streams, a Usage environment Description (UED) - Usage Constraints Description (UCD) module to describe the user environments and user preferences, some Quality of Service (QoS) probes to provide sufficient feedback to content adaptation servers or intermediate services, for example, the Adaptation Decision Taking Engine (ADTE) and key management systems for intellectual property management and protection. We can consider the Terminal Middleware as heart of the terminal and name it Terminal Device Manager (TDM).

In principal, Terminal Middleware has two responsibilities: 1) externally interacting with server or intermediate service, e.g. to feedback for QoS control for content adaptation or to contact with DIBrowser@Server for content browsing; and 2) locally coordinating the convergence of DIBrowser@Client, Media Player, UED-UCD, QoS Probes. In this paper, we focus on the inter-operability problem and thus discuss only the terminal middleware support for the Distributed Digital Item Browser.

4.1. Client-Server Distributed Architecture

As the Distributed Digital Item Browser, TDM also adopts client-server distributed architecture in which a TDM component is deployed at the adaptation or server side, we can call it TDM@Server and multiple terminal components are deployed at client side, shortly TDM@Client. As common client-server architecture, TDM@client initially gets registered at a TDM@server and sends user environment parameters from the UED and UCD modules to TDM@server. TDM@server assigns a Terminal ID to TDM@client for identity.

4.2. Client-Server Distributed Architecture

The Terminal Middleware coordinates DIBrowser at Server and Client sides for content search, selection and play functions. The TDM@Client first sets up DIBrowser with Terminal ID and TDM@Server address.

Sequentially, 1) SearchDI: the user initiates a "SearchDI" request with DIBrowser@Client. Such a request passes through DIBrowser@Server and TDM@Server to a service provider that offers a search service via a Search Server module; the search engine of the Search Manager, looks into an MPEG-21 relational data base to find relevant results; the Search Manager then creates a DID with the list of DIDs stored in the MPEG-21 repository that matched the user query (identified by the search engine) and returns this DID to the DIBrowser@Server; the DIBrowser@Server processes the elements of that DID, generating a set of objects; these objects are used by the DIBrowser@Client to generate the graphical presentation of the DID elements to be rendered in the user terminal through a normal Web browser. 2) SelectDI: after navigation through the DID with the list of results and browsing some more appealing DIDs, the user finally selects the content to consume; at that point the user chooses a specific DI through the DIBrowser@Client; this "SelectDI" request is received by the DIBrowser@Server, which sends information to identify the requested DI, the terminal and the quality chosen by the user to a TDM@Server, the TDM@Server adds UED and the location of the TDM@Client and sends to the service provider, which returns the identity of the service and content DID. 3) PlayDI: requested to play DI, the DIBrowser@Server sends the identity of the service returned at "Select DI" step to TDM@Server, the TDM@Server adds UED and the location of this TDM@Client and sends to the service provider, which returns the URL of DI as result.

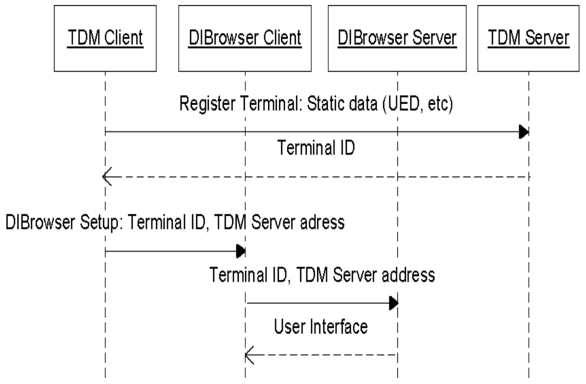

TDM Client
DIBrowser Client
DIBrowser Server
TDM Server
Register Terminal: Static data (UED, etc)
Terminal ID
DIBrowser Setup: Terminal ID, TDM Server adress
Terminal ID, TDM Server address
User Interface

