Memoria de Trabajo de Iniciación a la Investigación

A decision system to select media adaptation tools

Estudiante: Fernando López Hernández
Tutor: Dr. José María Martínez Sánchez
Fecha: Madrid, Julio del 2006
Área de conocimiento: Teoría de la señal y comunicaciones
Departamento: Ingeniería Informática. UAM.

Table of Contents

1. Introduction

This section recapitulates the work carried out within present *Trabajo de Iniciación a la Investigación*, motivate the work, and setup the main objectives followed though the document.

1.1. Motivation and objectives

During the last years a lot of technologies and devices have been developed to access a rich and extensive amount of multimedia information. The main target of a group of technologies that can be grouped under the umbrella of Universal Multimedia Access (UMA) [1], is to allow access to a large variety of multimedia resources using diverse terminals, through different networks, and taking into account the user preferences and handicaps.

For personalized delivery of multimedia content, the international standards MPEG-7 [2] (mainly Part 5 [3]) and MPEG-21 [4] (mainly Part 7 [5]) play an important role. Based on these standards, the next-generation of multimedia services will be able to automatically prepare the digital content before delivery according to the client's terminal capabilities, the network conditions, or even the user's characteristics and preferences.

This work was performed within the CAIN framework [6], a content adaptation engine that integrates different content adaptation tools, and that uses media and usage environment metadata in order to identify the best adaptation tools from the available ones. The CAIN framework is introduced in section 2.6.

The main target of the work presented in this document is to design a decision module capable of selecting both: the adaptation tool and the parameters needed to perform the best adaptation of the media content according to the usage environment conditions. Therefore, three objectives have been set:

- To evaluate the state of the art with regard to current multimedia adaptation approaches.
- To design and implement a flexible decision module capable of choosing the best adaptation in real time, and integrate it into CAIN.
- To evaluate the quality of the results using objective and subjective techniques, as well as its computational execution cost.

1.2. Structure of the document

The rest of the document is structured as follows: Section 2 presents the state of the art with respect to multimedia adaptation techniques, and introduces CAIN; the content adaptation framework within the research work developed and presented here was developed.

Section 3 presents the architecture of the Decision Module (DM), which is the module in charge of taking a decision about which adaptation tool to use, and with which parameters. Section 4 presents the initial model used to develop the DM, and section 5 evaluates simplifications and improvements to the initial model which reach better results and reduce the execution time dramatically.

Section 6 presents achieved results and evaluates them. Finally, Section 7 exposes conclusions and future research lines.

2. State of the art

2.1. Universal Multimedia Access (UMA)

Traditionally service providers have supplied multimedia services where multimedia content was provided to users without changes. However, the advent of the new-generation of data networks capable of delivering multimedia content in real time (e.g. Internet), the current wide and growing range of terminals, advances in signal processing which allows to change the signal modality, the diversity of media formats, and the increasing need to fulfil a great variety of user handicaps and preferences for each multimedia service, has turned into a important issue to adapt multimedia contents accordingly before delivery.

Content adaptation is the main objective of a set of technologies that can be grouped under the umbrella of Universal Multimedia Access (UMA) [1]. This means the capability of accessing to rich multimedia content through any client, terminal, and network. In this way content adaptation bridges the gap between content authors and content consumers in a world of increasing multimedia diversity.

An alternative to real-time media adaptation is creating content in multiple formats and at multiple bitrates, and making an appropriate selection at delivery time [7]. This solution is storage-intensive, and could be sufficient for cases in which the types of receiving terminals are limited, but for the general case, where no limitation exists on the terminals or access network, real-time adaptation is necessary.

The description of the content and of the context of use is a critical issue within the UMA framework. An effective way to describe multimedia content is provided by the multimedia metadata standard MPEG-7 [3], and are described in section 2.4. These annotations are useful to automatically select the content to adapt [8]. In addition, the MPEG-21 [4] standard provides different tools that allow efficient description of the context of use, as well as other tools for supporting content adaptation technologies. These tools are specified in the MPEG-21 Part 7 [5], and are described in section 2.5.

2.2. Media adaptation taxonomy

Currently, there are a wide variety of adaptation techniques, although they can be classified in two dimensions [9]:

From the point of view of the information driving the adaptation process, adaptation techniques can be divide in two levels:

1. **Signal driven adaptation.** This kind of adaptation use signal level information to perform adaptation. Examples of this kind of adaptation are spatial or temporal resolution reduction, or mosaic generation of a wide view concatenating various frame of a video panning. This type of adaptation usually applies well to every kind of media signal.

2. **Semantic driven adaptation.** Here semantic information is searches through in order to find important events in the media (e.g. scoring points in basketball videos, news breaking broadcast programs, or security-breaking events in surveillance video). This type of adaptation is domain specific, and a prior knowledge of the domain and ontology must be set before decision algorithms can be applied to detect an event in the scene.

From the point of view of the produced adapted content, adaptation techniques can be divided in two groups:

1. **Transcoding.** Where input and output media are of the same media kind (e.g. video to video, image to image, audio to audio) and only media size or format change. The most common kind of transcoding is format change (e.g. MPEG-2 to MPEG-1, or BMP to JEPG). Another kind of transcoding are spatial, temporal, or quality reduction, all of them used primarily to adapt a signal to low bandwidth networks or to smaller screens. Also, in this situation colour depth or DCT coefficients quantification reduction is usually applied. All these kinds of media size reduction can also be done using Scalable Coding technologies [10][11][12][13][14].

2. **Transmoding**. When transmoding adaptation techniques are applied, original content and adapted content media type change (e.g. video to images, image to text, audio to text). One kind of transmoding is keyframe generation where the most important frames of a video are selected (usually from the semantic point of view). Another kind of adaptation is storyboards, where a group of images of the initial video are selected to describe the whole history. Another example of transmoding is voice to text using a speech recognizer.

Mainly there are two places where content adaptation can be performed [15]:

1. **Server-based adaptation.** In this architecture the server is in charge of discovering the client terminal capabilities, the user preferences, and the available network bandwidth in the client. Server based architecture has the advantage of allowing both static (off-line) and dynamic (on-line) content adaptation. The former refers to automatically creating multiple versions of the authored content, usually executing background tasks in the server. The latter refers to performing on-the-fly adaptations as each client request arrives. The server architecture provides more control to the authors since the adaptation can be tied to the content authoring policy and process, allowing the authors to provide hints on the adaptation in a semantic basis. As a drawback of served-based adaptation, the adaptation tasks results in an additional computational load and resource consumption on the server. Also, if a static adaptation approach is used to generate multiple versions of the content, it will make content management more cumbersome, and increase storage requirements.

2. **Proxy-based adaptation.** In a proxy based architecture the client connects to the media server through a proxy that makes the request to the server on behalf of the client. The proxy intercepts the reply from the server, perform the adaptations, and send the adapted content to the client. It is usually true that

the bandwidth between the proxy and the server is much higher than between the client and the proxy, thus the time to move the original content from the server to the proxy is negligible. The more important advantage of proxy-based adaptation is that there is no need to change existing client and server system. In this way, traditional adaptationless, and initially incompatible content, can be transparently adapted and connected. The more important drawback is that there exists less authors control in the outcome of the adaptation, and that in secure environments, where content is encrypted, allowing content adaptation to the proxy can open a possible security hole.

2.3. Existing approaches for content adaptation

Several approaches have been proposed to perform content adaptation. In this section we remark three of them.

2.3.1 Metadata driven adaptation

In [8] the authors suggest annotating the content, the usage environment and the usage history in order to support adaptation decisions. In particular, the authors discuss the use of MPEG-7 metadata, although MPEG-21 can be used too. These advices were taken into account during the initial development of the CAIN framework.

2.3.2 Utility function maximization

In [9][16][17][18] the authors propose an alternative adaptation, based in a resource-utility trade-off. Due to the fact that there could be multiple adaptation operations, the authors propose to choose adaptations that maximize the user experience. The **resource-utility** relationships represent analogous but broader concepts than the rate-distortion relationship where only two dimensions in the resources space exists.

Specifically this framework model three multi-dimensional spaces, knows as **ARU** and relations among these spaces: **adaptation** (operation to perform: e.g. requantization, frame dropping, DCT coefficients dropping, display size change...), **resource** (media feature: e.g. screen size, colour depth, bitrate...) and **utility** (the quality of the video from the point of view of the user satisfaction). The term "space" is used in a loose sense to indicate the multiple dimensionalities involved in each concept (see Figure 1). For instance, adaptation space could be a 4-dimentional space composed of 4 adaptation operations to perform: e.g. requantization, frame dropping, DCT coefficients dropping, and image size change. Resource space represents media features like bandwidth, computational capability, power, or display size. Resource space must have the same number of dimensions that the utility space and assigns a utility value to each resource value. Each entity is associated with certain resource requirements and utility values.

The authors propose to use a vectorial **utility function UF** to measure the quality of the media after undergoing adaptation of each resource. The independent variables of the UF must be combined to weight the adaptation. Utility can be measure in two levels: the objective level (e.g. PSNR), and the subjective level (e.g. subjective scores, comprehension level...).

As Figure 1 shows, each adaptation operation transforms a pair of points representing a resource and a utility value into a new resource and utility value. The box in the resource space shows constraints imposed by the usage environment to the resources. A valid adaptation operation must keep the resource point into the box. The objective is to find the set of adaptations that keeps the resource point into the box, and reaches the longer distance from the centre in the utility space. The weight of the dimensions in the utility space is not fixed, and is heavily affected by the user preferences.

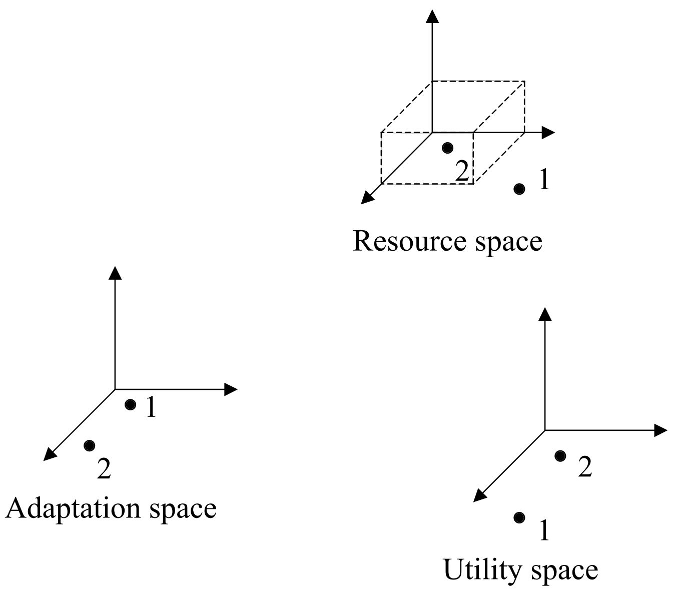

Figure 1: ARU spaces

Key issues arise in realizing the framework in practice: (1) identifying media resources and adaptations that are applicable to entities; (2) modelling resource and utility associated with each resource value; (3) selecting optimal operations given constraints over resources.

2.3.3 Knowledge-based adaptation

In [19] the authors propose a knowledge-based method for building adaptation services. In this approach the original media is transformed in multiple adaptation steps, performed by an extensible set of externals adaptation tools, where the target is to find an adequate adaptation sequence using a planning algorithm [20].

The authors put forward that does not seem realistic that a single adaptation tool will be able to perform all required adaptation steps. Thus they need a mechanism that is, on the one hand, expressive enough for describing such adaptation steps, and in the other hand independent from their actual implementation. Then, interoperability among adaptation tools of different vendors would be possible. In addition, and open and extensible approach has to be chosen, such that changes in the general mechanism are not required when new forms of adaptations emerge, or new adaptation tools become available.

