CAIN-21: Automatic adaptation decisions and extensibility in an MPEG-21 adaptation engine

Fernando López[1,2], José M. Martínez[2], Narciso García[3]

[1] VINTEC, Universidad Internacional de la Rioja, Spain, *fernando.lopez@unir.net*
[2] VPULab, EPS - Universidad Autónoma de Madrid, Spain, *{f.lopez, josem.martinez}@uam.es*
[3] GTI, ETSIT - Universidad Politécnica de Madrid, Spain, *narciso@gti.ssr.upm.es*

***Abstract* — This paper presents the progress and final state of CAIN-21, an extensible and metadata driven multimedia adaptation in the MPEG-21 framework. CAIN-21 facilitates the integration of pluggable multimedia adaptation tools, automatically chooses the chain of adaptations to perform and manages its execution. To drive the adaptation, it uses the description tools and implied ontology established by MPEG-21. The paper not only describes the evolution and latest version of CAIN-21, but also identifies limitations and ambiguities in the description capabilities of MPEG-21. Therefore, it proposes some extensions to the MPEG-21 description schema for removing these problems. Finally, the pros and cons of CAIN-21 with respect to other multimedia adaptation engines are discussed.**

***Keywords* — ontology, multimedia, adaptation, decision, mpeg-7, mpeg-21**

I. Introduction

As time goes by, the variety of multimedia formats and devices has significantly increased, and still does. Multimedia content providers need to distribute their photos, videos and audio to a wide-range of devices and independently of the underlying delivery technology. *User-centric adaptation* [1] places the user in the centre of multimedia services and is also referred as *Universal Multimedia Experiences* (UME) [2].

The MPEG-21 standard [3] addresses the construction of a general multimedia framework that is consistent with the idea of UME. The MPEG-21 description tools enable the representation of a large set of concepts and relationships. MPEG-21 relies on the XML Schema to define the structure of the content and define an *implied ontology* in the text of the standard. Parts of the standards have been extended with description languages with a higher level of expressiveness. Particularly, the *explicit ontology* is represented using semantic description languages such as OWL (Web Ontology Language). The multimedia research com-munity has frequently accepted and used this MPEG-21 (pseudo)-ontology.

This paper compiles the evolution and final state of an adaptation engine named CAIN-21 [4] (Content Adaptation INtegrator in the MPEG-21 framework)[1]. The main purpose of CAIN-21 is to automate interoperability among multimedia formats and systems. Interoperability is implemented by means of an extensibility mechanism. With this mechanism, pluggable software tools are incorporated to progressively address wider ranges of adaptations. CAIN-21 automates interoperability by incorporating a decision mechanism for multimedia adaptation. This mechanism selects the adaptation tools and parameters that have to be executed to adapt multimedia. Furthermore, CAIN-21 exploits multi-step adaptation. Multi-step adaptation enables the combination and execution in several steps of the pluggable adaptation tools. With multi-step adaptations the range of feasible adaptations that can be achieved increases.

CAIN-21 also aims to provide a framework in which multimedia adaptation tools can be integrated and tested. The representation of the multimedia elements has to be formalized in order to make these tests[2] repeatable. To represent the multimedia elements of the tests, a set of MPEG-21 description tools have been selected. Currently, MPEG-21 is the most comprehensive multimedia description standard for the deployment of multimedia applications/systems. However, in practice description standards never cover 100% of the concepts. In the case of CAIN-21, we have encountered some difficulties using the MPEG-21 description elements. These difficulties were solved extending the description tools and implicit ontology that MPEG-21 provides. After presenting CAIN-21 architecture, this paper discusses these issues. We consider helpful to highlight it for people involved in the construction of multimedia adaptation systems, especially if they are determined to provide MPEG-21 interfaces to their users. The clarification of these problems may also be useful for people who intend further interaction with other non-MPEG-21 compliant multimedia systems.

[1] The *CAIN-21 software* together with a *CAIN-21 demo* are publicly available at *http://cain21.sourceforge.net*

[2] This paper demonstrates our proposal with an empirical study. We use the term *test* (instead of *experiment*) to indicate that its execution always yields the same results.

DOI: 10.9781/ijimai.2013.241

The publication in [4] summarizes the interfaces, the architecture of CAIN-21, and the evolution from Early CAIN to CAIN-21, and provides a preliminary comparison with other adaptation engines. Now that CAIN-21 has reached a stable and mature state, this publication supersedes [4] by providing an extended, comprehensive and updated description of CAIN-21.

In particular, this new publication describes the delivery and adaptation methods used in CAIN-21 as well as the binding modes. The publication also describes and provides usage examples of the *ConversionCapabilities* and *ConversionCapabilities* description tools, incorporates the properties relationships, the KISS principle behind this design, the use of composed properties to address complicated relationships, and proposes new ideas, such as the distinction between implied and explicit ontologies and the advantages of considering MPEG-21 as a simple (pseudo)-ontology. The updated multimedia adaptation engines comparison in Section VI adds *ConversionLink* to the comparison, adds new aspects to the comparison (i.e., multistep, extensibility and semantic adaptation), discusses the reasoning behind the different approaches taken over the years and the pros and cons of the different decision methods. Finally, this publication appends several tests that illustrate the multimedia adaptation method proposed in this paper, and justifies the need for the proposed extensions to the MPEG-21 standard.

In the rest of this paper, Section II reviews the state of the art concerning semantic web description and the description tools that MPEG-21 provides for multimedia adaptation. It also introduces some automatic multimedia adaptation techniques. Section III describes the main features and elements of CAIN-21. Section IV offers innovative description tools that fill the description gaps identified in the standard and justifies their usefulness. Section V provides a set of tests that demonstrate and validate these extensions. Section VI provides a comparative analysis between CAIN-21 and other multimedia adaptation engines. Finally, Section VII gathers the innovations and advantages of the adaptation techniques explained in the paper and it provides some conclusions.

II. STATE OF THE ART

A. Semantic web for multimedia

The Semantic Web [5] aims to represent knowledge in a format that can be automatically processed without human intervention. For this purpose the machine must be capable of understanding the concepts and relationships thereby described. The *Semantic Web Stack* [5] defines a stack of languages in which each layer uses the description capabilities of the layer below it to provide a higher level of expressiveness. In this stack, the technologies up to RDF, OWL and SPARQL have been standardized and accepted. The term *ontology* is used to refer to the concepts (usually defined with a formal *vocabulary*) and relationships in a specific domain. This ontology is frequently represented with OWL creating a *semantic graph*. The technologies in the top of the stack use the semantic graph to infer additional knowledge. Currently, it is not clear how to implement the technologies on the top of the stack. Automatic reasoning has been frequently proposed to infer this additional knowledge. However, the results of these top-level technologies are still limited to achieve the ultimate aim of the Semantic Web: the sharing, processing and understanding of data by automatic systems in the same manner that people can do.

To build a multimedia system that automatically manages and understand multimedia content, it is crucial to define the ontology of its multimedia concepts: Fig. 1 depicts this idea. Bold lines represent better levels of understanding. The figure shows that the user is capable of understanding the meaning of the media, but has more difficulties reading the description of the content. For instance, it is easier for the user to identify a dog in a picture than to interpret its MPEG-7 description [6]. On the other hand, the computer can extract information from metadata more easily than it can analyse the corresponding media resource.

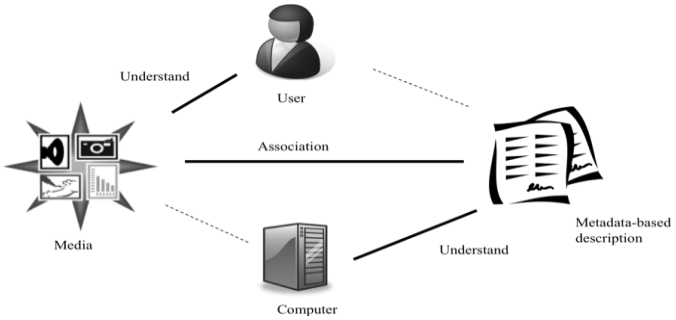

Fig. 1: Semantic description of multimedia content

In the field of multimedia, two widely accepted implied ontologies are the MPEG-7 [6] standard for the media content and MPEG-21 [3] for the whole multimedia system. These standards make use of metadata to achieve a better understanding of multimedia. Particularly, these standards propose several vocabularies to represent a detailed description of the meaning of the multimedia elements. MPEG-7 and MPEG-21 tend to define the external interface of the multimedia systems and leave the algorithms that implement it (e.g. reasoning) to the industry and research community. The W3C Consortium has also initiated a project to represent multimedia ontology called *Multimedia Vocabularies on the Semantic Web* [7]. Even although this standard fully exploits OWL, which has a higher level of expressiveness and ability to represent knowledge, at time of writing, the majority of multimedia research relies on MPEG-7 and MPEG-21.

B. MPEG-21

This section reviews the state of the art for the MPEG-21 description tools to which this paper contributes. A complete description for the MPEG-21 standard can be found in [3].

1) Content description and conditional elements

The notion of *Digital Item* (DI) is a fundamental concept within MPEG-21. A DI is a general representation for any multimedia element. This element can represent both the multimedia content and the multimedia context. MPEG-21 Part 2 [3] standardises the representation of a DI in the case of multimedia content. A DI may contain one or more *Component*[3] elements. Each *Component* includes one *Resource* element and zero or more *Descriptor* elements. The *Resource* element references the media and the *Descriptor* element provides metadata for this media. The MPEG-21 allows optional, alternative and conditional elements. For the purposes of this paper we are only going to describe conditional elements. A *conditional element* is an element of the DI that appears only when certain conditions are true. Certain elements of the DI are configurable, i.e., their content varies depending on the value of *Predicate* elements. A *Predicate* element can take the values *true, false* or *undecided*. The *Choice* element enables a "menu". The options of this menu are provided through *Selection* elements. The *Selection* elements are used to define in runtime the values of the *Predicate* elements. The MPEG-21 standard does not define how the values of the *Predicate* elements are obtained. These values can be asked to the user or automatically decided by the multimedia system. The value of some *Predicates* can be even unknown in runtime, in which case they take the *undecided* value. A *Condition* is a conjunction (and operator) of one or more predicates. The *Condition* elements are used to specify which elements of the DI are valid in runtime. Only the elements for which the *Condition* is true are considered part of the DI. For instance, several *Component* elements may contain a *Condition* element. In runtime, only the *Component* whose *Condition* is true is considered part of the DI.