Fig. 5 Initial Set up

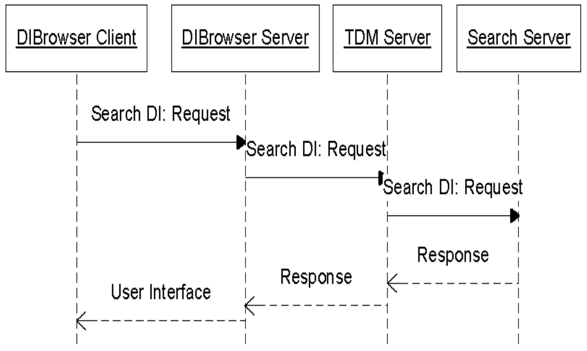


Fig. 6 Search Digital Item

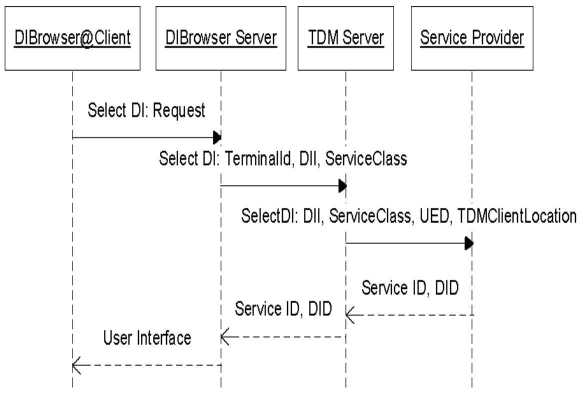


Fig. 7 Select Digital Item

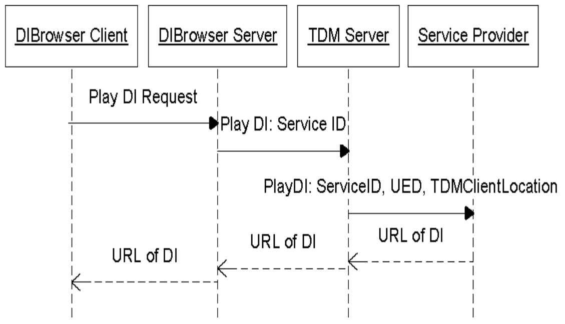


Fig. 8 Play Digital Item

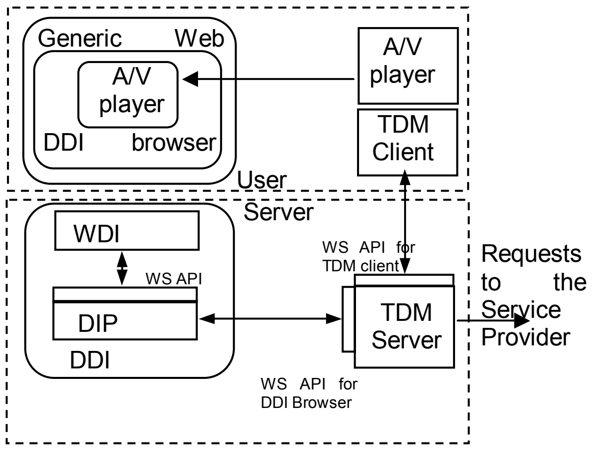


Fig. 9 Distributed DI Browser integrated with TDM

5. Implementation

This section presents the implementation details. The Client-Server architecture of Distributed DI Browser and terminal middleware has been shown in Fig. 9. To have a focus, we only abstract the most functional part and highlight them in a block schema. The entire multimedia terminal has been being designed and developed on Windows, Windows Mobile and Symbian platforms in order to maximize the portability to various platforms. Such implementation exploits the requirements of heterogeneity and inter-operability. As a result, it allows the terminal to be easily installed on multiple devices, such as TV, Set-Top-Boxes, PCs, PDAs and mobile phones.

6. Conclusion

This paper presents a multimedia terminal supporting inter-operability for content delivery over heterogeneous networks and devices. This terminal includes a MPEG-21 compliant digital item browser which provides presentation, navigation and interaction with MPEG-21 Digital Item Declaration (MPEG-21 DID) to support universal media access. We also investigate the architecture design in a distributed and Web Service approach to achieve system portability flexibility for a higher degree of inter-operability.

10. References

[1] A. Vetro and C. Christopoulos and T. Ebrahimi, “Special Issue on Universal Multimedia Access”, IEEE Signal Processing Magazine, Vol.20 (2), Mar. 2003
[2] ISO/IEC 21000, "Information Technology - Multimedia Framework (MPEG-21)", 2002
[3] ISO/IEC TR 21000-2:2003 "Information Technology - Multimedia Framework (MPEG-21)-Part2: Digital Item Declaration", 2003
[4] F. Pereira and J. R. Smith and A. Vetro, Special Issue on MPEG-21, IEEE Transaction on Multimedia, Vol.7 (3), Jun. 2005
[5] F. Keukelaere, W. Neve, P. Lambert, B. Rogge, and R. Walle, “MPEG-21 Digital Item Processing Architecture”, Proceedings of Euromedia, 2003.
[6]http://mpeg-21.itec.uni-klu.ac.at/cocoon/mpeg21/mpeg21Demo.xml.
[7] S. Lauf, and I. Burnett, “Implementation of a Mobile MPEG-21 Peer”. ACM Multimedia Conference (MM’05), Singapore, 2005.
[8]http://www.enikos.com/
[9]http://www.adactus.no/
[10]G. Ciobanu, M. Andrade, P. Carvalho, E. Carrapatoso, An MPEG-21 Web Peer for the consumption of Digital Items", Proc. CISTI 2007, June, 2007