The process of constructing the adaptation plan is modelled as a classical state-space planning problem [20]. Starting with the description of a given multimedia resource, the goal is to apply a sequence of transformations operation on the resource such that the goal state is reached.

Adaptation operations are represented as actions of the planning algorithm. Each action is described in term of input, output, preconditions and effects.

2.4. Multimedia Description Schemes

MPEG-7 defines several **Multimedia Description Schemes (MDS)**. Clause 8 of MPEG-7 Part 5 [3] specifies tools for describing the media features of the coded multimedia data. The coded multimedia data may be available in multiple modalities, formats, coding versions, or multiple instances. The `MediaInformation` description scheme defined in clause 8 allows the description of the original instance of the media, and the multiple variations of the original instance.

The `MediaInformation` description scheme is composed of an optional `MediaIdentification` descriptor, and one or more `MediaProfile` description schemes.

The `MediaIdentification` includes a unique identifier for the content entity independently of the available media profiles, and associated media instances.

The `MediaProfile` allows the description of the various sets of coding parameters, and is subdivided in four major descriptors:

1. The `MediaFormat` descriptor describes the coding parameters, format of the media, coding size, and bitrate among others. As section 2.6.4.3 state, this descriptor have been used to represent media format information within CAIN.
2. The `MediaInstance` descriptor identifies different instances of the media, that is, original content and possible variations (modalities) of the content.
3. The `MediaTranscodingHints` descriptor specifies transcoding hints of the media profile being described. The purpose of this descriptor is to help with improving the quality of transcoded content, and reduces the complexity of transcoding operations.
4. The `MediaQuality` descriptor contains both subjective and objective quality rating about the signal quality of the media profile. This quality can be decreased whenever the signal is compressed or transcoded.

Clause 13 of MPEG-7 Part 5 [3] specify description tools for navigation and access of multimedia content. These descriptors have not been used in this work, but they are summarized here because MPEG-7 has proposed them to be used during adaptation.

Concretely three groups of descriptors are proposed:

1. `Summarization` descriptor describes multimedia summaries and abstracts for efficient browsing and navigation of multimedia content. It's used primarily to specify summaries of time varying audiovisual data, and hierarchical navigation.
2. `Partition`, `Filter`, `View` and `ViewDescomposition` descriptors describe partitions, views and decompositions of image, audio and video signals in the space, time and frequency (e.g. low resolution views or subregions of images).
3. `Variations` descriptor describes the relationship between different variations (modalities) of multimedia content.

2.5. DIA description tools

One of the most important objectives of the international standard MPEG-21 [21] has been the definition a set of interoperability techniques for UMA where concepts, terminology, protocols, and description tools for context description and media adaptation can be formalized.

For the industry, the term **content** is widely used across different technologies, and in many different ways. For this reason the term is deliberately avoided within the context of the MPEG-21 international standard. This term has been replaced by the defined terms **Digital Item (DI)**, **media resource**, or **resource**.

The term **Digital Item Adaptation (DIA)** was proposed by MPEG-21 to define the need to provide content adaptation that fulfils UMA requirements.

MPEG-21 Part 7 [21] specified the syntax and semantics of description tools that may be used to assist the adaptation of Digital Items.

It results vital to make a distinction between **adaptation tools**, which are programs capable of performing a modification of the software, and **description tools**, which in the context of MPEG-7 and MPEG-21 are synonymous for metadata schemes that drive the creation of descriptions.

These adaptation description tools are know as **DIA description tools**, or merely **DIA tools**, and can be used to satisfy transmission, storage, and consumption constraints, as well as quality of service management by the various users.

It is important to emphasise that the adaptation engine itself is out of the DIA tools standardization scope, and left to the industry and researchers' community. MPEG-21 Part 7 only specifies the syntax and semantics of metadata tools that may be used to assist the adaptation of Digital Items.

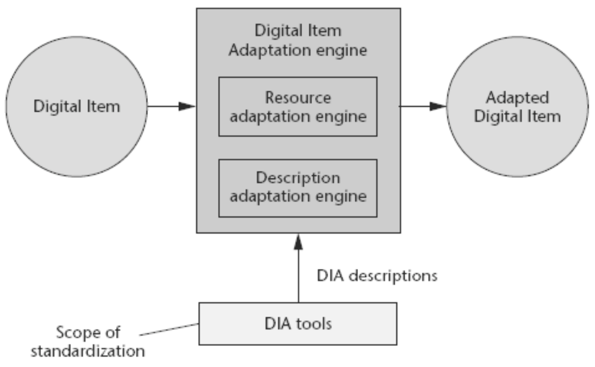

Figure 2: Digital Item Adaptation

Figure 2 best summarizes the concept and principles that govern the development of DIA tools. Note that both resource (media) and description (metadata) about the resource are passed to the DIA engine. Also note that only description tools to assist adaptation have been standardized, and the mechanism of adaptation is left to proprietary implementations.

As proposed by MPEG-21 Part 7, in order to perform content adaptation it's necessary to have a description of the content, a description of the terminal capabilities, and a description of the network conditions. Besides, and in order to enhance the user experience [21][22], not only terminals and network parameters, but also user personalization and environmental conditions should be taken into account during adaptation.

It should be noted that the W3C Consortium is also engaged in efforts to bridge this gap between multimedia content and devices. In particular, the Device Independent Working Group has recently completed a first version of the Composite Capability/Preference Profiles (CC/PP) [23], which specified a structure and vocabularies for devices capabilities and user preferences. Anyway, in the rest of this document only MPEG-7 and MPEG-21 adaptation recommendation have been used.

2.5.1 Classification of DIA description tools

The DIA tools are clustered into eight categories. These categories are built according to their functionality, and use for DIA around the scheme tools and low-level data types. The **scheme tools** provide uniform root elements for all DIA description, as well as some low level and basic data types, such as a set of unsigned integers, single, vector, and matrix data types, which can be used by several DIA tools independently.

The eight description tools defined by MPEG-21 Part 7 [5][7] are:

1. **Usage Environment Descriptor (UED)**. This description tool is further divided in user characteristics, terminal capabilities, network characteristics, and natural environment characteristics.
2. **Bitstream Syntax Description (BSD)**. A bitstream is defined as a structure of fields. DIA use XML to describe the structure of a bitstream fields, and how they are organized in layers of packets.
3. **Bitstream Syntax Description Link (BSDLink)**. Provide a referencing mechanism to many of the tools specified within DIA. In particular, reference is made to: 1) description tools that are capable of steering the adaptation, 2) the bitstream subject to the resource adaptation, 3) the BSD describing the structure of the bitstream, and 4) several transformations including appropriate parameterization.
4. **Terminal and Network Quality of Service**. This description tool, named `AdaptationQoS`, describes the relationship between constraints, feasible adaptations operations satisfying these constraints, and associated qualities.
5. **Universal Constraints Description Tool**. The UCD tools describe constraints on the provider side and on the consumer side. Constraints can be formulated as limitations or optimizations.
6. **Metadata Adaptability**. Metadata is widely used to associate additional information to multimedia content. Whenever the content is adapted, the

associate metadata must change accordingly. This description tool assists with this issue.

7. **Session Mobility**. The means by which DIs are transferred from one device to another device is an important consideration. Session mobility refers to the transfer of both configuration-state (that pertains to a DI in one device to a second device), and application-state (which pertains to information specific to the application currently rendering the DI).
8. **DIA Configuration**. This description tool provides the information required for configuration of the DIA engine, as well as the location of the actual adaptation, that is, the receiver side, the sender side, or either side.

2.5.2 Adaptation engine model

The model [17] of an adaptation engine envisaged in MPEG-21 DIA for UCD and `AdaptationQoS` description tools is shown in Figure 3. The engine absorbs various kinds of metadata to enable fast and efficient adaptation, for instance, based on terminal and network constraints. In a functional basis the module is decoupled in two submodules:

- The **Adaptation Decision Taking Engine (ADTE)** that receives metadata and constraints specifications to make appropriate adaptation decisions.

- The **Bitstream Adaptation Engine (BAE)** that uses the decision provided by the ADTE to perform the actual bitstream adaptation.

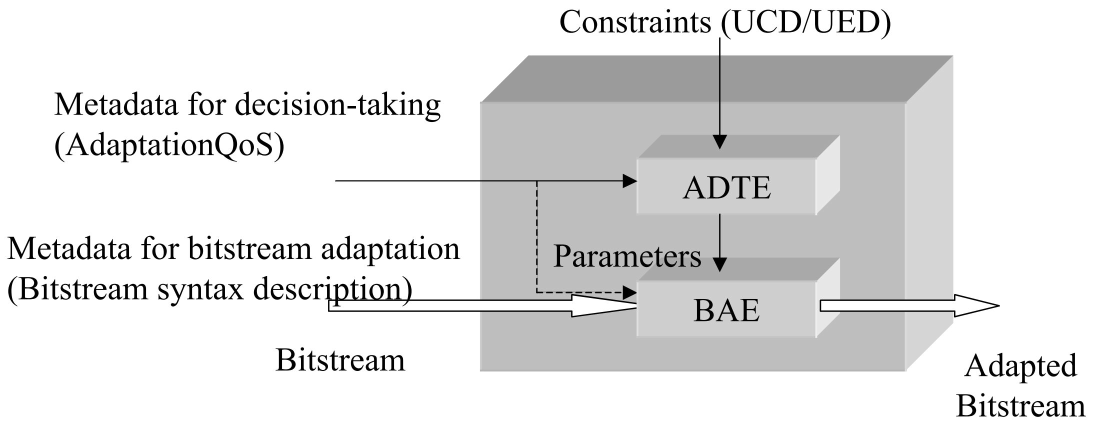
Constraints (UCD/UED)
Metadata for decision-taking
(AdaptationQoS)
ADTE
Metadata for bitstream adaptation
(Bitstream syntax description)
Parameters
BAE
Bitstream
Adapted
Bitstream

Figure 3: Adaptation engine model proposed by MPEG-21 DIA

Note again that while the metadata and constraint specifications are normative in MPEG-21 DIA, the implementation of the ADTE and BAE are left to externals implementations.

2.6. Overview of CAIN

In this section we summarize CAIN [6] (Content Adaptation INtegrator), the framework within the work described in this document is developed. CAIN is a content adaptation manager designed to provide metadata-driven content adaptation [8]. This engine integrates different Content Adaptation Tools (CATs) capable of performing different types of adaptation (see section 2.2): transcoding, transmoding, scalable content, temporal summarization, that may be just signal driven or include some level of semantic driven adaptation [24]. Regardless the kind of adaptation each CAT can perform, CATs point out their adaptation capabilities and the range of parameter values available to carry out the adaptation.

Figure 4 summarizes the CAIN architecture. When CAIN is invoked; it receives the media content, a MPEG-7 MDS (see section 2.4) [1][3] and MPEG-21 BSD [1][5] compliant content descriptions, and a MPEG-21 DIA [1][5] usage environment description (user characteristics and preferences, terminal capabilities, and network conditions). All these inputs are parsed, and the resultant information is sent to the Decision Module (DM). The DM is in charge or deciding which of the available CATs (if any) must be launched in order to produce adapted content and metadata. As output CAIN generate an adapted content and an adapted media description (according to MPEG-7 and MPEG-21 (g)BSD).