2) MPEG-21 adaptation tools

MPEG-21 Part 7 [3] has defined a set of *description tools* (or merely *tools*) for multimedia adaptation. These tools do not specify how the adaptation has to be performed; they only gather the information necessary for adapting a DI. These tools are collectively referred as *Digital Item Adaptation* (DIA) *tools*. The instances of these tools are referred as DIA *descriptions* (or merely *descriptions*). This section reviews the *Usage Environment Description* (UED) tools, the *DIA Configuration* tools and the *ConversionLink* tools.

3) UED tools

These tools enable the description of the terminal capabilities, the network constraints, the user's characteristics, preferences and natural environment. The term *usage environment description* (or merely *usage environment*) refers to an instance of one or more *UED* tools. Further description of the *UED* tools can be found in [3].

[3] MPEG-21 capitalises and italicises XML description tools. This paper adopts this rule.

4) DIA Configuration tools

The DI author can use the *DIA Configuration* tools to recommend how to adapt the content to the usage environment. Specifically, the *DIA Configuration* tools include two tools to drive the adaptation. The first tool allows the DI author to indicate how to obtain the options of the *Choice* conditional mechanism explained above. For this tool, the standard defines only two values: *UserSelection* indicates that the selection has to be done by the user. *BackgroundConfiguration* indicates that the system has to automatically perform this decision.

The second tool is the *SuggestedDIADescription*. The DI author uses this tool to point out which parts of the DI or DIA descriptions have to be used to decide the adaptation. Specifically, XPath [8] expressions are used to provide this information. For instance, the DI author may recommend using the *Format* element of the *VideoCapabilitiesType* in the *UED* to make the adaptation decision. A further description of these tools can be found in [3].

5) ConversionLink tools

The *ConversionLink* tools appear in [9] to complement the *BSDLink* tools. The *ConversionLink* tools are intended to address generic adaptation (e.g. transcoding, transmoding, summarization) whereas the *BSDLink* are intended for scalable bitstream adaptation. The MPEG-21 standard defines a *conversion* as a processor (software or hardware) that changes the characteristics of a *Resource* or of its corresponding *Descriptor* elements. The *ConversionLink* tools include the *ConversionCapabilitiesType* tool. This tool expresses the types of conversions that a terminal is capable of performing. The content of this tool is not standardised, instead, it provides a derivation-by-extension mechanism allowing the inclusion of conversion descriptions. A further description of these tools can also be found in [10].

C. Multimedia adaptation-decision making methods

Typically, multimedia adaptation is performed in two phases, which usually execute in a sequential manner [11][13][14][15]. Firstly, a *decision phase* is used to evaluate which adaptations best suits the constraints of the usage environment. Secondly, in the *execution phase*, these conversions are performed on the media and metadata conveyed in the DI. For the decision phase, two different methods have been widely investigated in the literature:

1) Quality-based methods [11][12][13] (also referred as *optimisation-based methods*) aim at finding the adaptation parameters that maximise the quality (also referred to as *utility*) resulting from the adaptation to the constraints of the usage environment. These methods operate by solving an optimisation problem in the Pareto frontier. Frequently, the MPEG-21 Part-7 DIA tools have been used to point out these relationships between the adaptation parameters and corresponding utilities.

2) Knowledge-based methods [14][15][16] have been used primarily to determine whether a conversion can be executed

and which parameters must be supplied to adapt the content. These methods usually consider the concatenation of several conversions in a sequence. They have also been referred as multi-step adaptation.

CAIN-21 (described in Section III) combines both methods in sequence. Firstly, the knowledge-based methods use the media format to decide which conversions have to be carried out in order to adapt the content to the usage environment. This method is further explained in [17]. Secondly, certain "intelligent" conversion tools incorporate the capability to select the parameters that optimise their output. The quality-based methods that CAIN-21 incorporates are demonstrated in [18].

D. Related multimedia adaptation engines

This subsection introduces related multimedia adaptation engines. Section VI compares these adaptation engines with CAIN-21.

Mariam [10] has studied the applicability of a standard *AdaptationQoS* description tool to drive general (scalable and non-scalable) resource adaptation. This investigation concludes developing the ConversionLink[4] adaptation engine togueter with the *ConversionLink* description tool. The *ConversionLink* tool was later standardized in [9]. This tool has already been described in Subsection II.B.

Debargha et al. [11] explained the basis of the *AdaptationQoS* description tool and its usage. Christian et al. [12] builds on this description tool to implement the idea of coded-independent resource adaptation for scalable resources. To this end, they have researched the *BSDLink* tools (introduced in Subsection II.B). In [12] they explains the use of the notion of Pareto optimality and multi-attribute optimisation to identify the scalable layers that best suit the terminal constraints [16].

Jannach et al. [15] developed the koMMa framework in order to demonstrate the use of Artificial Intelligence planning in multistep multimedia adaptation. They exploited Semantic Web Services to address interoperability. They also proposed an extensibility mechanism by means of pluggable Web Services.

Anastasis et al. describe the DCAF adaptation engine in [20]. This research showed how to use heuristic genetic algorithms to identify the parameters of the *AdpatationQoS* description tool. The *UED* and *UCD* description tools are used to represent the context of the adaptation. The notion of Pareto optimality is also introduced to rank the possible decisions.

Davy et al. [21] have built on the aforementioned *AdaptationQoS*, *BSD* and *UED* description tools to develop the NinSuna adaptation engine. This engine provides both coding-format independence and packaging-format independence. The major innovation of this engine is leveraging Semantic Web technologies to accomplish semantic adaptation decisions. The semantics are explicitly represented with RDF tuples and in this way they introduce formal semantics in the exiting MPEG-21 adaptation description tools.

[4] Note that the symbol ConversionLink is not italicized to refer to the adaptation engine. However, it is italicized to refer to the description tool.

E. Delivery and adaptation methods

From the standpoint of the media client, there are two main media *delivery models* [22]: *download*, where the client starts to play the media content after completely receiving the media from the server, and *streaming* where media content is played while data reception is in progress. *Streaming servers* usually cover two methods to deliver video to the users:

1) Live video. Broadcast of live events in real time. This streaming is useful when the client expects to receive video as soon as it is available. Live events, video conferencing, and surveillance systems are commonly streamed over the Internet as they happen with the assistance of broadcasting software. The video recording software encodes a live source (video or audio) in real time and transfers the resulting media to the streaming server. The streaming server then serves, or "reflects", the live stream to clients. Regardless of when different customers connect to the stream, each sees the same point in the stream at the same time.

2) Video On Demand (VOD). Each customer initiates the reception of the media from the beginning, so no customer ever comes in "late" to the stream. For instance, this mode can be used to distribute movies to users who play those movies at different times.

According to the moment at which the adaptation takes place; media adaptation can be divided into three *adaptation modes*:

1) Offline Adaptation mode (*OffA mode*). The adaptation is performed in the background and before the media is available to the user. This mode is adequate for on demand media delivery. However, this mode is not suitable for live video because the user is expecting to watch the video event as soon as it occurs. This adaptation requires previous knowledge of the feasible terminal capabilities and network bandwidth. The media can be prepared for several terminals of network capacities. The main limitation of the OffA mode is that the user's preferences and natural environment constraints are not taken into account. These parameters are unknown when the media repository is created. While creating a repository of adapted resources for each user's profile is possible, it is unmanageable from a practical point of view when the number of user's profiles increases.

2) On Demand Adaptation mode (*OdA mode*). Adaptation takes place at the same time that the user asks for the resource. In this mode, the client's characteristics, preferences and natural environment can be taken into account. However, if the resource adaptation process is time consuming, the user has to wait until the whole resource is adapted. Therefore, this adaptation results useful for small resources (e.g. images), but can become unacceptable for long resources (e.g. video or speech).

3) Online Adaptation Mode (*OnA mode*). As with the OdA mode, user's characteristics, preferences and natural

environment can be taken into account. In this mode also the adaptation begins as soon as the user asks for the resource. However, in contrast to the OdA mode, in the OnA mode the resource begins to be delivered to the user before the whole resource has been adapted. This adaptation is appropriate for long resources (and perhaps also for small resources). The drawback of this approach is that, in general, implementing this solution efficiently is difficult. In OnA mode we need to ensure that media data fragments are delivered to the client in time to maintain playback continuity. The advantage is that once implemented, the OnA mode can be reutilized to simulate the OffA and OdA modes.

III. CAIN-21: SYSTEM DESCRIPTION

This section sequentially describes the CAIN-21 software interfaces, the architecture and the control flow. Section IV builds on this section to specify the description tools that CAIN-21 utilizes and justifies their extension.

A. Software interfaces

CAIN-21 serves adaptation requests through two external software interfaces (see Fig. 2 below): (1) The *media level transcoding interface* performs *blind adaptation* (i.e. semantic-less adaptation) of a media resource. In addition to the media level, this interface can also perform system level adaptation, i.e., videos composed of one or more audio and visual streams. The media level transcoding operations are implemented in the *Tlib* module. This module includes conventional software libraries such as *ffmpeg*, *imagemagick* as well as Java Native Interface (JNI) custom libraries. (2) The *DI level adaptation interface* is in charge of performing system level (semantic or blind) adaptations. In this case metadata is used during the adaptation.

The DI level adaptation interface complies with the MPEG-21 representation schema. The *Content DI* conveys the media resource together with its metadata to be adapted. To drive the adaptation, CAIN-21 uses four *DIA* description tools. Only the *Content DI* and *DIA* description tools follow fully the MPEG-21 recommendations. For the point of view of these interfaces, CAIN-21 is a replaceable black box. Fig. 2 provides a view of CAIN-21 consistent with the idea of an adaptation engine that the MPEG-21 Part-7 framework proposes.

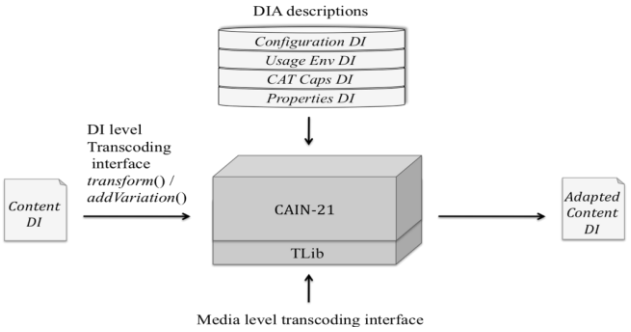

Fig. 2: Software interfaces of CAIN-21

In CAIN-21, metadata-based adaptation [23] is performed through the DI level interface and at the *Component* level. An MPEG-21 *Component* includes a media resource (in the *Resource* element) and its metadata (in the *Descriptor* element). The *Descriptor* elements use MPEG-7 Part 3, Part 4 and Part 5 [6] to describe the multimedia content. The DI level adaptation interface provides two different operations. The first one modifies the existing *Component* and the second operation adds a new *Component* element to the DI. More specifically: (1) the *transform*() operation takes a *Component* from the *Content DI* and modifies its media resource and metadata in order to adapt it to the usage environment; (2) the *addVariation*() operation takes a *Component* from the *Content DI* and creates a new *Component* ready to be consumed in the usage environment. At the end of this adaptation, CAIN-21 adds this adapted *Component* to the *Content DI*.