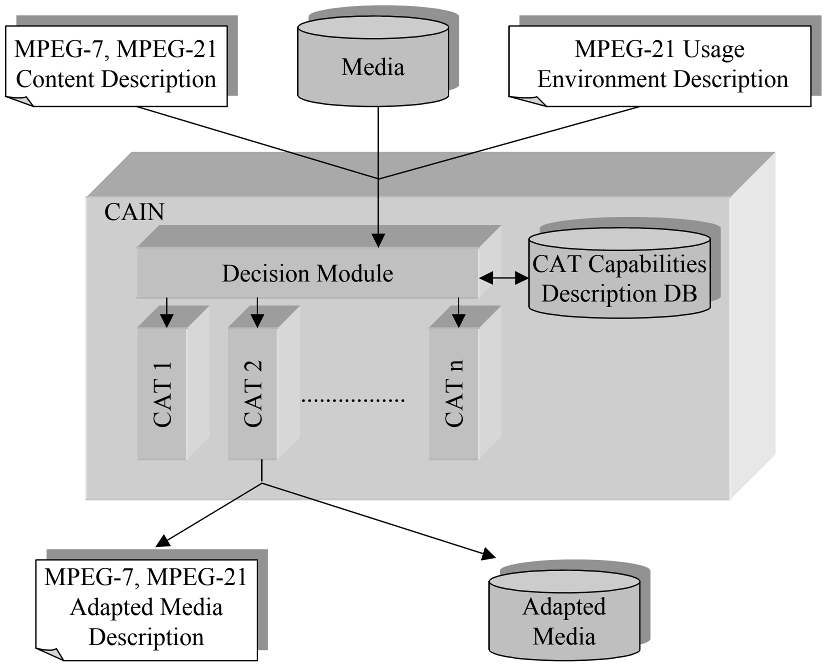

Figure 4: The CAIN architecture

2.6.1 Extensibility in CAIN and the DM

CAIN was proposed as an extensible content adaptation engine [25]. Besides CATs currently integrated in CAIN, it is essential the availability to integrate more CATs and codec in the future. With this prerequisite in mind, CAIN architecture has been designed to allow dynamically adding new CATs. In order to allow these additions, CAIN define both an API interface, and a CAT Capabilities Description File standard format where CATs developers must specify adaptation capabilities of their CATs, and define accepted input and output parameters ranges and values. In this way each new CAT to be added to CAIN must follows defined rules to register its capabilities using one CAT Capabilities Description File.

A XML CAT Capabilities Description Scheme [25] defines the list of adaptation capabilities, specifying in each case which kinds of adaptations the CAT is able to perform, and the possible list of parameters that define the mentioned adaptation, such as input format, output formats, and different features depending of which kind of adaptation is being defined: e.g. accepted input/output, frame-rate, resolution, channels, bitrate, etc...

As Figure 4 shows, CAT Capabilities Description Files are parsed when a CAT is registered, and stored in the CAT Capabilities Description DB. The DM use this stored information to know the available CATs, its features, which adaptation operation each CAT can perform, and which values are accepted in their parameters.

2.6.2 CAIN modular design and workflow

Figure 5 shows current CAIN´s modular design and workflow. When an adaptation request arrives, the Execution Module (EM) is in charge of coordinating the different tasks assigned to other modules.

First, the EM receives through the `adapt()` operation a media content identifier and a usage environment description (according to MPEG-21 DIA specification). Using the media content identifier, the EM request the Communication Module (CM), through the `download()` operation, to retrieve from the Media Repository the media content and its corresponding content description (according to MPEG-7 MDS and MPEG-21 BSD).

CAIN is currently implemented in Java, so these XML documents are parsed (using de Content Parser module and the Usage Environment Parser module) and represented as Java objects. The EM is also in charge of parsing the CAT Capabilities Description File (through the CAT Capabilities Parser module).

All of this parsed information is delivery to the DM, through the `decide()` operation, which has to look for the CAT that best fulfils the adaptation requirements, and which parameters reach the better results. The selected CAT and execution parameters are send back to the EM, which execute the requested CAT, through the `adapt()` operation, passing the retrieved media content and the parameters given by the DM. The CAT returns the adapted content and the adapted media description to the EM.

Finally, the adapted media content and its description (represented as MPEG-7 and MPEG-21 description files) are stored (using the `upload()` operation of the CM) in the Repository.

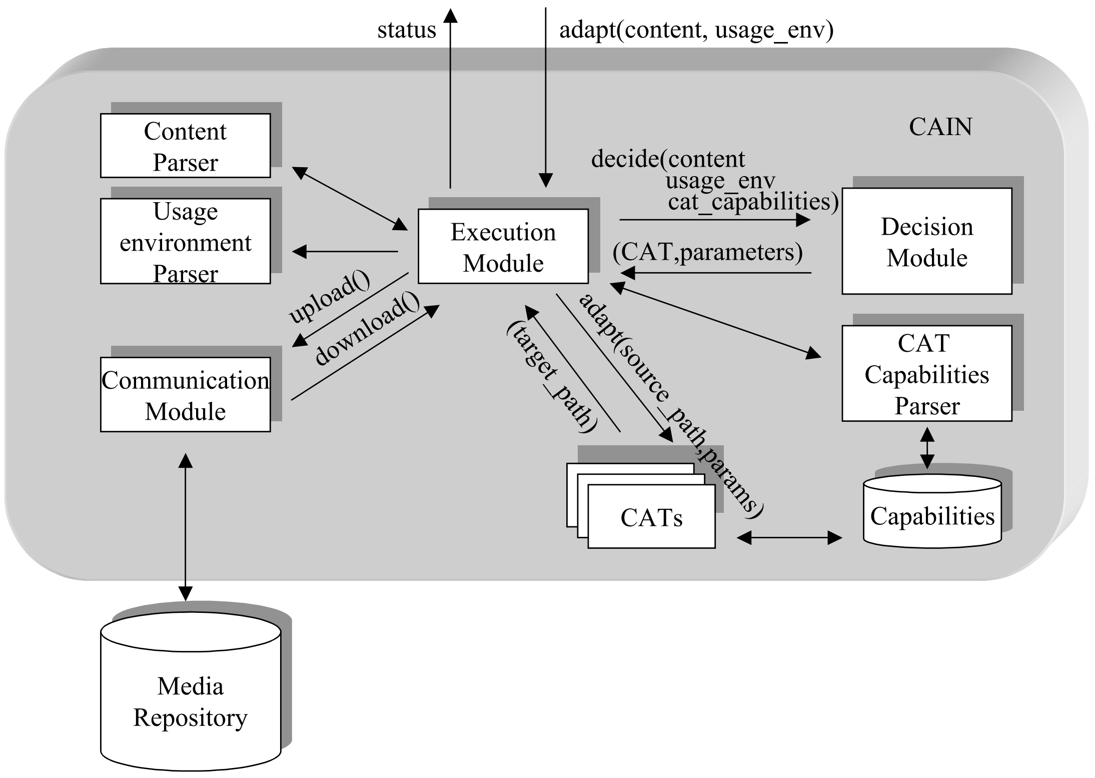

Figure 5: The CAIN modular design and workflow

2.6.3 Supported media

With regard to media resources, the current implementation of CATs in CAIN [26] supports mainly images and videos, as can be seen in Table 1 where the mapping between media formats and CATs categories is depicted. As image format JPEG-2000 has been selected due to its scalability features. In the case of video, various MPEG video formats and the scalable video coding format (aceSVC) introduced in [14] have been selected.

Media Type	**CAT category**
MPEG-1/2/4 SP video, MPEG-1 audio	Transcoding
JPEG 2000, SVC	Scalable content
MPEG-1/2	Semantic driven
MPEG-1/2/4 SP video	Transmoding

Table 1: Relationship between media types and CAT categories

2.6.4 Descriptors for metadata-driven adaptation

The following three subsections describe the descriptors used within CAIN to support metadata-driven adaptation, which are grouped in content, usage environment, and CAT capabilities descriptors. A more in deep description of these metadata can be found in [6] and [25].

2.6.4.1 Content descriptors

The Content descriptors are based on MPEG-7 MDS and MPEG-21 DIA BSD. Figure 6 summarizes the adaptation tools supported by CAIN, and the content descriptors used for this adaptation. The media descriptors metadata should provide support for the following content adaptation modalities:

- Media format adaptation: Supported by MPEG-7 media description (Media Information, transcoding hints...).
- Bitstream adaptation (truncation): Supported by MPEG-21 Bitstream Syntax Description (DIA BSD or gBSD). If the number of formats is reduced, BSD may be the best option, although it does not provide the capability of associating semantic labels (this may provide some "semantic" transcoding capabilities) as gBSD does.
- Media adaptation based on predefined variations and summaries: Supported by MPEG-7 variations and summaries descriptions.
- Semantic and knowledge-based adaptation: Supported by MPEG-7 and JPEG-2000 regions of interest with importance, MPEG-21 gBSD markers, ... annotated by users or labelled in an automatic or supervised way via analysis algorithms.
- Transmoding to text: Supported by MPEG-7 keywords, textual annotations,...

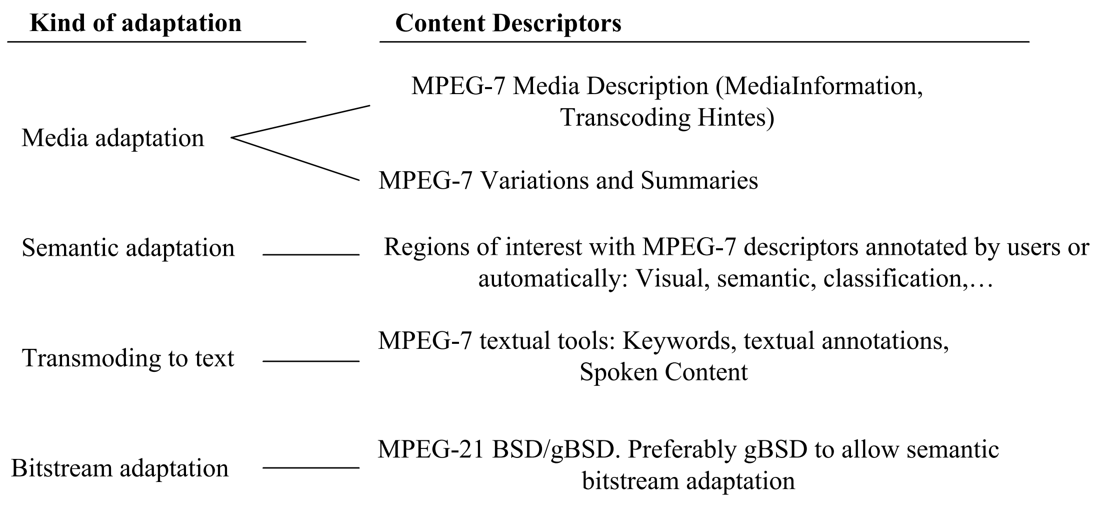
Kind of adaptation
Content Descriptors
Media adaptation
MPEG-7 Media Description (MediaInformation, Transcoding Hintes)
MPEG-7 Variations and Summaries
Semantic adaptation
Regions of interest with MPEG-7 descriptors annotated by users or automatically: Visual, semantic, classification,…
Transmoding to text
MPEG-7 textual tools: Keywords, textual annotations, Spoken Content
Bitstream adaptation
MPEG-21 BSD/gBSD. Preferably gBSD to allow semantic bitstream adaptation

Figure 6: Content descriptors for adaptation

2.6.4.2 Usage environment

Usage environment description includes the description of the terminal capabilities and the network conditions, the preferences and characteristics of the user, as well as of the natural environment. The context description is based on a subset (in the sense of an MPEG Profile [27]) of MPEG-21 DIA Usage Environment Descriptions Tools as shows in Figure 7. The UED tools (MPEG-21 DIA) includes:

- User characteristics: With user interactions (imported from MPEG-7 MDS), presentation preferences, accessibility and location characteristics.
- Terminal description: Currently it uses static terminal descriptors, leaving the possibility of using dynamic characteristics (CPU load, available free storage space, free RAM, ...) for further versions. In any case it will be required to receive from the client the current information about the terminal being used (either the complete description or a pointer to a static description).
- Network description: Currently its uses a static network description too, leaving the possibility of using dynamic characteristics (current congestion, error rate, delay time, ...) for furthers versions. In any case it will be required to receive from the client the current information about the network being used (either the complete description or a pointer to a static description).

Usage Environment description tools (MPEG-21 DIA)
User Description Tools
Usage Preferences
Media Format: content, bitrate, visual coding (format, frame height, frame width, frame aspect ratio and frame rate), audio coding (format, audio channels, sample rate, bits per sample).
Presentation Preferences
o AudioPresentationPreferences: Volume, output device, balance.
o DisplayPresentationPreferences: Colour temperature, brightness, saturation, contrast.
o ConversionPreferences: Media type conversion preferences and priorities.
o PresentationPriorityPreferences: Modality (audio, video...) priorities
Terminal Capabilities Description Tools
Codec Capabilities
Audio, video and image coding/decoding supported formats.
Display Capabilities
Supported display modes (resolution, refresh rate), screen size, colour bit depth.
Audio Output Capabilities
Supported audio modes (sampling frequency, bits per sample), low frequency, high frequency, number of channels…
Storage Characteristics
Input transfer rate, output transfer rate, size, writable.
Network Characteristics Description Tools
Network Capability
Maximum capacity and minimum guaranteed.

Figure 7: Usage Environment Descriptors for adaptation

2.6.4.3 CAT Capabilities

Obviously not every CAT can perform every adaptation operation (bitrate reduction, transcoding, transmoding, audio/video summarization...). In order to achieve CAIN extensibility it's necessary to annotate CATs capabilities. The selected CAT Adaptation Capabilities Description Scheme [26] (see Figure 8) is based on the `MediaFormat` description fool (from MPEG-7 Multimedia Description Schemes [3]), which describes the information related to a file format and coding parameters of the media.

The following list recapitulates the main adaptation capabilities descriptions elements used to describe CAT capabilities:

1. **Header**. The header allows the identification of the described CAT and includes a name and an optional textual description.

2. **Adaptation modality**. This element allows the definition of each adaptation modality, with the possible media formats each adaptation modality is able to receive and to produce. It's composed of an adaptation mode (defines as an MPEG-7 Classification Scheme that allows to describe in details the modality), and a reference to one or more media systems the CAT is able to perform. For example, for a CAT performing video summarization, there can be different modalities, like keyframe replication (which do not reduce the timeline to allow easy audio synchronization), video skimming, and image storyboard.

3. **Media systems**. Each instance of this element allows the definition of media formats at system level by indicating: File format name, file format extension, references to zero or more visual elementary stream, references to zero or more audio elementary stream, and optionally a scene coding format. These elements allow the description of the media system formats each CAT adaptation modality is able to read (input), write (output), or both (common to avoid redundancies).

4. **Elementary streams**. These elements allow de description of video and audio coding parameters. Besides the type of the stream (video, audio, image) the parameters are grouped on input, output and common (in order to reduce redundancies) parameters. The set of parameters are based on MPEG-7 MDS `MediaFormat` description scheme, with some simplifications and extensions looking to allow the definition of adaptation capabilities. When defining a coding format, the DM considers each feasible parameter as a restriction. If no restriction is imposes over a particular property of the codec, it must be considered that the codec is able to deal with any value of this property.

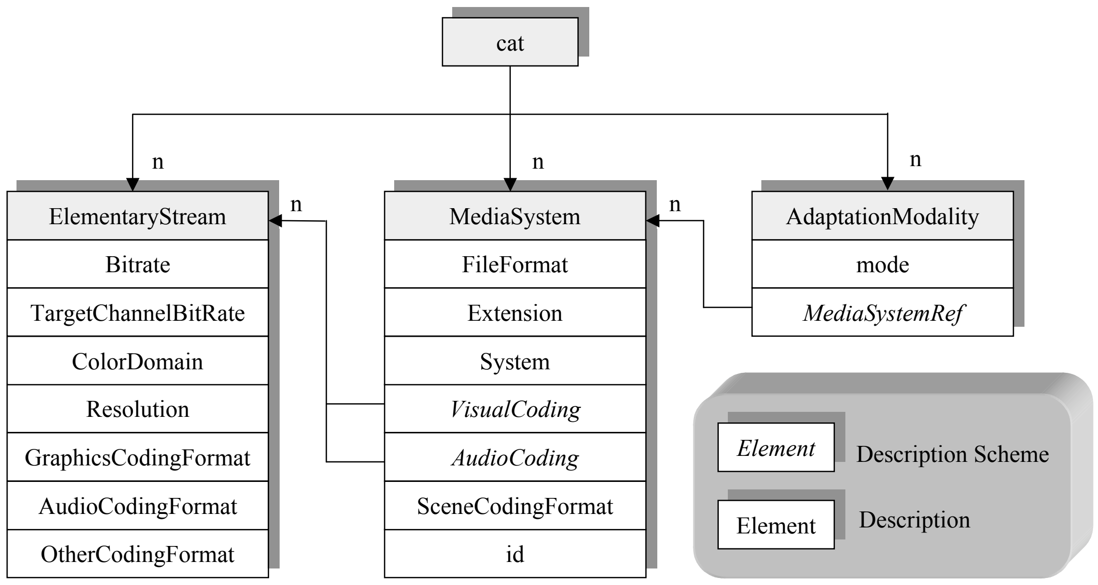

Figure 8: CAT Capabilities Description scheme

2.7. Techniques used to take decision in other kind of problems

A good approach to identify techniques to implement a decision module within the adaptation problem under evaluation is to recap techniques used through the literature to take decision in different contexts.

Some of these decision approaches resulted obviously inappropriate. Between those that deserve a more in deep attention, the following list evaluate their advantages and drawbacks:

1. **Optimization methods**. These methods try to identify target functions to maximize. For instance [9] proposes to identify utility functions and maximize them in order to find the best adaptation. This approach was initially discarded to build the whole system because it appears difficult to find the utility function expression, and because an optimization process doesn't guarantee that every constraint is fulfilled. However, the utility function is being evaluated as future work for the optimization step (see section 7.2.1).

2. **Rule based methods** [28]. Rule based methods are useful when an expert provides rules about the problem and the computer applies this rules to extract automatically new knowledge. It was found out that this kind of knowledge representation was not useful for our purposes because there is no need to infer new knowledge from rules, but of finding the CAT whose adapted digital content match with requested usage environment, and with the CAT capabilities.

3. **Constraints Satisfaction Problems** [29]. It was found out that this method results very useful to solve our decision problem, and it was the selected method as background to implement our initial solution (see section 4).

4. **Planning algorithms**. This method appears useful to solve the adaptation problem, and was the proposed method in [19]. This paper proposes interesting alternative approaches, but due to its publication date we have not used it for our current work. Nevertheless, this method has been evaluated as a future work (see section 7.2.2).

3. The Decision Module

The DM is a software module in charge or receiving an input in form of content description, a **mandatory usage environment description**, and a **desirable usage environment description**. This module is in charge of searching for the CAT that produces the best content adaptation, that is to say, the adaptation that matches more constraints, and therefore yields the best experience to the user.

Terminal capabilities and network characteristics have been included in the mandatory usage environment description, whereas the user preferences have been included in the desirable usage environment description. Some user preferences (user's handicaps) have been moved to the mandatory usage environment description.

Figure 9 illustrates a hypothetical adaptation process that the DM has to look for. As input we have a content description that stand for an unadapted video, that is available in a specific format, bitrate and colour depth (this information must be provided by the Media Repository; if this is not the case, CAIN include a media description generation module in charge of obtaining the media description of the content).

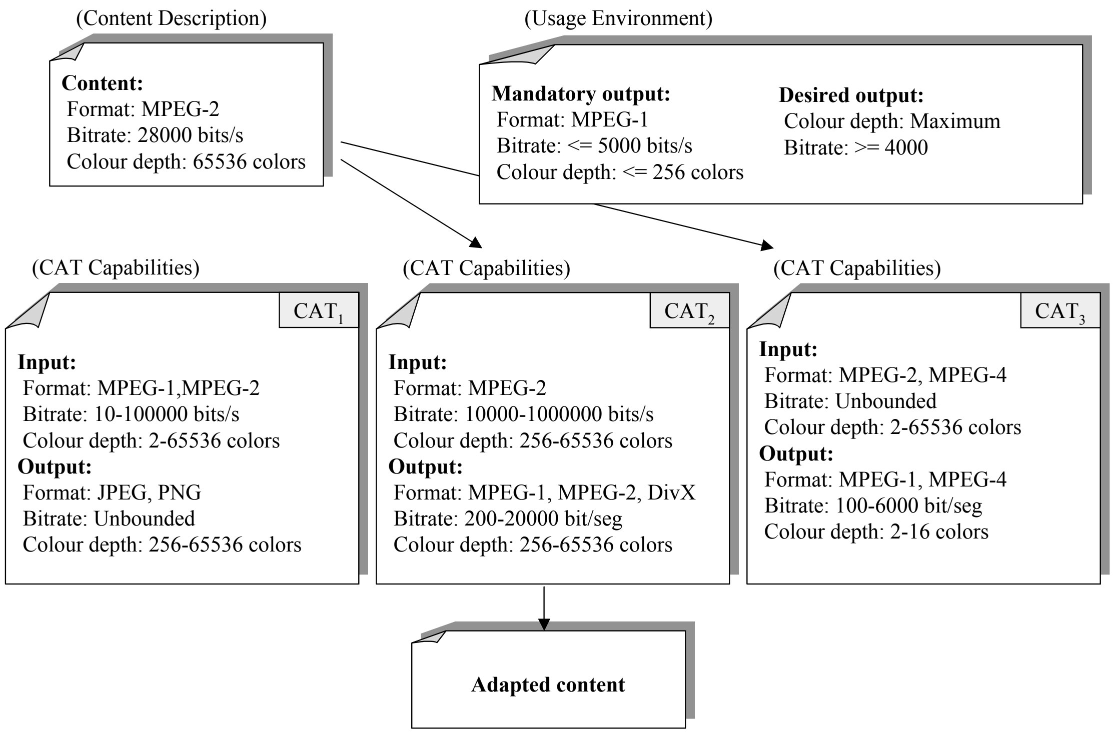

Figure 9: CAT selection to adapt de content

The mandatory usage environment description of the example describes constraints to be imposed to the adapted content (a different video format, a smaller bitrate, and a smaller colour depth). The desired usage environment describes the user preference for maximized colour depth, and enough quality bitrate.

Thus, the DM searches through the CATs' capabilities descriptions and selects the CAT that produces an output that fulfils all these constraints, or at least the mandatory ones. The selection criterion is semantically blind in the sense that one CAT that fulfils all the constraints (mandatory and desirable), at the expense of reducing excessively some parameter not mentioned by the constraints (e.g. screen size), is preferable to another that doesn't fulfils all the constraints given by the usage environment.

If the DM cannot find a CAT capable of fulfilling all the requirements, constraints imposed by the desirable usage environment are incrementally removed (see section 4.2), trying to find a CAT that at least fulfils mandatory constraints.

In the opposite side, if the DM finds various CATs that satisfy all constraints, an optimization process (see section 4.3) is carried out in order to select the one yielding the best user experience.

4. Modelling the DM as a CSP with optimization

This section presents the early approximation followed to model de DM as a CSP (Constraints Satisfaction Problem). After this initial model, section 5 describes some simplifications to the initial model that allow us to reduce significantly the computational cost of the decision process.