B. Architecture

This section provides a detailed description of the CAIN-21's modules. Fig. 3 depicts CAIN-21's functional modules and the control flow along the adaptation process. The rest of this subsection explains the modules and description tools in the figure.

1) Adaptation Management Module (AMM)

The AMM is responsible for coordinating the entire DI level adaptation process. Modules below the AMM perform different tasks initiated by the AMM.

2) Adaptation Decision and Execution Modules (ADM and AEM)

Subsection 8.C explained that frequently adaptation engines divide the decision and the execution into two different phases. Firstly, a decision phase is used to decide which adaptation best suits the constraints of the usage environment. Secondly, in the execution phase, these adaptation actions are performed on the media conveyed in the DIs. CAIN-21 also includes this distinction implemented in the *Adaptation Decision Module* (ADM) and the *Adaptation Execution Module* (AEM), respectively.

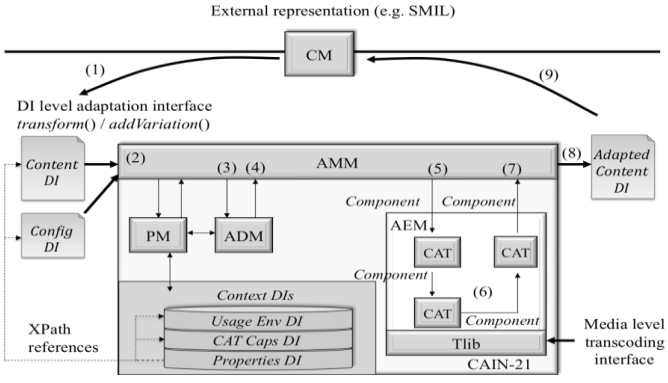

Fig. 3: Modules and control flow within CAIN-21

3) Conversions and Component Adaptation Tools (CATs)

As explained in Subsection II.B, MPEG-21 Part-7 defines a

conversion as the process that changes the characteristics of a resource. In general, a conversion performs the act as defined by the MPEG-21 Part-6 term *adapt*. In CAIN-21, a *Component Adaptation Tools* (CATs) is a pluggable software module that implements one or more conversions. Multi-step adaptation allows for the sequential execution of the conversions implemented in one or more CATs. The ADM uses metadata to determine the sequence of conversions and parameters that should be executed over a *Component* element of the *Content DI*. Subsequently, the AEM executes such sequence of CATs on the original *Component*. When a CAT is executed, both the conversion to execute and the parameters of the conversion have to be provided. If CAIN-21 receives multiple requests to adapt the same content to the same usage environment, a caching mechanism speeds up this process by bypassing the execution of the Planner and Executer several times.

During their execution, CATs have the option of appending information to the *Descriptor* element of the *Component* so that subsequently CATs can use it. We use the name *static decisions* to refer to metadata-based decisions. Static decisions do not depend on the resource content (only the *Descriptor*) and the ADM is responsible for these decisions. On the contrary, we use the term *dynamic decisions* to refer to adaptation decisions that perform operations over the resource content. Dynamic decisions cannot be taken until the resource is available and the CATs take them. These dynamic decisions usually correspond to semantic decisions or quality-based decisions. Frequently semantic decisions assume particular content (e.g. faces, soccer, news items, violent scenes in the movie). For example, in [19] we assume the existence of faces in the images. Quality-based decision methods have been described in Subsection II.C and demonstrated in [18].

4) Context Repository

As further described in Subsection IV.A, CAIN-21 defines a type of DI referred to as *Context DIs.* These *Context DI* elements store *DIA* descriptions with information concerning the context in which the adaptation takes place.

The *Context Repository* in Fig. 3 includes the three *Context DIs*. The *Usage Environment DI* describes the available usage environments using several MPEG-21 *UED* elements (i.e. instances of the *UED* tools). Each *CAT Capabilities DI* describes the different *conversions* that a CAT is able to perform. Each conversion has a set of valid input and output properties along with their corresponding values. The relationships among these elements are described in more detail in Subsection IV.C.

CAIN-21 includes an addressing mechanism in which changes in the metadata descriptors will not imply changes in the underlying source code. This mechanism is described in detail in Subsection IV.E. The mechanism represents all the multimedia information by means of *properties*. Each property has one *key* and one or more *values*. The advantage of this representation is that it suits the decision mechanism that we have developed for CAIN-21 [17]. The *Properties DI* is intended to store a set of keys and corresponding *xpointer*() [24] expressions providing access to the actual values. In Fig. 3, dashed arrows indicate that the *xpointer*() expressions in the *Properties DI* are stored in the other DIs.

5) Configuration DI

The *Configuration DI* is a *DIA* description indicating which description of the terminal, network and user – from the ones available in the *Usage Environment DI* – to use during a adaptation request. Subsection II.B explained that MPEG-21 recommends using the *Choice* descriptor and *DIA Configuration* description tool to specify the adaptation to perform. CAIN-21 does not use this standard mechanism; instead it uses the *Configuration DI* to indicate the parameters of the adaptation to perform. Subsection IV.A justifies this change and explains the advantages that this proposal yields.

6) Parsing Module (PM)

The PM is responsible for resolving the values of the aforementioned properties. Firstly, the PM accesses the *Properties DI* to obtain the set of property keys and corresponding *xpointer*() expressions. Secondly, after resolving these expressions, the values of these properties are generated. During this step, the rest of the metadata is loaded from the *Content DI*, *Configuration DI*, *Usage Environment DI* and *CAT Capabilities DI*. After parsing the different DIs, all the metadata is represented as a set of properties. The value of these properties can be multi-valued (e.g. *bitrate* = [1000..200000], *audio_format* = {*aac*, *mp3*}).

7) Coupling Module (CM)

A wide range of multimedia representation standards exists to represent multimedia content (e.g. HTML, SMIL, NewsML, MPEG-4 BIFS). CAIN-21 can be integrated into heterogeneous multimedia systems that may be using external representation technology (i.e., non-MPEG-21 technology). The CM is the gateway that enables such integration. To this end, this module transforms the external representation of multimedia into an MPEG-21 compliant input *Content DI* that afterwards CAIN-21 processes. In addition, the CM is responsible for transforming the adapted output *Content DI* into its external representation. Instances of the CM are interchangeable modules created to interact with different external representations. In practice, there is a semantic gap during this interaction with the external multimedia description standards, i.e., a direct correspondence between the external descriptors and the MPEG-7/21 descriptors might not exist. To provide these additional meanings, MPEG-7 Part 5 offers a set of open *Classification Schemes* (CSs) [6], which indicates what these external descriptors mean.

C. Control flow

The numbers in Fig. 3 indicate the control flow of the tasks in the adaptation process. (1) When interacting with external systems, the CM transforms the external multimedia representation into a *Content DI* that CAIN-21 can process. (2)

The *Content DI* together with a *Configuration DI* arrives via the DI level interface *transform*() or *addVariation*() operations. (3) The AMM is in charge of coordinating the whole DI level adaptation process. Specifically, the AMM invokes in sequence the ADM and the AEM to (4) decide and (5) execute the corresponding adaptation on the original *Component*. (6) The CATs use the *TLib* services to adapt the media resource. The CATs might also change or append information to the *Descriptor* element of the *Component* so that the subsequent CATs may use it. (7) Once all the conversions of the sequence have been executed, (8) the AMM returns the adapted *Content DI* to the caller. (9) Frequently, the adapted *Content DI* may need to be transformed to an external representation and in this case, the CM performs this transformation.

IV. CAIN-21'S EXTENSIONS TO THE MPEG-21 SCHEMA

CAIN-21 uses the description tools that MPEG-21 standardises. The following subsections identify a set of limitations and ambiguities in the description capabilities of MPEG-21. They then propose some extensions to the MPEG-21 description schema. The additions are justified in order to remove these limitations and ambiguities. The following subsections also discus how these extensions make possible to address a new range of multimedia adaptation problems.

A. Content DI, Context DI and Configuration DI

Subsection II.B explained that in MPEG-21 framework different DIs are used throughout the consumption and delivery chain. The DIs can be classified according to their purpose. One initial approach in the literature has divided the DIs into *Content DI*s and *Context DI*s. The *Content DI* is a DI intended to carry out the multimedia resource and corresponding metadata. The *Context DI* is intended to contain a description of the usage environment. The notions of *Content DI* and *Content DI* have been considered by the MPEG-21 standard (see for instance [25]) although they have not been finally incorporated to the standard. However, some authors have informally used these notions in their systems [26][27].

Particularly, these authors have used the term *Context DI* only to reference the usage environment [25][26][27]. In [28], we proposed to extend the idea of *Context DI* to represent the context information. Particularly, in CAIN-21 there are three types of context elements: the *Usage Environment DI*, the *CAT Capabilities DIs* and the *Properties DI*. Subsection III.B described these elements.

Furthermore, CAIN-21 configures the adaptation using the *DIA Configuration* description tools (described in Subsection IIII.B). After an adaptation request, the *DIA Configuration* tools can be used to specify the target usage environment. Although there are scenarios in which the *DIA Configuration* tools is applicable, we have identified two limitations in the standard *DIA Configuration* mechanism:

1. The standard *Content DIs* uses the *Choice* description element to enclose alternative adaptation options, which depends on the available terminals. This produces a dependency between the *Content DI* (which contains the *Resource* and optionally a *DIA Configuration* description) and the *Usage Environment DI*. This dependency implies changing the *Content DI* whenever the *Usage Environment DI* is modified (e.g. one of the terminal descriptions is changed).

2. *DIA Configuration* assumes that the entire usage environment is known when the *Usage Environment DI* is created.

The idea of using three DIs avoids the first limitation:

1. The *Content DI* with the multimedia resource and corresponding metadata.

2. The *Context DI* that acts as a database where usage environment, adaptation capabilities and metadata properties under consideration are stored.

3. The *Configuration DI* that includes a *DIA Configuration* description.

The *Configuration DI* also solves the second limitation: the *Content DI* and the *Context DI* are created and stored in CAIN-21 during its development or deployment. The *Configuration DI* is dynamically created to provide to CAIN-21 information about the adaptation request to be performed.