Constraints formalize the dependencies of the physical world in terms of logical relations among several unknowns. Methods for solving Constraints Satisfactions Problems (CSP) [29] allows efficient navigation of large search spaces to find out a solution that satisfies given constraints.

In our approach, we propose that content adaptation descriptors can be formalized as variables, and usage environment and CAT Capabilities can determine both the domain of the variables and the constraints between these variables.

In **propositional logic**, a **clause** is a disjunction $\vee$ of **literals** where each literal take a boolean value, and can be negated $\neg$ (e.g. $p \vee \neg q \vee r$). **First order logic** extends propositional logic replacing literals with **predicates** with the form $P(x_1,..,x_n)$ where P is a boolean function and $x_1,..,x_n$ are variables. In first order logic variables are quantified using the universal $\forall$ or existential $\exists$ quantificators.

A **Horn clause** [20] is one clause in which at most one term is positive, a **definite Horn clause** is one in which exactly one term is positive. A **Horn formula** is a conjunction $\wedge$ of Horn clauses. Inference with Horn clauses can be done [20] through the **forward chaining** and **backward chaining** algorithms. Both of these algorithms determine in lineal time where the conclusions of the rules are entailed by the knowledge of the premises

We have found that a Horn definite formula with lineal arithmetic is enough to model all the adaptation constraints. This is due to the fact that usage environment and CAT capabilities can be expressed as equalities and inequalities with only one term in each side of the constraint equation. In this way constraints act as predicates of clauses (e.g. see formula (4) and (5)). This observation allows us to use fast (real time) resolution methods for predicates, like Gauss-Jordan elimination, as well as fast optimization methods like the simplex algorithm.

4.1. Applying mandatory constraints

In the example proposed in Figure 9, based on the content description we could define the following variables: initial video format F_0, initial bitrate B_0, and initial colour depth C_0. Also, based on the usage environment we could define as target variables: terminal accepted format F_n, network maximum bitrate B_n, and terminal maximum accepted colour depth C_n. Thus, based on the media and the terminal of the example we have the following constraints:

$$\begin{array}{ll} F_0 = MPEG\text{-}2 & F_n = MPEG\text{-}1 \\ B_0 = 28000 & B_n \leq 5000 \\ C_0 = 65535 & B_n \geq 4000^* \\ & maximize(C_n)\ ^* \\ & C_n \leq 256 \end{array} \quad (1)$$

* These constraints are desired ones

Based on the existing CAT we could define FI_j, FO_j as sets with, respectively, the available input and output format of each CAT_j in the CAT Capabilities Description file. In the same way we define BI_j, BO_j as the input and output bitrate accepted range of each CAT_j, and CI_j, CO_j as the available colour depth in the input and output of each CAT_j.

For instance, in the previously example we have the following domain for each variable:

$$\begin{array}{ll} FI_1 = \{MPEG\text{-}1, MPEG\text{-}2\} & FO_1 = \{JPEG, PNG\} \\ BI_1 = [10..100000] & BO_1 = Unbounded \\ CI_1 = [2..65536] & CO_1 = [256..65536] \\ & \\ FI_2 = \{MPEG\text{-}2\} & FO_2 = \{MPEG\text{-}1, MPEG\text{-}2, DivX\} \\ BI_2 = [10000..1000000] & BO_2 = [200..20000] \\ CI_2 = [256..65536] & CO_2 = [256..65536] \\ & \\ FI_3 = \{MPEG\text{-}2, MPEG\text{-}4\} & FO_3 = \{MPEG\text{-}1, MPEG\text{-}4\} \\ BI_3 = Unbounded & BO_3 = [100..6000] \\ CI_3 = [2..65536] & CO_3 = [2..16] \end{array} \quad (2)$$

Note that some variables domains in formula (1) are constrained by equalities, and other variables are constrained by inequalities. These inequalities can be transformed in sets as follows:

$$\begin{array}{ll} F_0 = MPEG\text{-}2 & F_n = MPEG\text{-}1 \\ B_0 = 28000 & B_n = [min..5000] \\ C_0 = 65535 & B_n = [4000..max]^* \\ & maximize(C_n)\ ^* \\ & C_n = [\text{min}..256] \end{array} \quad (3)$$

* These constraints are desired ones

Where *min* is a constant minimum value for the parameter (usually cero), and *max* the maximum constant value for this parameter.

Based on the CAT capabilities we can define $CAT_1,...,CAT_3$ as boolean variables which indicate where each CAT_j satisfy the constraints of the problem, and the following set of rules can be built:

$$F_0 \in FI_1 \wedge F_n \in FO_1 \wedge B_0 \in BI_1 \wedge B_n \cap BO_1 \wedge C_0 \in CI_1 \wedge C_n \cap CO_1 \rightarrow CAT_1$$
$$F_0 \in FI_2 \wedge F_n \in FO_2 \wedge B_0 \in BI_2 \wedge B_n \cap BO_2 \wedge C_0 \in CI_2 \wedge C_n \cap CO_2 \rightarrow CAT_2$$
$$F_0 \in FI_3 \wedge F_n \in FO_3 \wedge B_0 \in BI_3 \wedge B_n \cap BO_3 \wedge C_0 \in CI_3 \wedge C_n \cap CO_3 \rightarrow CAT_3 \qquad (4)$$

Note that whenever one parameter take a value and the other takes a range the term appears as a **belonging relation (∈)**, and when both parameters are sets we use an **intersection relation (∩)**.

Now we apply the forward chaining algorithm [20] to determine which CATs can be applied and we reach the solution $\{CAT_1=unknown, CAT_2=true, CAT_3=true\}$. This result indicates that only CAT_2 and CAT_3 are applicable. Observe that CAT_1 is *unknown* and not *false*. To discriminate between $CAT_j=false$ and $CAT_j=unknown$ we need to convert the implication (a sufficient condition) into an equivalence (a sufficient and necessary condition) adding the following group of rules to the problem:

$$\neg(F_0 \in FI_1 \wedge F_n \in FO_1 \wedge B_0 \in BI_1 \wedge B_n \cap BO_1 \wedge C_0 \in CI_1 \wedge C_n \cap CO_1) \rightarrow \neg CAT_1$$
$$\neg(F_0 \in FI_2 \wedge F_n \in FO_2 \wedge B_0 \in BI_2 \wedge B_n \cap BO_2 \wedge C_0 \in CI_2 \wedge C_n \cap CO_2) \rightarrow \neg CAT_2$$
$$\neg(F_0 \in FI_3 \wedge F_n \in FO_3 \wedge B_0 \in BI_3 \wedge B_n \cap BO_3 \wedge C_0 \in CI_3 \wedge C_n \cap CO_3) \rightarrow \neg CAT_3 \qquad (5)$$

This later group of rules increases the cost of solving the problem, providing a more accurately *false/unknown* response, instead of an *unknown* response, from the algorithm. As we are only interested in knowing when $CAT_j=true$, we do not add this group of rules in the implementation, obtaining a solution good enough for our purposes while maintaining the computational cost bounded.

It should be noted that at this point there is not a unique solution to the problem, but a set of solutions equally valid from the mandatory usage environment point of view. Also note that we are only interested in pruning output parameters, because they are the only ones parameters we need to know to execute the CAT. Concretely applying CAT_2 constraints, output variables take the following domains:

$$F_n = MPEG\text{-}1$$
$$B_n = [200..5000] \qquad (6)$$
$$C_n = [256]$$

And applying CAT_3 constraints, output variables take the domains:

$$F_n = MPEG\text{-}1$$
$$B_n = [100..6000] \qquad (7)$$
$$C_n = [2..16]$$

4.2. Applying desirable constraints

If there is not a CAT capable of fulfilling mandatory constraints, the DM fails reporting that it's not possible to adapt the content with the environment given.

Otherwise, if we suppose that the above mandatory constraints have been fulfilled by one or more CATs (as in the previous example), we apply the desirable constraints as we detail below.

Desirable constraints differs from mandatory constraints with regard to three aspects:

- First, desirable constraints must not be completely fulfilled. Even, they could not be fulfilled at all.
- Second, they are ordered by a desirable constraints priority list explained below.
- Third, besides equalities and inequalities constraints, desirable constraints also can have maximization and minimization functions.

The **Desirable Constraints Priority List (DCPL)** is an ordered list of constraints desirable to be applied. This list is ordered from high relevance to lower relevance. The DCPL is by default system defined, so usually the user doesn't need to provide this information, if he/she doesn't want. A graphical user interface can be build to allow the user to edit the DCPL.

Desirable constraints are applied following below algorithm:

1. Take the first constraint of the DCPL and try to fulfil it.
2. If after applying the prior constraint there is not a feasible adaptation that fulfils requirements, ignore these constraints. Else keep this constraint and reduce the range of the domain of variables in formula (6) and (7) accordingly.
3. Repeat this algorithm with the rest of the DCPL.

In the above example there exist two desirable constraints, supposing that they are priorized from top to bottom:

$$\begin{aligned} &maximize(C_n) \\ &B_n = [4000..max] \end{aligned} \qquad (8)$$

After applying these desired constraints, following above proposed algorithm, the output variables of the CAT_2 reach the following domains:

$$\begin{aligned} &F_n = MPEG\text{-}1 \\ &B_n = [4000..5000] \\ &C_n = 256 \end{aligned} \qquad (9)$$

And CAT_3 output variables are restricted to this one domains:

$$\begin{aligned} &F_n = MPEG\text{-}1 \\ &B_n = [4000..5000] \\ &C_n = 16 \end{aligned} \qquad (10)$$

4.3. Content adaptation selection with optimization

In the previous example, several (namely two) CATs reached the desired target, and therefore in order to select only one CAT, a final optimization step is required.

Although in the previous example F_n and C_n are assigned to one value, B_n have a range of values that "a priory" are equally valid from the statement of the problem. We say that **a solution is defined** if there is no variable with more than one possible value.

During this optimization step we pretend to reach two objectives:

- Select the preferred CAT to perform the adaptation
- Select a value for those parameters with more than one feasible value.

That is to say, we pretend to reach one and only one defined solution.

Note that this step can be considered an optimization step because we pretend to select the CAT that provides the best adaptation from those one that fulfil all the mandatory constraints, and as many desirable constraints as possible.

We have proposed to use a **Content Provider Optimization Priority List (CPOPL)** as a list supplied by the media content provider to priorize some variables over others. The CPOPL is a list composed of **optimization elements**, where each optimization element is a constraint defined in a way that only equalities, maximizations, and minimizations are allowed (no unequallies are allowed in order to avoid more that one solution equally valid to the problem).

The algorithm followed by this final optimization step is as follows:

```
1. For each optimization element of the CPOPL
    a. If this optimization element is applicable
        i. Apply the optimization element to each solution
        ii. If there is only one solution, select this solution and abandon
            this loop
2. Use the rest of the CPOPL optimization elements to transform the solution in a
   defined solution
```

At this point it must be observed that the finally selected CAT depends on the CPOPL. If the content provider prefers to maximize colour depth over bitrate, a CPOPL must be defined in the following order:

$$\begin{aligned} &F_n = MPEG\text{-}4 \\ &maximize(C_n) \\ &minimize(B_n) \end{aligned} \tag{11}$$

In this case the first optimization element is ignored because there is no solution where video output format can be *MPEG-4*. The second optimization element selects CAT_2 over CAT_3 because C_n=256 in CAT_2 is bigger than C_n=16 in CAT_3.

If, on the other hand, the content provider prefers to minimize bitrate over colour depth, a CPOPL must be defined in the following order:

$$\begin{aligned}&F_n = MPEG\text{-}4\\&minimize(B_n)\\&maximize(C_n)\end{aligned}\tag{12}$$

In this case, again, the first optimization element is ignored because there is no solution where video output format can be *MPEG-4*. The second optimization element reduces domains of the CAT_2 output to:

$$\begin{aligned}&F_n = MPEG\text{-}1\\&B_n = 4000\\&C_n = 256\end{aligned}\tag{13}$$

And of the CAT_3 output to:

$$\begin{aligned}&F_n = MPEG\text{-}1\\&B_n = 4000\\&C_n = 16\end{aligned}\tag{14}$$

Now, the third optimization element chooses CAT_2 over CAT_3, and this is the selected solution.

Note that in this specific optimization example in both cases the algorithm has selected CAT_2 over CAT_3. This is due to the fact that both CAT have the same limit over the minimum bitrate of B_n=4000.

Also note that, in order to ensure that every parameter of the selected CAT has a unique value, the CPOPL must be complete, that is to say, every parameter must appear (as an equality, maximization or minimization) in the CPOPL.

5. Simplifications to the DM

5.1. Binary CSP

A CSP (Constraints Satisfaction Problem) is a powerful tool because it's capable or representing relations between groups of variables, and finding an assignment to the variables that fulfil all the constraints. Figure 10 represents a general CSP where balls represent variables and lines represent constraints within various variables.

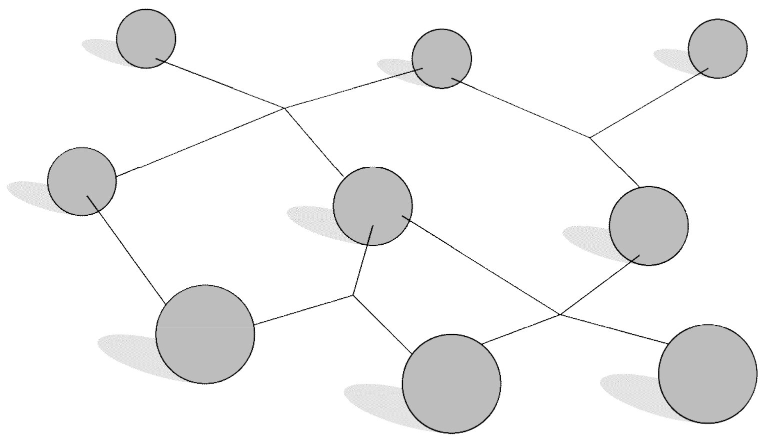

Figure 10: A general CSP

A well knows kind of CSP is a **binary CSP**, that is to say, a CSP where each primitive constraint involves at most two variables. As shows in Figure 11 a binary CSP has the advantage that can be represented as undirected graphs where each variable is represented as a node, and each constraint as an arc: A unary constraint can be represented as a reflexive arc, and a binary constraint as an arc between two nodes.

As next section concludes, we have identified that relations between the variables of our problem are binary. Figure 12 shows the intuition behind this observation. This conclusion will allow us to reduce the cost of finding a solution to our problem drastically.

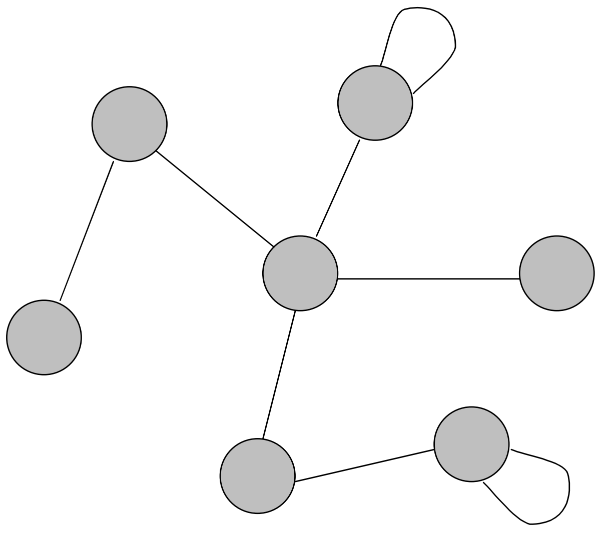

Figure 11: A binary CSP

5.2. Modelling the DM as a CMP

After a more in deep study of the proposed adaptation problem we have found that every constraint between variables can be represented as set intersection pair relations, where pairs are independent among them. For example, in our previous example the format constraint $F_0 \cap FI_1$ is independent with respect to the colour constraint $C_0 \cap CI_1$.

This observation leads us to remodel the adaptation problem as a **Constraints Matching Problem (CMP)**, that is to say, a problem where only intersection relations between pair of sets exists.

Although every relation between pair of variables can be generalized as set intersection constraints, we have defined three groups of constraints in order to improve performance:

- **Equality constraints** (=). This relation exists whenever both variables to match have a unique value. For instance media type is T_0=*Video* and CAT accepted media is TI_1=*Video* too.
- **Set belonging constraints** ($\in$). This relation exists whenever one parameter value is fixed and the other parameter has a set of values. For instance, input video format could be F_0=*MPEG-2* and the CAT accepted video formats are FI_1={*MPEG-1*, *MPEG-2*, *MPEG-4*}.
- **Intersection constraints** ($\cap$). This relation exists whenever both parameters are sets. For instance, the CAT colour depth output could be $CO_1=\{16,256,65536\}$ and the terminal accepted colour depth range is $16 \leq C_T \leq 256$, that is to say C_T=[16..256].

According to this classification parameters can belong to two classes:

- **Value parameters**. They are parameters that have a fixed value.
- **Set parameters**. They are parameters whose feasible values are represented with a set.

As shown in Figure 12, according to entities of the model, we have identified the following types of parameters:

- **Media parameters**. Parameters provided by the media content. As they define specific features of the media, they are value parameters.
- **Adaptation input parameters**. Parameters that represent accepted values by the CAT. Because they represent accepted values they usually are defined as set parameters, but also could be value parameters. As Figure 12 shows, they always relate to media parameters.
- **Adaptation output parameters**. These parameters represent output parameters of the CAT, and could be related to both, the terminal and the network. They usually are set parameters, but could have a single value too.

- **Network parameters**. They define de network status. Sometimes network status is defined by a value parameter (e.g. the type of network), and other times they are set parameters (e.g. accepted bitrate).
- **Terminal parameters**. They define the terminal accepted parameters. Usually they are set parameters, but could be defined using value parameters.

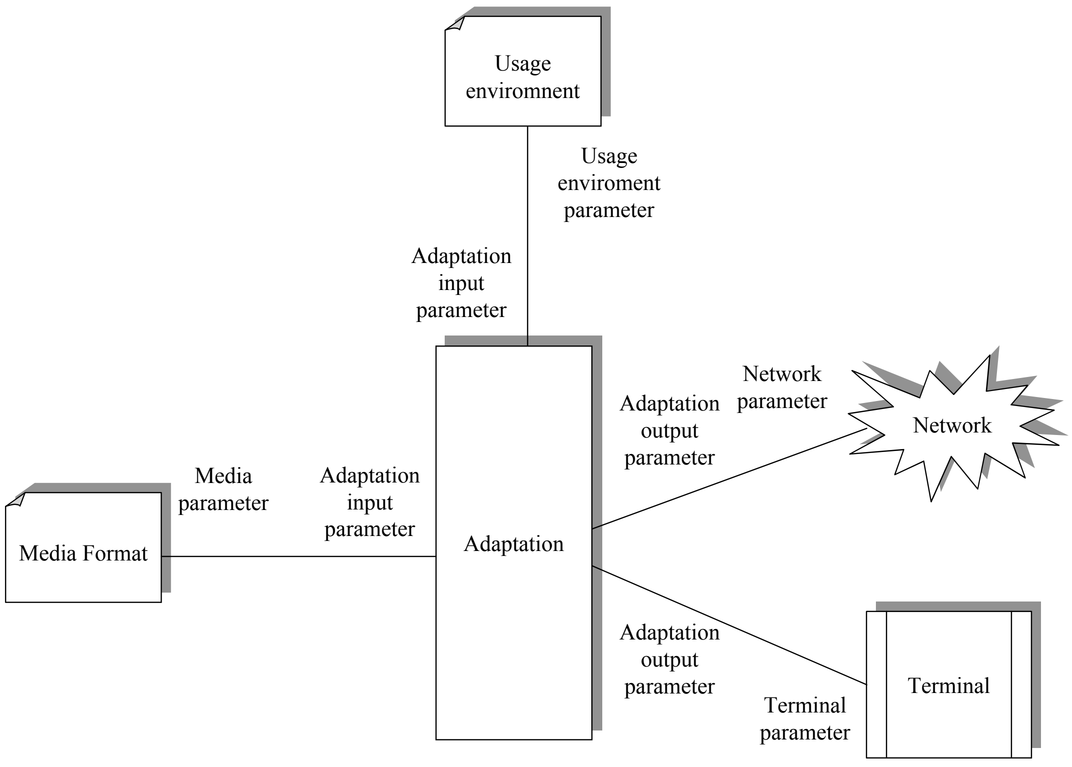

Figure 12: Adaptation relations according to the CMP model

This simplification of the model for the adaptation problem reduces drastically the cost of finding a solution to a problem. In the early model with *n* source variables and *n* target variables 2*n* variables need to be compared with the other ones in groups of 2 variables, 3 variables, ... 2*n*-1 variables, that is to say about $c \approx (2n-1)!$ set intersections amid variables must be tested, with lead to about $(2n-1)!$ evaluations that have to be performed. In contrast, in the simplified model where, due to the fact that there are only pairs relations, and there is *n* source variables and *n* target variables, only *n* set intersections have to be evaluated.

5.3. Evaluating adaptations

It's important to note that each CAT can perform various adaptations. This is due to the fact that each CAT can have different modalities (see Figure 8). Furthermore, each modality can have various input and output media formats.

This is the reason for evaluating adaptations (and not CAT capabilities) in our final approximation. We have created `Adaptation` Java objects to represent each feasible adaptation operations the adaptation engine can perform. Figure 13 shows the structure or these important objects. In this way, if a CAT Capabilities Description file has *m* modalities, *i* input media system formats, and *o* output media system formats, *m*x*i*x*o* `Adaptation` objects would be created.

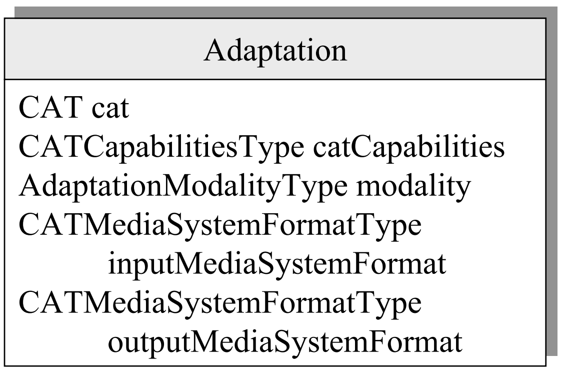
Adaptation
CAT cat
CATCapabilitiesType catCapabilities
AdaptationModalityType modality
CATMediaSystemFormatType inputMediaSystemFormat
CATMediaSystemFormatType outputMediaSystemFormat

Figure 13: The `Adaptation` class

During the generation of `Adaptation` objects the algorithm can produce various `Adaptation` objects with a different `cat` field, but with the same `modality`, `inputMediaSystemFormat` and `outputMediaSystemFormat`. This situation is interpreted as a loosely CAT Capabilities definition because two CAT have been described capable of performing just the same kind of adaptation with the same parameters. Note that this special situation entails that every parameter of the `inputMediaSystemFormat` and `outputMediaSystemFormat` must match.