Next section describes the *ARC* description tool that the *Configuration DI* conveys. The main aim of our proposal is that the *Content DI* will not be modified when the *Usage Environment DI* changes.

B. The ARC description tool

Section IIII.B described the two *DIA Configuration* description tools that MPEG-21 Part 7 standardises: (1) The *UserSelection*/*BackgroundConfiguration* elements indicate whether the DI *Choice*/*Selection* mechanism must be presented to the user or automatically decided by the system. (2) The DI's author uses the *SuggestedDIADescriptions* to suggest which *DIA Description* elements should be used for the adaptation. Both methods assume the existence of a negotiation mechanism. Authors such as [26][29] have followed this approach incorporating the *DIA Configuration* description in the DI to be consumed. CAIN-21 is not a network agent (as in the *DIA Configuration* usage model developed in [3]) but a middleware providing an API. Previous subsection introduces the problem of selecting zero or one instance of the standard MPEG-21 Part 7 *UED* description tools (i.e., *Terminal*, *Network* and *User*[5] elements) from the *Usage Environment DI*. If we relax the network agent negotiation assumption we can utilise the *DIA Configuration* to specify the particular usage environment. CAIN-21 extends the *DIA Configuration* to provide this information, i.e., it defines a third *DIA Configuration* tool (non-considered in MPEG-21). This extension is called *Adaptation Request Configuration* (*ARC*) tool. Consider, for instance, two terminals in the *Usage Environment DI*, a mobile terminal and a laptop terminal. In this case, an *ARC* description can be used

[5] Currently CAIN-21 does not consider the *NaturalEnvironment* description tool, but its inclusion would be a direct process.

to indicate the target terminal. The *Content DI* and the *Usage Environment DI* can be deployed before starting the adaptation engine. On the contrary, the *ARC* description is only created when an adaptation is going to be executed.

C. CAT Capabilities

The large quantity of multimedia adaptations that could be envisioned makes it unfeasible to implement all of them. Subsection III.B has introduced the notion of pluggable CATs. Their adaptation capabilities are described in *CAT Capabilities DIs* (also introduced in Subsection III.B). One CAT can be used as soon as this CAT and its corresponding *CAT Capabilities DI* are plugged in CAIN-21.

1) CAT Capabilities and Conversion Capabilities

The notion of *CAT Capabilities* was introduced in [28]. The following paragraphs describe the current *CAT Capabilities* description tool of CAIN-21 and compare it with the standard *ConversionLink* [9].

Subsection II.B explained that MPEG-21 Part 7 Amendment 1 defines a *conversion* as an (software or hardware) element capable of performing multimedia adaptation. The original *CAT Capabilities* only allowed describing one conversion. The final *CAT Capabilities* can incorporate several conversion elements. During the development of CAIN-21, we observed the practical fact that conversion capabilities are not always easy to describe with only one conversion. With some types of adaptations, we need to divide the capabilities of an individual *CAT Capabilities* element into several *Conversion Capabilities* elements. Consider, for example, a CAT that is capable of accepting JPEG and PNG images, but PNG images are accepted only in greyscale, whereas JPEG images are accepted in both colour and greyscale. In this case, the *CAT Capabilities DI* must be split into two separate *Conversion Capabilities*. The first *Conversion Capabilities* element states that PNG images are accepted in greyscale. The second *Conversion Capabilities* element states that JPEG images are accepted in both colour and greyscale.

The second major feature implies the description of the values that properties can take. In the CAIN-21 decision process, preconditions, postconditions and parameters can take several possible values (e.g. *format* = {*mpeg*-1, *mpeg*-2, *mpeg*-4}). We have modified the description of the conversions so that each input and output property can take multiple values.

```
<dia:DIA xmlns="urn:vpu:cain21-cat-capabilities"
         xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-
NS"
         xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
 <dia:Description xsi:type="CATCapabilitiesType"
id="video_transcoder_cat">

<CATClassName>es.vpu.cain21.cats.VideoTranscoderCA
T</CATClassName>
  <Platform>
   <ValueSet>
    <Value href="Windows XP">Windows</Value>
    <Value href="Linux">Linux</Value>
    <Value href="Mac OS X">Mac OS X</Value>
   </ValueSet>
   </Platform>
   <!-- Online MPEG conversion using the ffmpeg
library -->
   <ConversionCapability
xsi:type="ConversionCapabilityType"

id="online_mpeg_transcoder">
    <ContentDegradation>0</ContentDegradation>
    <ComputationalCost>1.0</ComputationalCost>
    <Preconditions>
     <URL>
      <AnyValue/>
     </URL>
     <Binding>
      <ValueSet>
       <Value href="urn:mpeg:mpeg21:2007:01-BBL-
NS:handler:HTTP">HTTP</Value>
       <Value href="urn:mpeg:mpeg21:2007:01-BBL-
NS:handler:FILE">FILE</Value>
      </ValueSet>
     </Binding>
     <Content>
      <ValueSet>
       <Value
href="urn:mpeg:mpeg7:cs:ContentCS:2001:2">Audiovis
ual</Value>
       <Value
href="urn:mpeg:mpeg7:cs:ContentCS:2001:4.2">Video<
/Value>
      </ValueSet>
     </Content>
     <FileFormat>
      ..............
     </FileFormat>
     <Bitrate>
      <RangeValueSet from="5000" to="1000000"/>
     </Bitrate>
     ................
    </Preconditions>
    <Postconditions>
     ..................
    </Postconditions>
   </ConversionCapability>
   <!-- On Demand MP4 conversion using the ffmpeg
command -->
   <ConversionCapability
xsi:type="ConversionCapabilityType"

id="ondemand_mp4_transcoder">
    ..................
    ...................
  </ConversionCapability>
 </dia:Description>
</dia:DIA>
```

Listing 1: CAT Capabilities DI example

The XML Schema of the *CAT Capabilities* description tool that we propose is available in the file *ccatc.xsd* of the CAIN-21 software. The *CATCapabilitiesType* represents a CAT. The *ConversionCapabilitiesType* represents each conversion that the CAT is capable of performing. Listing 1 shows a fragment of one of the *CAT Capabilities DIs* fully available in the CAIN-21 demo. The CAT comprises two *ConversionCapability* elements named *online_mpeg_transcoder* and *ondemand_mp4_transcoder*. The *Preconditions* and *Postconditions* elements contain information related to the media format that each conversion accepts and produces. These properties are inspired by MPEG-7 Part 5. Note that the properties can be single-valued or multi-valued by means of the *ValueSet* element. The

RangeValueSet element enables the description of ranges. The *AnyValue* element represents a placeholder whenever the value of the parameter must be provided, but every value is acceptable.

2) Comparison with the ConversionLink

Subsection II.B describes that MPEG-21 Part 7 Amendment 1 has standardised the *ConversionLink* description tool. This tool provides a means for linking steering description parameters and conversion capabilities description. *ConversionLink* uses the *ConversionCapability* element to describe the adaptation capabilities.

The *CATCapabilitiesType* of CAIN-21 is defined as a derivation by restriction of the *DIADescriptionType* of MPEG-21. Therefore, the *CATCapabilitiesType* can be seen as a (non-MPEG-21 standardised) *DIA* description tool.

More specifically, the *ConversionCapabilityType* of MPEG-21 is a generic container that used the following type to enable any description:

```
 <any                          namespace="##other"
processContents="lax" minOccurs="0"/>
```

The *ConversionCapabilityType* of CAIN-21 is defined as a derivation by extension of this *ConversionCapabilityType*. Therefore, the *ConversionCapabilityType* of CAIN-21 can be seen as an instance of the generic *ConversionCapabilityType* that MPEG-21 provides. In particular, CAIN-21 describes the conversions by means of preconditions and postconditions. This description model suits the automatic decision mechanism of CAIN-21.

Authors such as [10] also use the *ConversionCapability* element[6] together with the *ConversionLink*. In this case, the author makes use of RDF tuples to describe the adaptation capabilities and its semantics. CAIN-21 instead uses preconditions and postconditions that best suit its decision-making mechanism.

D. Binding modes

Subsection II.E explained the Offline/On-demand/Online adaptation modes. CAIN-21 supports all these modes. Subsection II.E has highlighted the difference between adaptation and delivery. Although CAIN-21 is focused on adaptation, delivery is supported to a certain extend. *Binding modes* have been introduced in CAIN-21 to support media delivery. In particular, delivery can be envisioned as a type of adaptation. The binding modes indicate the delivery mechanism that the conversion uses to receive and transmit the media (such as *FILE*, *HTTP* or *RSTP*). This work proposes to use the *mpeg21:Handler* description tool of the *Bitstream Binding Language* (BBL) [30]. The binding modes are used with two purposes: (1) to transfer the media between CATs in a sequence of CATs and (2) to transfer the media from the last CAT in the sequence to the consumption terminal. TABLE I shows the binding modes currently available in CAIN-21. The *INPROCESS* binding mode allows efficient transfer of the media resource between CATs.

In CAIN-21 each *ConversionCapabilities* element must provide in its preconditions and postconditions the available *binding modes*. For instance, in Listing 1, the first *ConversionCapabilities* element supports *FILE* and *HTTP* in its preconditions (i.e. in the input of the corresponding conversion). The *Terminal* element of the *Usage Environment DI* must also indicate the delivery modes that it supports to receive media. Listing 3 below shows how the binding modes of a terminal are provided into its terminal description. Listing 1 and Listing 3 show that the binding mode, of both the *online_mpeg_transcoder* and the terminal, can take more that one value. In these examples, both the conversion described in Listing 1 and the terminal described in Listing 3 support the *FILE* and *HTTP* binding modes.

The current release of CAIN-21 includes one CAT (named *HttpVideoStreamingCAT*), which only purpose is to provide HTTP video delivery. If necessary, the decision mechanism automatically adds this CAT to the sequence of CATs. Specifically, this CAT is added at the end of the sequence when the last CAT of the sequence does not provide *HTTP* binding mode in its postconditions (for instance, because the CAT only provides *FILE* binding mode in its postconditions) and the terminal binding mode is defined as *HTTP* only capable.