5.4. Dealing with null parameters

In order to allow media, CAT Capabilities, and Usage Environment Description files manageable, we have realised that a lot of variables that take part in the constraints must be allowed to be omitted, that is to say to be assigned to *null*, signalling that this parameter value is unknown or irrelevant for the actual configuration. In the above situations we have decided that the system must behaves in the following way:

1. If the *null* variable is a source parameter, we interpret this situation as a unknown parameter from the source point of view, and then there is two possible options:
    - If the target parameter is *null* too, we interpret this relation as irrelevant from the target point of view too, and no test is performed.
    - If the target parameter is not *null*, we would consider this situation as a relevant constraint from the target point of view, and, because the source doesn't provides values, we call this situation an **invalid constraint configuration**. Whenever an invalid constraint configuration is found, it's interpreted as an insatisfactible constraint, and we must discard the adaptation. For example, if F_0=*null* and

FI={*GIF*, *BMP*} we have an unsatisfied constraint, and we must discard the adaptation. Note that a configuration can be an invalid constraint configuration from one adaptation of one CAT point of view, but not from other adaptation of other CAT point of view. Thus we must not produce an error when we reach this situation.

2. If the *null* variable is a target parameter, we would assume that there are no constraints regarding the parameter from the target point of view (e.g. if *FI=null* means that the CAT would accept every input formats). Then no test is performed.

5.5. The decision process

Although initially we proposed to search for the best CAT (see section 4), due to the fact that each CAT can have various modalities and various media input and output formats (as explained in section 5.3) we have decided to carry out selection in a adaptation basis, as exposed in this section.

The decision process followed by our final algorithm is taken in five main steps:

1. From each available `CAT` object we generate *as many* `Adaptation` *object as necessary* with the fields illustrated in Figure 13. These `Adaptation` objects stand for each feasible adaptation that each `CAT` can perform. Because a `CAT` can perform various adaptation modalities, and each modality can support various input and output `CATMediaSystemFormat`, usually from one `CAT` we got various `Adaptation` objects.

2. *Filter by mandatory constraints*. In this step we discard `Adaptation` objects whose parameters don't fulfil mandatory constraints. The media format, the usage environment, the terminal, or the network can impose these constraints. We have identified that there is a lot of constraints variables assigned to *null*. As described in section 5.4, in this case we must act as follows:

    1. If the *null* variable is a target parameter, we assume that there are no constraints regarding this parameter. For instance, *FI=null* means that the CAT accepts every kind of input format).
    2. If the *null* variable is a source parameter, and the target parameter is not *null*, we consider this situation as a invalid constraint configuration and we discard the adaptation (e.g. if F_0=*null* and *FI*={*GIF*,*BMP*} we have a unsatisfied constraint.
    3. Finally, if source and target parameters are *null*, we interpret this relation as irrelevant, and no test is performed. For instance, if F_0=*null* and *FI=null* this is a valid configuration where there is no constraint at all.

    If after executing this step there is no `Adaptation` object that fulfils all the constraints we report an error and exit.

3. *Filter by desirable constraints*. If, after step 2, we have more than one `Adaptation` object (which represent different feasible adaptations), we apply preferable constraints following the order imposed by the **Desirable Constraints Priority List (DCPL)**. After applying each constraint we evaluate the number of remaining `Adaptation` objects. If do not remain any `Adaptation` object we discard previous preferable constraints and continue with the next preferable constraint. At the end of this step it is not guaranteed that there remains only one adaptation, neither that every adaptation is defined[1].

4. *Select one defined adaptation*. As explained in section 4.3, during this step we select one adaptation and assign its parameters to defined values. This step can be considered an optimization step because we select one adaptation among the ones that fulfil all the constraints (mandatory and desirable). To choose one of the available adaptations we apply the content provider optimization priority list (CPOPL) in the following way (see also section 4.3):

    1. We select each optimization element from the list trying to apply it to the adaptation objects.
        i. If there remains only one adaptation that fulfils current optimization element, select this adaptation.
        ii. Otherwise apply the rest of the optimization element of the CPOPL.
    2. After reaching a unique adaptation we must ensure that every parameter has been fixed. This is the reason to apply the rest of the optimization elements after reaching a unique adaptation. Only when every parameter of the adaptation is set to a unique value, we can leave this step.

5. *Build the* `Decision` *object*. Finally this step is devoted to build a `Decision` object with the solution, that is to say, indicates the CAT and parameters selected to execute the CAT that the DM must return as the best adaptation to perform.

[1] As explained in section 4.3, a defined adaptation is one where every parameter has a unique value.

6. Results

6.1. Software implementation and test

Software has been implemented in Java and integrated within the CAIN framework.

To ensure final quality of the outgoing product we have developed two kinds of tests:

- **Validation**. At the end of the project we compare outcome product with the initial specifications to validate that the software fulfils original stipulations.

- **Verification**. We have developed a set of test cases (using JUnit) to guarantee that the software responses under different parameter values are the expected ones. These test cases serve two purposes: as a *black box test* to ensure that expected input and output are given, and as a *regression test* to evaluate during future improvements that the software responds in the same way as before the modifications.

Also this software have been successfully integrated in European Commission IST FP6-001765 aceMedia project.

6.2. Content adaptation tools

Currently there exist only five CATs:

- `ImageCAT`, capable of transforming image format and resolution. Input and output images must always be colour images.
- `VFCCAT`, capable of transforming image format and resolution. Always receive a colour image and produce a greyscale image.
- `FfmpegCAT`, capable of transforming video format resolutions and bitrates.
- `RCCAT`, capable of performing a semantic driven video adaptation.
- `J2KBsdCAT`, capable of performing JPEG2000 image BSD (Bitstream Syntax Description) method adaptation.

This group of CAT can be increased in order to reach a wider adaptation scope.

6.3. Adaptation constraints

Currently the DM is capable of choosing the best adaptation according to the following constraints:

- Media type and format must match CAT input types and formats.
- Media size must match CAT accepted input size.
- Media colour depth must match CAT accepted colour depths.
- CAT output type and format must match terminal accepted formats.

- CAT output size must match terminal accepted size.
- CAT output colour depths must match terminal accepted colour depths.
- CAT output bitrate should be bigger than network minimum guaranteed throughput.
- CAT output bitrate must be smaller than network maximum capacity.

Although this group of parameters have enough results for our actual requirements, future CAT additions might need to discriminate according to other parameters.

6.4. Adaptation example

Figure 14 shows an adaptation example where there exists three CAT: `ImageCAT` capable of transforming an image, in different formats and sizes, from colour image to another colour image. `VFCCAT` capable of transforming an image, in different formats and sizes, from colour image to another colour image, and `J2KBsdCAT` capable of transforming a JPEG2000 image to different size using scalable techniques [12].

Also we have two terminals capable of accepting GIF, TIFF, BMP and JPEG images. The unique difference is that first of them have a greyscale screen that only accept 64x48 and 80x80, and the second one have a colour screen that has not specify image size.

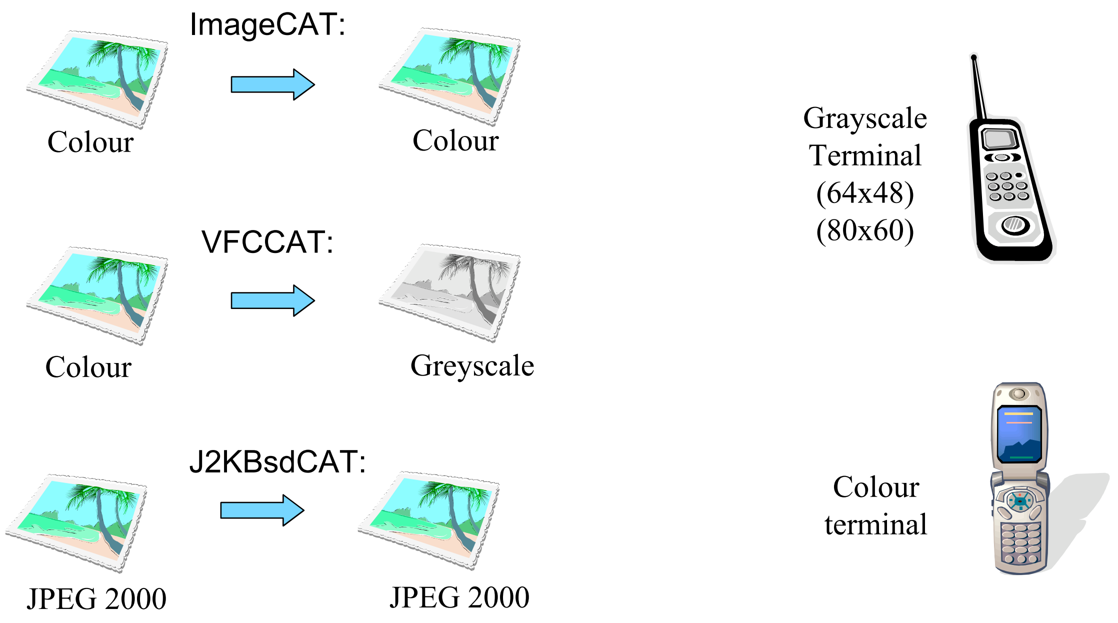

Figure 14: Adaptation example problem

Listing 1, Listing 2 and Listing 3, in Appendix B, show the XML based CAT Capabilities of the CAT of this example, and Listing 5 and Listing 4 show the UED where terminal capabilities are specified. Note again that the greyscale terminal accepts only images at sizes 64x48 and 80x60, while the colour terminal does not specifies accepted image sizes.