TABLE I
BINDING MODES PROPOSED BY CAIN-21

Binding mode	Description
urn:mpeg:mpeg21:2007:01-BBL-NS:handler:INPROCESS	In-process technique used to transfer information between CATs. In the case of CAIN-21, objects loaded in memory use the pull model to request data by means of a memory buffer.
urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE	Can read/write files provided in the URL. This is an appropriate binding for OdA mode
urn:mpeg:mpeg21:2007:01-BBL-NS:handler:TCP	Can read/write TCP sockets. The IP+port are provided in the URL. This is an appropriate binding for OnA mode.
urn:mpeg:mpeg21:2007:01-BBL-NS:handler:HTTP	Can read/write HTTP protocol. The IP+port are provided in the URL. This is an appropriate binding for OnA mode.
urn:mpeg:mpeg21:2007:01-BBL-NS:handler:RTSP	Can read/write RTSP protocol. The IP+port are provided in the URL. This is an appropriate binding for OnA mode.

E. Properties DI

The *Properties DI* tool gathers all the information required by the multimedia adaptation process following a declarative approach. The main purpose of this tool is that changes in the set of multimedia properties do not imply changes in the

[6] The author uses the name *ConversionDescription* to refer to the notion of *ConversionCapability*.

underlying source code. Particularly, with the *Properties DI* tool all the information is described consistently using so called *multimedia properties*. These multimedia properties include the *Content DI*, the *Usage Environment DI* and the *CAT Capabilities DI*. Each property is represented as a label with an associated XPath [8] expression.

1) Addressing mechanism

Even though the PM is still responsible for parsing the documents and loading them in memory, the ADM does not directly access these properties. In this way, changes in the metadata do not imply changes in the underlying source code. Instead, these changes imply only modifying the *Properties DI*.

```
<dia:DIA xmlns="urn:vpu:cain21-properties-di"
         xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-
NS"
         xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
 <dia:Description xsi:type="PropertiesDIType">
  <DIProperties>
    <Property name="genre" required="false"

xpath="/Item/Descriptor/Statement/Mpeg7/Descriptio
nUnit/Genre/@href"/>
  </DIProperties>
  <ComponentProperties>
   <Property name="id" required="true"
xpath="/@id"/>
   <Property name="url" required="true"
xpath="/Resource/@ref"/>
   <Property name="mime_type" required="false"
xpath="/Resource/@mimeType"/>
    ..................
   <ComposedProperty name="visual_frame"
required="false">
    <Value
xpath="//Mpeg7/Description/MediaInformation/MediaP
rofile

//MediaFormat/VisualCoding/Frame/@width"/>
    <Value
xpath="//Mpeg7/Description/MediaInformation/MediaP
rofile

//MediaFormat/VisualCoding/Frame/@height"/>
   </ComposedProperty>
  </ComponentProperties>
  <CATProperties>
   <Property name="id" required="true"
xpath="/@id"/>
   <Property name="cat_class_name" required="true"
xpath="/CATClassName"/>
   ..................
  </CATProperties>
  <ConversionProperties>
   <Property name="id" required="true"
xpath="/@id"/>
   <Property name="content_degradation"
required="true"
             xpath="/ContentDegradation"/>
   <Property name="computational_cost"
required="true"
             xpath="/ComputationalCost"/>
   <!-- Input properties -->
   <Property name="pre_url" required="true"
xpath="/Preconditions/URL"/>
   <Property name="pre_binding" required="true"
xpath="/Preconditions/Binding"/>
   <Property name="pre_content" required="true"
xpath="/Preconditions/Content"/>
    ..............
   <!-- Output properties -->
   <Property name="post_url" required="true"
xpath="/Postconditions/URL"/>
   <Property name="post_binding" required="true"
xpath="/Postconditions/Binding"/>
   <Property name="post_content" required="true"
xpath="/Postconditions/Content"/>
    ..................
  </ConversionProperties>
  <UsageEnvProperties>
   <TerminalProperties>
    <Property name="id" required="true"
xpath="/@id"/>
    <Property name="binding" required="true"

xpath="/TerminalCapability[@type='cde:HandlerCapab
ilitiesType']
          /Handler/@handlerURI"/>
     ..........
   </TerminalProperties>
   <NetworkProperties>
    <Property name="id" required="true"
xpath="/@id"/>
    <Property name="max_capacity" required="false"

xpath="/NetworkCharacteristic/@maxCapacity"/>
    <Property name="min_guaranteed"
required="false"

xpath="/NetworkCharacteristic/@minGuaranteed"/>
   </NetworkProperties>
   <UserProperties>
    <Property name="id" required="true"
xpath="/@id"/>
     ..............
    <Property name="pref_focus_of_attention"
required="false"

xpath="/UserCharacteristic/ROI/@uri"/>
   </UserProperties>
  </UsageEnvProperties>
 </dia:Description>
</dia:DIA>
```

Listing 2: Properties DI example

The expression of each property points out to the part of the DI where its values are located. XPath expressions are relative to the document. Therefore, the *Properties DI* stores only the XPath of the property. The document that contains these properties is determined during the execution of the adaptation. The *Configuration DI* (introduced in Subsection IV.A) is used to identify these documents. Furthermore, properties are only resolved on-demand. In this way, properties that are never used are not extracted from de DIs. Internally, CAIN-21 uses *xpointer*() [24] expressions to reference both the document and the XML element or attribute to be accessed. The standard Xalan processor [31] is used in our work to gather all these properties.

The *Properties DI* schema that we propose is available in the file *cpr.xsd* of the CAIN-21 software. The *PropertiesDIType* is defined as a derivation by restriction of the MPEG-21 standard *DIADescriptionType*. This type includes four important elements that correspond to the five groups of properties: *DIProperties*, *ComponentProperties*, *CATProperties*, *ConversionProperties* and *UsageEnvProperties*. Listing 2 shows the more relevant parts of the current *Properties DI* of CAIN-21 (the whole document is available in the file *pr.xml* of the CAIN-21 demo). For

instance, in Listing 2 the *ConversionProperties* element contains the property *pre_url* whose XPath expression is "*/Preconditions/URL"*. On resolving this XPath expression in Listing 1, *AnyValue* is obtained indicating that the conversion accepts any value for this property. As another example, on resolving the *pre_binding* property in Listing 2, the *FILE* and *HTTP* binding modes are obtained from Listing 1.

2) Properties and relationships

Subsection IIII.A introduced the Semantic Web. Semantic Web languages such as OWL allow explicitly representing and storing concepts and their relationships in a semantic graph. Software tools such as Protégé facilitate loading this graph from disk to memory. Frequently, automatic-reasoning techniques use this graph to search for relationships among the values and to infer additional information.

In CAIN-21, the concepts are represented by means of properties and the relationships are limited. Specifically, relationships are just intended to assist the matching algorithm developed in [17]. In this way, CAIN-21 uses a delimited subset of the rich relationships that the Semantic Web provides.

The *Properties DI* complies with the *Keep It Short and Simple* (KISS) design principle. This principle recommends to avoid unnecessary complexity and construct systems as simple as possible, but no simpler. The main purpose of the *Properties DI* is not information inference, but to elude changes in the decision algorithm when the metadata under consideration evolves. In contrast, depending on the reasoning techniques, changes in the relationships of the semantic graph imply changes in the underlying reasoning algorithms.

In a nutshell, the matching mechanism tests whether the input of one CAT accepts the output of the previous CAT in the sequence. The matching mechanism also tests whether the terminal accepts the output of the last CAT. In order to perform these tests, usually the simple properties-based representation mechanism has demonstrated to be enough [17]. These tests demonstrated its suitability and also demonstrated that the matching mechanism operates efficiently. However, during the tests, we encountered that occasionally it is convenient to consider more complicated relationships between properties. For instance, to maintain the ratio in the adapted media the width and height should be considered together. In these cases, we use *composed properties*. Listing 2 shows an example of these composed properties. The *visual_frame* property uses the *ComposedProperty* element to gather the width and height elemental values. In the representation schema of CAIN-21, these elemental values can be represented by means of ranges or as a placeholder accepting any value.

F. Extensions to the UED

In particular, this document has identified the following handicaps in the current UED:

1. The *mpeg21:TerminalType* does not include any reference to the modalities of the content that the terminal consumes (images, video, audiovisual, audio, etc). The *mpeg7:Content* description serves this purpose by the *mpeg7:ContentCS* classification scheme

2. The terminal does not provide any description of the binding modes, i.e., the delivery mechanism (such as *HTTP* or *RTSP*) used to consume content as described in Subsection IV.D.

3. The standard MPEG-21 Part 7 *UED* tools do not specify whether the properties of the terminal are mandatory or optional. For instance, if the terminal is defined using the *mpeg21:AudioCapabilitiesType*, does it mean that the adapted media must include audio? Or does it mean that this audio format could be consumed if present?

These semantic gaps include both properties that can be inferred and properties that cannot be inferred (ambiguities). The first gap semantic can be addressed by inference [32] and the other two gaps by extending the current *mpeg21:TerminalType*. More specifically, the first gap can be addressed by inferring the media content (image, video, audiovisual, audio) from the *mpeg21:TransportCapabilitiesType* (illustrated in Listing 3).

To demonstrate how to address the other two gaps, Listing 3 shows a portion of the description of the terminal with *id="iphone"* from the CAIN-21 demo. The extensions that this subsection discussed are marked in bold. The XML Schema with these changes is publicly available in the file *cde.xsd* of the CAIN-21 software.

```
<Terminal id="iphone" xsi:type="cde:TerminalType">
 <TerminalCapability
xsi:type="cde:HandlerCapabilitiesType">
  <Handler handlerURI="urn:mpeg:mpeg21:2007:01-
BBL-NS:handler:FILE"/>
  <Handler handlerURI="urn:mpeg:mpeg21:2007:01-
BBL-NS:handler:HTTP"/>
 </TerminalCapability>
 <TerminalCapability
xsi:type="cde:CodecCapabilitiesType">
  <cde:Decoding
xsi:type="cde:TransportCapabilitiesType">
   <cde:Format
href="urn:vpu:cs:FileFormatCS:2009:3gpp">
    <mpeg7:Name xml:lang="en">
     3GPP file format
    </mpeg7:Name>
   </cde:Format>
  </cde:Decoding>
  <cde:Decoding
xsi:type="cde:VideoCapabilitiesType">
   <cde:Format
href="urn:vpu:cs:VisualCodingFormatCS:2007:1">
    <mpeg7:Name xml:lang="en">
     H.264 Baseline Profile @ Level 1.1
    </mpeg7:Name>
   </cde:Format>
   <cde:CodecParameter
xsi:type="CodecParameterBitRateType">
    <BitRate >32000</BitRate>
   </cde:CodecParameter>
  </cde:Decoding>
  <cde:Decoding
xsi:type="cde:AudioCapabilitiesType"
optional="true">
   <cde:Format
href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4
.3.1">
    <mpeg7:Name xml:lang="en">
     MPEG-2 Audio AAC Low Complexity Profile
    </mpeg7:Name>
```

```
    </cde:Format>
    <cde:CodecParameter
xsi:type="CodecParameterBitRateType">
     <BitRate>7950</BitRate>
    </cde:CodecParameter>
   </cde:Decoding>
 </TerminalCapability>
  ....................
</Terminal>
```

Listing 3: Extended mpeg21:TerminalType

The proposed extensions to the *mpeg21:TerminalType* are:

1. Representing the binding modes of the terminal in the *cde:HandlerCapabilitiesType* description tool. This element makes reference to the *mpeg21:Handler* description tool. shows how to describe that the iPhone terminal supports the *FILE* and *HTTP* binding modes.

2. Mandatory and optional constrains are instances of the hard and soft constraints model developed in [17]. To provide this description, CAIN-21 extends the *mpeg21:TerminalType* with the *optional* attribute. Listing 3 shows how to signal that the audio stream is optional using the optional attribute in the *cde:AudioCapabilitiesType*. If this attribute is absent, CAIN-21 considers the terminal description as a mandatory constraint.

V. TESTS AND VALIDATION

The CAIN-21 demo, publicly available at *http://cain21.sourceforge.net*, provides several tests demonstrating the multimedia adaptation approach of this paper. This section focuses on demonstrating and validating the extensions to the MPEG-21 standard proposed in Section IV.

Subsection III.A described the DI level adaptation interface. Both operations of this interface – i.e., *transform*() and *addVariation*() – have been used in the tests. In addition, to cover a wide range of multimedia adaptations, both images and videos have been selected for the tests reported in this paper. CAIN-21 can also convert images to video through the *Image2VideoCAT*. Its *image_2_video* conversion has also been covered in the tests.

A. Transforming an image to an small video terminal

Test 1 illustrates how a *Content DI* with an image (named *photo.xml*) is adapted to the *id="iphone"* video terminal (shown in Listing 3). The full description of these elements is available in the CAIN-21 demo. The *transform*() software interface receives a *Configuration DI* (described in Subsection IV.A) to indicate the target terminal. The PM of CAIN-21 (described in Subsection III.B) uses the *Properties DI* to gather the properties of the *Content DI*, *CAT Capabilities DIs* and *Usage Environment DI*. After that, the ADM (introduced in Subsection III.B) produces the following sequence of conversions *initial* → *image_transcoder* → *image_2_video* → *ondemand_video_transcoder* → *goal*. In this sequence, *initial* represents the properties of the original *Content DI*. The *Preconditions* and *Postconditions* of *image_transcoder*, *image_2_video* and *ondemand_video_transcoder* are described in their corresponding *ConversionCapabilities* elements as explained in Subsection IV.C. Lastly, *goal* represents the properties adapted content. The *image_transcoder* conversion transcodes the image format and size to the preconditions of the *image_2_video* conversion (i.e., JPEG image format and 3:4 aspect ratio). The *image_2_video* conversion accepts only JPEG images and produces only MPEG-2 video. *The ondemand_video_transcoder* (whose conversion capabilities appear in Listing 1) transcodes the MPEG-2 video to the constraints of the terminal (3GPP according to Listing 3). In this Test 1, the ADM has selected the *FILE* binding mode the conversions steps. This happened because all the conversions provide this transfer mechanism in their *Preconditions* and *Postconditions* description tools.

If we change the terminal of Test 1 from *"iphone"* to *"http_nokia_n95*", we have the didactic Test 2. This test fully demonstrates the usefulness of the binding modes. In Test 2, CAIN-21 produces a sequence with four conversions *initial* → *image_transcoder* → *image_2_video* → *ondemand_video_transdoder* → *http_delivering* → *goal*. Specifically, CAIN-21 has added to the end of the sequence the *http_delivering* conversion to change the binding property from *FILE* to *HTTP*. In Test 1 the *"iphone"* terminal supported the *FILE* delivery mechanism (see Listing 3), which corresponds to the binding property at the output of *ondemand_video_transcoder*. Therefore CAIN-21 did not add the *http_delivеding* conversion at the end of the sequence. However, in Test 2, the *"http_nokia_n95"* terminal only supports the *HTTP* binding mode, and therefore CAIN-21 has added the *http_delivering* conversion at the end of the sequence. This conversion has the *FILE* binding mode in its preconditions and the *HTTP* binding mode in its postconditions: this indicates that the purpose of this tool is to transfer the input file using the *HTTP* standard protocol.

B. Summarizing variations of video news items

Test 3 summarizes and adapts a *Content DI* containing a news item to three different terminals [33]. Listing 4 shows the original *Content DI* to be adapted.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
      xmlns:cdi="urn:vpu:cain21-di"
      >
 <Item xsi:type="cdi:ItemType">

  <!-- Classification -->
  <Descriptor>
   <Statement mimeType="text/xml">
    <Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2004">
     <DescriptionUnit xsi:type="ClassificationType">
      <Genre
href="urn:mpeg:mpeg7:cs:ContentCS:2001:1.1.13">
       <Name xml:lang="en">Natural disasters</Name>
      </Genre>
      <Genre
href="urn:mpeg:mpeg7:cs:ContentCS:2001:1.5.1">
       <Name xml:lang="en">Political</Name>
      </Genre>
     </DescriptionUnit>
    </Mpeg7>
```

```
   </Statement>
  </Descriptor>
  <!-- Original content -->
  <Component xsi:type="cdi:VideoComponentType"
id="original">
   <Descriptor xsi:type="cdi:Mpeg7DescriptorType">
    <!-- MPEG-7 MediaDescriptionType describing the
resource -->
    ................
   </Description>
   <Resource mimeType="video/mpeg"
ref="../mesh/didl/flood2video.mpg"/>
  </Component>
 </Item>
</DIDL>
```

Listing 4: Original DI to be summarized and adapted in Test 3

The MPEG-7 *ClassificationType* description type indicates that the news item contains natural disaster and political content. The video is stored in a *Component* element with *id="original"*. This *Component* contains and MPEG-7 *MediaDescriptionType* description of the *Resource* element. The original video is MPEG-1 video and has a resolution of 720x576. This video is summarized according to the methods explained in [33]. Subsequently, the DI is adapted to three terminals. The terminals for the adaptation are all MPEG-2 terminals and have, respectively, screen sizes of 720x576, 352x288 and 176x144. In Test 2, the *addVariation*() operation is used to create the adapted videos in additional *Component* elements of the *Content DI*. Listing 5 shows the adapted *Content DI* with four *Component* elements: the original video and three summarized and adapted variations. The MPEG-7 *VariationDescriptionType* description type indicates that the original video (with *id="original"*) has three variations in the *Component* elements with IDs *"big-sum"*, *"medium-sum"* and *"small-sum"*.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
      xmlns:cdi="urn:vpu:cain21-di"
      xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004">
 <Item xsi:type="cdi:ItemType">
  <!-- Classification -->
  ........................
  <!-- Original content -->
  <Component xsi:type="cdi:VideoComponentType"
id="original">
   ..........
   <Descriptor xsi:type="VariationDescriptionType">
    <VariationSet>
     <Source xsi:type="AudioVisualType">
      <AudioVisual>
       <MediaLocator>
        <MediaUri>#original</MediaUri>
       </MediaLocator>
      </AudioVisual>
     </Source>
     <Variation priority="1">
      <Content xsi:type="AudioVisualType">
       <AudioVisual>
        <MediaLocator>
         <MediaUri>#big-sum</MediaUri>
        </MediaLocator>
       </AudioVisual>
      </Content>
      <VariationRelationship>
       summarization
      </VariationRelationship>
     </Variation>
     <Variation priority="2">
      <Content xsi:type="AudioVisualType">
       <AudioVisual>
        <MediaLocator>
         <MediaUri>#medium-sum</MediaUri>
        </MediaLocator>
       </AudioVisual>
      </Content>
      <VariationRelationship>
       summarization
      </VariationRelationship>
     </Variation>
     <Variation priority="3">
      <Content xsi:type="AudioVisualType">
       <AudioVisual>
        <MediaLocator>
         <MediaUri>#small-sum</MediaUri>
        </MediaLocator>
       </AudioVisual>
      </Content>
      <VariationRelationship>
       summarization
      </VariationRelationship>
     </Variation>
    </VariationSet>
   </Descriptor>
   ..............
   <Resource mimeType="video/mpeg"
ref="../mesh/didl/flood2video.mpg"/>
  </Component>
  <!-- Big size summarized content -->
  <Component xsi:type="cdi:VideoComponentType"
id="big-sum">
   ................
  </Component>
  <!-- Medium size summarized content -->
  <Component
xsi:type="cdi:VideoComponentType"id="medium-sum">
  </Component>
  <!-- Small size summarized content -->
  <Component xsi:type="cdi:VideoComponentType"
id="small-sum">
   ...........
</Component>
 </Item>
</DIDL>
```

Listing 5: Summarized and adapted DI in Test 3

Test 3 uses three *Configuration DIs*. These *Configuration DIs* use the *ARC* descriptions (described in Subsection IV.B) to request the adaptation to three terminals respectively labelled as *"720x576"*, *"352x288"* and *"176x144"* in the *Usage Environment DI*. For Test 3, we needed to create a CAT named *RawVideoCombinerCAT*. Its *CAT Capabilities* appear in the file *raw_video_combiner_cat.xml* of the CAIN-21 demo. This CAT was necessary to retrieve the summarized video from the summarization module (further explained in [33]) through two TCP sockets: one for raw WAV audio and one for RAW video. To this end, we created an additional binding mode (see Subsection IV.D) labelled as *urn:mpeg:mpeg21:2007:01-BBL-NS:handler:TCP*.The three terminals in Test 3 were defined with the standard *HTTP* binding mode in **Table** . During the adaptation, the ADM produced a sequence with three conversions: *initial* → *raw_video_combiner* → *online_video_transcoder* → *http_delivering* → *goal*.

C. Extensions demonstrated in the tests

To recapitulate, justify and validate the extensions to the

MPEG-21 schema that this paper proposes the following conclusions are offered:

1. The proposes description schema enables the description of multiple terminals respectively labelled in the tests of this section as *"iphone"*, *"http_nokia_n95"*, *"720x576"*, *"352x288"* and *"176x144"*. To indicate the target terminal of the adaptation this information has to be provided. As MPEG-21 does not define a description tool for this purpose, Subsection IV.B has proposed this description tool.

2. To enable automatic adaptation decision, the inputs and outputs of the conversion tools have to be provided. Subsection IV.C proposed the *CAT Capabilities* description tools. The feasible inputs and output properties are defined using the *Preconditions* and *Postconditions* elements.

3. For automatic decision, it is also necessary to know how the media is going to be transferred to both the next CAT and the target terminal. This justifies the introduction of the binding mode in the description schema.

4. The modality of the content appears in the *mpeg7:Content* description tool. However, this information is not provided by the *mpeg21:TerminalType* description type. During the decision process the modality of the content that the terminal accepts has to be determined. The inference rule described in Subsection IV.F can be used in this case. Specifically, from the *mpeg21:TransportCapabilitiesType* description tool of Listing 3 it can be inferred that the content has to be visual or audiovisual. See [32] for a further explanation of this mechanism.

5. In Test 2 the decision process needs to know the binding mode to identify that the *http_delivering* conversion has to be added to the sequence. This information removes ambiguity and validates the first extension in Subsection IV.F.

6. Before adding the *optional* attribute to the *mpeg21:AudioCapabilitiesType* description (extension 2 in Subsection IV.F), CAIN-21 did not encounter a sequence for Test 1 and Test 2. This happened because the output of the *image_2_video* conversion did not contain this information. This problem has been further described in [32]. Labelling the audio as *optional* (see **Listing 3**) allows for ignoring the audio properties during the computation of the sequence.

VI. MULTIMEDIA ADAPTATION ENGINES COMPARISON

This section provides a comparative review of six multimedia adaptation engines, which operate in the MPEG-21 framework: ConversionLink [10], koMMa [15], BSD [12], DCAF [20] NinSuna [21] and CAIN-21. These engines have been introduced in Subsection II.D. TABLE shows the year of publication that this paper is analyzing.

A. Aspects to compare

The comparison based on six aspects, namely:

1. The automatic *decision-making method* that the engine implements.

2. Whether the engine supports *multi-step* adaptation.

3. Whether the engine provides a *complete-solution*, i.e., finds all the solutions.

4. The *extensibility mechanism* (if any).

5. The *multimedia content* that the engine is prepared to adapt.

6. The *semantic adaptations* that the engine considers.

TABLE II
SUMMARY OF COMPARATIVE OF MULTIMEDIA ADAPTATION ENGINES

	ConversionLink	koMMa	BSD	DCAF	NinSuna	CAIN-21
Year	2005	2007	2008	2008	2010	2013
Decision-making method	Ad-hoc	Knowledge-based	Quality-based	Quality-based	Quality-based	Knowledge-based + Quality-based
Multi-step	No	Yes	Yes	No	No	Yes
Complete solutions	Unspecified	No	Ranking	Ranking	Unspecified	Knowledge-based + Ranking
Extensibility mechanism	Yes	Yes	No	No	Yes	Yes
Multimedia content	Images + Video + Audio	Image + Video	Scalable content	General video	Scalable content	Images + Video + Audio
Semantic adaptation	Scene adaptation	OWL description	gBSD + IOPins	gBSD + IOPins	RDF + gBSD + SOIs	ROI

Subsection II.C divided automatic decision-making methods into quality-based methods and knowledge-based methods. koMMa and CAIN-21 rely on knowledge-based methods, whereas BSD, DCAF and NinSuna rely on quality-based methods. ConversionLink is a generic description engine that does not specify the algorithms used to make the adaptation decisions. BSD and DCAF engines use the notion of Pareto optimality. CAIN-21 also uses quality-based decisions during a second step (see, [18] for a further discussion on how CAIN-21 implements these quality-based decisions). Whereas BSD, Ninsuna and CAIN-21 rely on classical multi-attribute optimisation methods, DCAF exploits genetic algorithms to compute this optimization.

Section I introduced the advantages that multi-step adaptation provides. These advantages are frequently studied in knowledge-based methods. The koMMa and CAIN-21 adaptation engines use these methods. BSD is mainly devoted to performing the adaptation of the scalable resource in one step. Nonetheless, the authors have also studied the problem of distributed adaptation, which corresponds to the idea of multistep adaptation in different nodes.

In reference to completeness, quality-based methods usually obtain a complete solution, i.e., all the feasible solutions are obtained and ranked: this is the case of BSD, DCAF and CAIN-21. More specifically, these engines create a ranking among the available solutions. Well-known quality metrics such as PSNR or VQM [34] are used to create this ranking. Regarding the knowledge-based methods, koMMa only extracts one solution. CAIN-21 analyses all of them using both the knowledge-based and quality-based decision methods. NinSuna and ConversionLink do not specify the completeness of their decisions.

The idea of extensibility appears in ConversionLink, koMMa, NinSuna and CAIN-21. Both the *ConversionLink* and the *CATCapabilities* description tools include the standard *ConversionCapabilities* [9] description tool. The differences

between these descriptions were discussed in Subsection IV.C. BSD and DCAF do not examine their own extensibility. NinSuna discusses its extensibility regarding its format independence.

In reference to the supported media, BSD and NinSuna are particularly effective dealing with scalable media, while DCAF deals with general video resources. ConversionLink, koMMa and CAIN-21 are intended to deal with a wider range of media resources. Specifically, ConversionLink and CAIN-21 can manage images, audio and video, whereas koMMa provides adaptation tests involving images and video. The scalable content adaptation implemented in BSD and DCAF is one of the adaptations that CAIN-21 incorporates. Moreover, [18] discusses how scalable video adaptation is carried out inside a CAT called the SVCCAT. The scalable content adaptation corresponds to to the idea of resource conversion in the case of ConversionLink.

In reference to semantic adaptations, *ConversionLink* allows scene level adaptation. It addresses the question of semantic adaptation of documents based on temporal, spatial and semantic relationships between the media objects. koMMa relies on Semantic web Services to describe its adaptation capabilities and to identify the sequence of conversions. BSD and DCAF use the *gBSD* [3] and the *AdaptationQoS* with *IOPins* [3] description tools. *IOPins* are linked to semantics annotating the video stream on a semantic level. NinSuna uses RDF to describe semantic relationships. It also uses the *gBSD* description tools to provide semantic adaptation for the selection of *Scenes Of Interest* (SOIs) as well as for frame-rate reduction. CAIN-21 makes use of *Regions of Interest* (ROIs) inside some CATs such as the *Image2VideoCAT*. Tests involving semantic adaptation in CAIN-21 have been reported to [19].

B. Adaptation approaches comparison

This section describes the reasoning behind the different approaches chosen for the adaptations engines described above.

In the design of ConversionLink it can be observed an effort to create a general MPEG-21 description of multimedia adaptation, but without paying attention to the underlying adaptation algorithms. Several adaptations are described, but they are ad-hoc adaptations of a specific media item, that is, the decision and adaptation methods do not generalize to make them reusable for other media contents or formats without modifying the underlying implementation.

The major contribution of koMMa is to demonstrate how Web Services are able to represent and calculate multimedia adaptation sequences. koMMa studies in depth the semantic description and planning of the sequence, but defers the study and exploitation of the signal level features of the media to be adapted.

The reasoning behind quality-based methods (i.e., BSD, DCAF and NinSuma) is to find the parameters that maximize the quality or utility of the adaptation. These parameters exist or are applicable only to specific media formats (e.g., scalable video), and hence these methods do not aim to accomplish the adaptation of the widest possible range of multimedia formats. Conversely, knowledge-based methods (such as koMMa or the first phase of CAIN-21) focus on the reusability of the adaptation algorithms as a mean to archive the widest possible range of adaptations. With this purpose, knowledge-based methods propose the use of pluggable adaptation tools, and elaborate a decision method that, without human intervention, finds the adaptation tools and corresponding parameters to accomplish each adaptation scenario.

CAIN-21 contributes to the previous ideas by proposing the combination of knowledge-based and quality-based methods in two steps. Firstly, a descriptions-based method that finds all the feasible adaptations, secondly the CATs use media features to select the parameters that maximize the quality or utility of the adaptation.

VII. CONTRIBUTIONS AND CONCLUSIONS

The objective of the Semantic Web is to represent knowledge in a format that can be automatically processed without human intervention. This paper contributes to this objective by introducing the idea of implied and explicit ontology, envisioning MPEG-21 as a implied ontology, and demonstrating how this (pseudo)-ontology is enough to accomplish multimedia adaptation decision-making automatically (i.e., without human intervention in the decision process).

This paper has explained CAIN-21, its extensibility mechanism and the infrastructure to perform automatic adaptation decisions. So, assuming that enough CATs are available, CAIN-21 is capable of managing all content that can be represented as a DI.

As said, CAIN-21 embraces MPEG-21 and shows its good level of expressiveness, as most of new descriptions for concepts and relationships can be represented with this standard. However, this paper has identified and discussed several handicaps in the MPEG-21 description capabilities. As discussed in Section 4, extensions were provided to solve these handicaps. In particular, the paper identifies gaps in the MPEG-21 schema to represent the UED, in the binding modes, in the *ConversionCapabilities* to represent preconditions and postconditions, and in the *mpeg7:Content* description tool.

Another important unique aspect of CAIN-21 is that semantic and quality-based adaptations have been put apart from the knowledge-based decision mechanism and transferred to the CATs. Therefore, our proposal for the knowledge-based decision method makes the adaptation engine independent of the semantics in the content to be adapted. Specifically, the independence is achieved by making decisions according to the media format. Subsequently, quality-based and semantic adaptation for particular content (e.g. soccer, news items) can be integrated inside the CATs. As can be seen in TABLE , CAIN-21 combines these two major decision-making methods and integrates a complete algorithm, i.e., an algorithm that identifies all the feasible adaptations that produce content

satisfying the usage environment constraints.

The paper also includes the reasoning behind and comparison of the different multimedia adaptation decision approaches.

REFERENCES

[1] J. Lachner, A. Lorenz, B. Reiterer, A. Zimmermann, H. Hellwagner. "Challenges Toward User-Centric Multimedia". Proceedings of Second International Workshop on Semantic Media Adaptation and Personalization (SMAP 2007), pp 159-164, London, UK, Dec. 2007.

[2] F. Pereira, I. Burnett. "Universal multimedia experiences for tomorrow". IEEE Signal Processing Magazine, 20(2):63-73, Mar. 2003.

[3] I. Burnett, F. Pereira, R. V. de Walle, R. Koenen (editors). "The MPEG-21 Book". John Wiley and Sons, Apr. 2006.

[4] Authors. "CAIN-21: An extensible and metadata-driven multimedia adaptation engine in the MPEG-21 framework", Lectures Notes in Computer Science, Springer Verlag, vol. 5887, pp. 114-125, 4th International Conference on Semantic and Digital Media Technologies (SAMT 2009), Dec. 2009.

[5] N. Shadbolt, T. Berners-Lee, W. Hall, "The Semantic Web Revisited". IEEE Intelligent Systems. 21(3):96-101, May 2006.

[6] B. S. Manjunath, P. Salembier, T. Sikora (eds.). "Introduction to MPEG-7 Multimedia Con-tent Description Interface". Wiley & Sons, Jun. 2002.

[7] WWW Consortium (W3C). "Multimedia Vocabularies on the Semantic Web". http://www.w3.org/2005/Incubator/mmsem/, Jul. 2007 (last verified Jun. 2012).

[8] WWW Consortium (W3C). "XML Path Language (XPath) Version 1.0", Nov. 1999.

[9] ISO/IEC 21000-7:2005. " Information technology - Multimedia framework (MPEG-21) - Part 7: Digital Item Adaptation, AMENDMENT 1: DIA Conversions and Permissions", Jan. 2005.

[10] M. Kimiaei. "Adaptation de Contenu Multimédia avec MPEG-21: Conversion de Ressources et Adaptation Sémantique de Scènes". PhD. Thesis. Ecole Nationale Supérieure des Télé-communications, Jun 2005.

[11] D. Mukherjee, E. Delfosse, J.G. Kim, Y. Wang. "Optimal adaptation decision-taking for teminal and network quality-of-service", IEEE Transactions on Multimedia vol. 7(3), pp. 454-462, Jun. 2005.

[12] C. Timmerer, "Generic Adaptation of Scalable Multimedia Resources". Verlag Dr. Muller, May 2008.

[13] M.C. Angelides, and A.A. Sofokleous, A game approach to optimization of band-width allocation using MPEG-7 and MPEG-21, Multimedia Tools and Applications, pp. 1-23, DOI:10.1007/s11042-011-0981-0. Mar. 2012.

[14] B. Girma, L. Brunie, J.M. Pierson. "Planning-Based Multimedia Adaptation Services Composition for Pervasive Computing". Proceedings of 2nd International Conference on Signal-Image Technology Internet based Systems (SITIS 2006) pp. 132.143, Hammamet, Tunisia, Dec. 2006.

[15] D. Jannach, K. Leopold. "Knowledge-based multimedia adaptation for ubiquitous multimedia consumption", Journal of Network and Computer Applications, 30(3):958-982, Aug. 2007.

[16] Feng X., Tang R., Zhai Y., Feng Y., Hong B., "An MPEG-21-driven multimedia adaptation decision-taking engine based on constraint satisfaction problem", Proc. SPIE 8878, Fifth International Conference on Digital Image Processing (ICDIP), 2013.

[17] Authors, Bounded non-deterministic planning for multimedia adaptation, Applied Intelligence, Springer, vol. 36:1, pp. 29-60 Mar. 2012.

[18] Authors. Improving Scalable Video Adaptation in a Knowledge-Based Framework. Proceedings of the 11th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS 2010), ISBN 978-1-4244-7848-4, Desenzano, Italy, Apr. 2010.

[19] Authors. "Automatic adaptation decision making using an image to video adaptation tool in the MPEG-21 framework". Proceedings of the International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS 2009), pp. 222-225, London, UK, May 2009.

[20] A. A. Sofokleous, M. C. Angelides, "DCAF: An MPEG-21 Dynamic Content Adaptation Framework". Multimedia Tools and Applications, 40(2):151-182, Nov. 2008.

[21] D. Van Deursen, W. Van Lancker, W. De Neve, T. Paridaens, E. Mannens, R. Van de Walle, "NinSuna: a fully integrated platform for format-independent multimedia content adaptation and delivery using Semantic Web technologies". Multimedia Tools and Applications, 46(2):371-398, Jan. 2010.

[22] J. Lee, "Scalable continuous media streaming systems", Ed. Wiley. Jul 2005.

[23] P. van Beek, J.R. Smith, T. Ebrahimi, T. Suzuki, J. Askelof. "Metadata-driven multimedia access". IEEE Signal Processing Magazine, 20(2):40-52, Mar 2003.

[24] WWW Consortium (W3C), "XPointer xpointer() Scheme", http://www.w3.org/TR/xptr-xpointer/, Dec. 2002 (last verified Jun. 2012).

[25] J. Bormans, J. Gelissen, A. Perkis, "MPEG-21: The 21st century multimedia framework". IEEE Signal Processing Magazine, 20(2):53-62, Mar. 2003.

[26] S. De Zutter, R. Van de Walle, "Enhanced Quality of Experience in Heterogeneous Environments from a Content Author's Perspective". Proceedings of Sixth FirW PhD Symposium, Ghent University, Nov. 2005.

[27] [M. Einhoff, .J. Casademont, F. Perdrix, S. Noll. "ELIN: A MPEG Related News Framework". Proceedings of the 47th International Symposium (ELMAR 2005), pp. 139-140, Zadar, Croatia, Jun. 2005.

[28] Authors. Towards a fully MPEG-21 compliant adaptation engine: complementary description tools and architectural models. Lectures Notes in Computer Science, Springer Verlag, vol. 5811, pp. 155-169, 6th International Workshop on Adaptive Multimedia Retrieval (AMR 2008), Jun. 2008.

[29] L. Rong, I. Burnett, "Facilitating Universal Multimedia Adaptation (UMA) in a Heterogeneous Peer-to-Peer Net-work". Second International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution, 2006. AXMEDIS apos;06. pp. 105-109. Dec. 2006.

[30] ISO/IEC 21000-18:2008 FDAM 1"Information technology - Multimedia framework (MPEG-21) - Part 18: Digital Item Streaming", Mar. 2008.

[31] Apache Java Xalan XSLT Processor. Available online at http://xml.apache.org/xalan-j/ (last verified Jun. 2012).

[32] Authors. Automatic adaptation decision making in the MPEG-21 framework: mechanisms and complementary description tools. Proceedings of the 11th Inter-national Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS 2010), ISBN 978-1-4244-7848-4, 12-14 April, Desenzano, Italy, Apr. 2010.

[33] Authors. Instant Customized Summaries Streaming: a service for immediate awareness of new video content. Lectures Notes in Computer Science, Springer Verlag, vol. 6535, pp. 24-34, (ISBN 978-3-642-18449-9), 7th International Workshop on Adaptive Multimedia Retrieval (AMR 2009), Madrid, Spain, Sep. 2009.

[34] C. Timmerer, V.H. Ortega, J.M. González, A. León, "Measuring quality of experience for MPEG-21-based cross-layer multimedia content adaptation", ISBN: 978-1-4244-1967-8, Proceedings of the 2008 IEEE/ACS International Conference on Computer Systems and Applications (AICCSA 2008), pp. 969-974, Doha, Qatar, Apr. 2008.

Fernando López works as a post-doctoral senior researcher and assistant lecturer at the Universidad Internacional de la Rioja (UNIR). He is focused on informal learning with mobile devices and specializes in technology-enhanced learning, mobile applications, multimedia, programming languages, research and innovation. Before joining UNIR, he worked for the Universidad Autónoma de Madrid (UAM) as a post-doctoral researcher in European research projects. He obtained his EU-wide recognized PhD degree in Computer Science and Telecommunications in 2010 at the Video Processing and Understanding Lab (VPU Lab) of the UAM, a period during which he worked in 2 European research projects related to his thesis. He has published two journal papers, ten papers in international conferences and workshops, one book and a book chapter. In addition, he worked as assistant professor in the Knowledge Engineering and Multimedia laboratories in the UAM. His current research interests lie in the areas of mobile applications, technology enhanced learning and entrepreneurship. .

José M. Martínez José M. Martínez received the Ingeniero de Telecomunicación degree (six years engineering program) in 1991 and the Doctor Ingeniero de Telecomunicación degree (PhD in Communications) in 1998, both from the E.T.S. Ingenieros de Telecomunicación of the Universidad Politécnica de Madrid. He is Associate Professor at the Escuela Politécnica Superior of the Universidad Autónoma de Madrid. His professional interests cover different aspects of advanced video surveillance systems and multimedia information systems. Besides his participation in several Spanish national projects (both with public and private funding), he has been actively involved in European projects dealing with multimedia information systems applied to the cultural heritage (e.g., RACE 1078 EMN, European Museum Network; RACE 2043 RAMA, Remote Access to Museums Archives; ICT-PSP-FP7-250527 ASSETS, Advanced Search Services and Enhanced Technological Solutions for the Europeana Digital Library), education (e.g., ET 1024 TRENDS, Training Educators Through Networks and Distributed Systems), multimedia archives (e.g., ACTS 361 HYPERMEDIA, Continuous Audiovisual Digital Market in Europe) and semantic multimedia networked systems (e.g., IST FP6-001765 acemedia, IST FP6-027685 Mesh). He is author and co-author of more than 100 papers in international journals and conferences, and co-author of the first book about the MPEG-7 Standard published 2002.

He has acted as auditor and reviewer for the EC for projects of the frameworks program for research in Information Society and Technology (IST). He has acted as reviewer for journals and conferences, and has been Technical Co-chair of the International Workshop VLBV'03, Special Sessions Chair of the International Conference SAMT 2006, Special Sessions Chair of the 9th International Workshop on Image Analysis for Multimedia Interactive Services WIAMIS 2008, Program co-chair of the 7th International Workshop on Content-based Multimedia Indexing CBMI 2009 and General chair of the 9th International Workshop on Content-based Multimedia Indexing CBMI 2011.

Narciso García received the Ingeniero de Telecomunicación degree (five years engineering program) in 1976 (Spanish National Graduation Award) and the Doctor Ingeniero de Telecomunicación degree (PhD in Communications) in 1983 (Doctoral Graduation Award), both from the Universidad Politécnica de Madrid (UPM), Madrid, Spain. Since 1977 he has been a member of the faculty of the UPM, where he is currently a Professor of Signal Theory and Communications.

He leads the Grupo de Tratamiento de Imágenes (Image Processing Group) of the Universidad Politécnica de Madrid. He has been actively involved in Spanish and European research projects, serving also as evaluator, reviewer, auditor, and observer of several research and development programmes of the European Union. He was a co-writer of the EBU proposal, base of the ITU standard for digital transmission of TV at 34-45 Mb/s (ITU-T J.81). He has been Area Coordinator of the Spanish Evaluation Agency (ANEP) from 1990 to 1992 and he is General Coordinator of the Spanish Commission for the Evaluation of the Research Activity (CNEAI) since 2011.

He was awarded the Junior and Senior Research Awards of the Universidad Politécnica de Madrid in 1987 and 1994, respectively. His professional and research interests are in the areas of digital image and video compression and of computer vision.