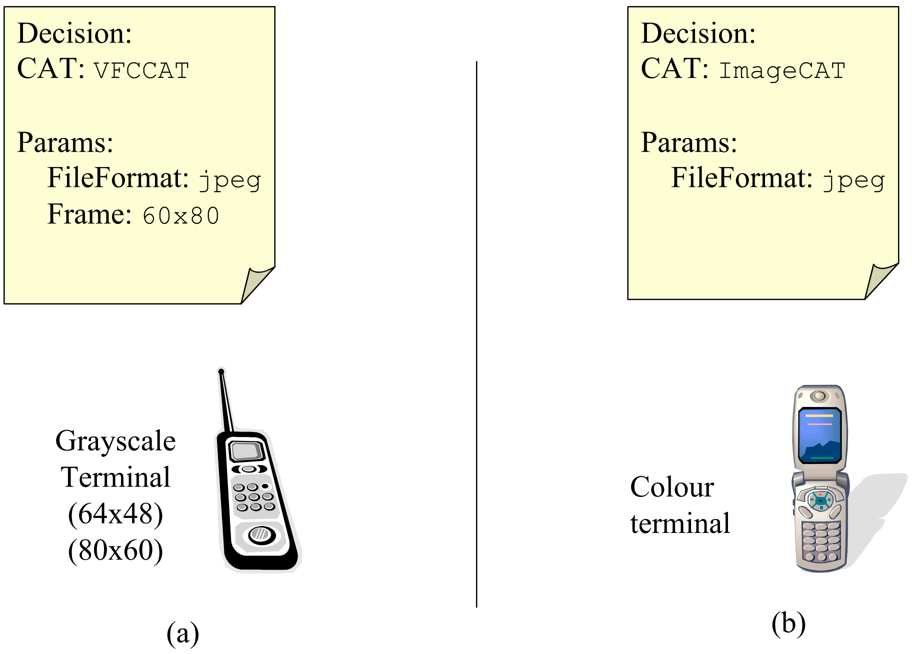

Figure 15: Adaptation example decision

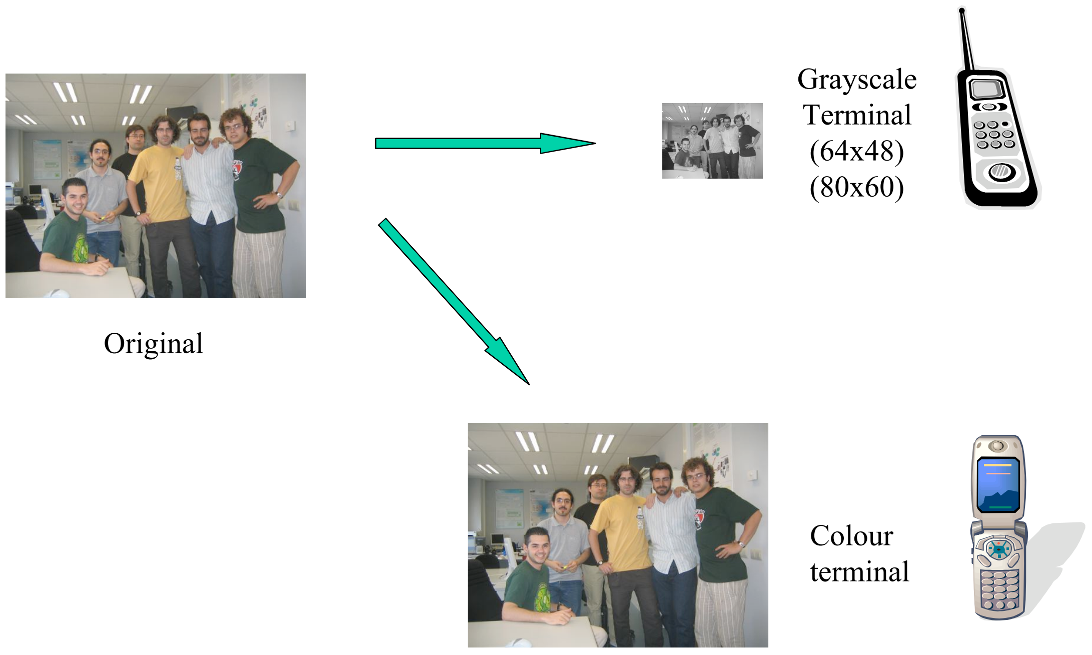

Figure 16: Adaptation example solution

When we execute CAIN with a JPEG image of 1024x768 pixels, the CAT capabilities of the example, and the UED of Listing 4 (greyscale terminal), the decision of Figure 15 (a) is generate. While when we execute CAIN with the same JPEG 1024x768 image and CAT capabilities, but the UED of Listing 5 (colour terminal), the decision of Figure 15 (b) is generated. Note that in both case JPEG file format is the target format selected from the various formats accepted by the terminals. The DM has selected this selection because in this way no format changes in necessary (a desirable constraint). Also note that in the first case the frame size is specified, but in the second case, because the UED doesn't specify the terminal accepted image size, the frame size is assigned to *null*, and the output image size doesn't change. Finally, Figure 16 show the final adaptation produced by CAIN in both cases.

7. Conclusions and future work

7.1. Conclusions

At the end of this work CAIN major objectives have been fulfilled, implemented in software, verified, and validated. At this time CAIN is a multimedia adaptation engine whose more important features are:

- It's a metadata driven adaptation engine compliant with parts of the MPEG-7 MDS and MPEG-21 DIA specifications.

- It's an extensible adaptation engine that can accept and use new CATs at runtime, and without the need to restart the engine.

- It's capable of making decisions based only on existing metadata: It doesn't need any other kind of configuration.

With regard to the DM, the most important objectives have also been achieved:

- To design a flexible architecture for the decision module that is capable of choosing the best adaptation (CAT and parameters).

- To implement this architecture in a software module capable of taking the decision step in real time. This decision is taken according to the content and context descriptions, as well as the CAT capabilities description.

- To verify and validate software response using software engineering techniques, as well as its computational execution cost.

7.2. Future work

Currently CAIN is mature enough to be used in different context where adaptation operations could be needed. Whereas the software is at present completely functional, a lot of work could be done in different areas that we are going to evaluate.

Besides adding more CATs and adaptation parameters, there are new research issues to take into account for future work. We overview the three one we consider more relevant.

7.2.1 Optimization driven by an utility function

As described in section 4.3, the DM optimization step is devoted to finding a defined solution, that is, a unique solution where every parameter is set. Currently the optimization step is driven by the CPOPL (Content Provider Optimization Priority List), which is a priority list where the criterion to prune the adaptation domains that have been set (usually set by the manufacturer).

As described in section 2.3.2, some authors [9] have proposed an adaptation method based on maximizing a vectorial utility function. This method exposes two important advantages over our CPOPL:

- The utility function takes into account different parameters (called resources by the authors), and finds the adaptation that maximizes the pondered sum of them, and maps to the maximum value of the utility function. In contrast, the CPOPL searches the maximum only in a few axes of the utility space. Only when the first optimization element of the CPOPL cannot be applied, or does not find a unique defined solution, the next optimization element is evaluated.

- The utility function can be used to find a pleasant configuration from the point of view of the user, and not only one that fulfil metadata configuration.

7.2.2 Multiple CAT concatenation

Currently the DM selects one and only one CAT from the available ones. This configuration induces various drawbacks that can be solved following a different approach:

- If media content is perfectly adapted to the usage environment requirements, no CAT execution is needed at all. This problem can be partially solved with a **null CAT**, that is to say, a CAT that doesn't modify the content at all. But under this situation a better solution could be not to execute any adaptation at all.

- As [19] proposes, it does not seem realistic that a single adaptation tool will be able to perform all required adaptation steps. Thus, a more difficult problem is what happens when no CAT is capable of performing the adaptation, but a concatenation of CATs does. Under this scenario the best option is to divide the adaptation process in steps where each step perform a partial adaptation of the content.

Scheduling appears to be the best way to find a multiple steps solution to the adaptation process.

7.2.3 Universal constraints description tools

Although actually the DM have been developed following the UED description tools. As described in Appendix A, a more advanced tools for describing constraints and optimization have been defined by MPEG-21 Part-7.

In the general model of the DM detailed in section 4, three main steps are performed: mandatory constraints, desirable constraints, and optimization. This steps map closely to the limitations and optimization constraints found in the UCD model, as well as the `AdaptationQoS` tool. Concretely, constraints map to UCD limitations, and optimization map to both UCD optimizations and the `AdaptationQoS` tool.

8. Publications

Part of this work has produced the following publication:

- Fernando López, José M. Martínez, Víctor Valdés, "Multimedia Content Adaptation within the CAIN framework via Constraints Satisfaction and Optimization", Proceedings of the Fourth International Workshop on Adaptive Multimedia Retrieval-AMR06, Geneva, Switzerland, 27-28 July 2006, in press (to be published also as post-conference LNCS proceedings).

References

[1] A. Vetro. C. Christopoulos, T. Ebrahini (eds.), "Universal Multimedia Access (special issue)", Proc. of IEEE Signal Processing Magazine, 20(2), March 2003.

[2] B. S. Manjunath, Philippe Salembier, Thomas Sikora. "Introduction to MPEG-7: Multimedia Content Description Interface", Ed. Wiley. 2002.

[3] ISO/IEC 15938-5, Information Technology – Multimedia Content Description Interface – Part 5: Multimedia Description Schemes.

[4] Ian S. Burnett, Fernando Pereida, Rik Van de Walle, Rob Koenen. "The MPEG-21 Book", Ed Wiley. 2005

[5] ISO/IEC 21000-7, Information Technology – Multimedia Frameworks – Part 7: Digital Item Adaptation

[6] J.M. Martínez, V. Valdés, J. Bescós, L. Herranz "Introducing CAIN: A Metadata-Driven content Adaptation Manager Integrating Heterogeneous Content Adaptation tools". Proceedings of the WIAMIS'2005. Montreux. April 2005.

[7] Anthony Vetro and Christian Timmerer. "Digital Item Adaptation: Overview of Standardization and Research Activities". Proceeding of IEEE International Transaction on Multimedia, vol. 7, nº 3, June 2005.

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

[9] Shih-Fu Chang, Anthony Vetro. Video Adaptation: Concepts, Technologies, and Open Issues. Special Issue on Advances in Video Coding and Delivery, Proceedings of IEEE, 2005.

[10] J. R. Ohm. "Advances in Scalable Video Coding". Proc. of the IEEE, Vol. 93, Issue 1, January 2005.

[11] N. Sprljan, M. Mrak, G.C.K. Abhayaratne, E. Izquierdo. "A Scalable Coding Framework For Efficient Video Adaptation. Proc. 6th International Workshop on Image Analysis for Multimedia Interactive Services". WIAMIS 2005, Montreux, Switzerland, April 2005.

[12] Xiaolin Wu, Dumitrescu, S., Ning Zhang. "On multirate optimality of JPEG2000 code stream". IEEE Transactions on Image Processing, vol 14, issue 12, pag 2012-2023, Dec. 2005.

[13] Moriya, T., Iwakami, N., Jin, A., Mori, T. " A design of lossy and lossless scalable audio coding", Proceedings of acoustics, speech, and signal processing of ICASSP '00. IEEE International Conference, vol 2, pag 889-892, June 2000.

[14] N. Sprljan, M. Mrak, G. C. K. Abhayaratne, E. Izquierdo, "A scalable coding framework for efficient video adaptation". Proceedings of the WIAMIS'2005. Montreux. April 2005.

[15] Hongjiang Zhang, "Adaptive Content Delivery: A New Application Area For Media Computing Research", Microsoft Research, China.

[16] Y. Wang, J. G. Kim, S.F. Chang. "Content-based utility function prediction for real-time MPEG-4 video transcoding". ICIP 2003, pp 189-192, Barcelona, Spain. September 2003.

[17] Debargha Mukherjee, Eric Delfosse, Jea-Gon Kim, Yong Wang. "Optimal adaptation decision-taking for terminal and network quality-of-service". Proceedings of IEEE transaction on multimedia, vol. 7, pag. 434-462. June 2005.

[18] Jae-Gon Kim, Yong Wang, Shih-Fu Chang. "Content-adaptive utility-based video adaptation". Proceeding of Multimedia and Expo 2003. ICME '03.

[19] Dietmar Jannach, Klaus Leopold, Christian Timmerer, Hermann Hellwagner " A knowledge-based framework for multimedia adaptation" International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies, Springer, 2006.

[20] Stuart Russell, Peter Norvig. "Artificial Intelligence: A modern approach". Ed. Prentice Hall.

[21] I. Burnett et al., "MPEG-21: Goals and Achievements", IEEE Multimedia, 10(6):60-70, Nov.-Dec. 2003.

[22] F. Pereida, I. Burnett, "Universal Multimedia Experiences for Tomorrow", IEEE Signal Processing Magazine, 20(2):63-73, March 2003.

[23] W3C Recommendation, "Composite Capability/Preference Profiles (CC/PP): Structure and vocabularies 1.0". 2004.

[24] J.R. Smith, "Semantic Universal Multimedia Access", in Visual Content Processing and Representation-VLBV03, LNCS Vol. 2849, pp.13-14, Springer-Verlag, 2003.

[25] Víctor Valdés, José M. Martínez, "Content Adaptation Capabilities Description Tool for Supporting Extensibility in the CAIN Framework", en Multimedia Content Representation, Classification and Security-IWMCRS2006, B.Günsel,A.K.Jain, A.M. Tekalp, B. Sankur (eds.), Lecture Notes in Computer Science, Vol. 4105, pp. 8 (in press)

[26] Víctor Valdés, José M. Martínez, "Content Adaptation Tools in the CAIN framework", en Visual Content Processing and Representation, L. Atzori, D.D. Giusto, R. Leonardi, F. Pereira (eds.), Lecture Notes in Computer Science, Vol. 3893, Springer Verlag, 2006, pp. 9-15

[27] J.M. Martínez, V. Valdés, L. Herranz, J. Bescós, "A Simple Profile for MPEG-21 Usage Environment description tools", Doc. ISO/MPEG M11239, MPEG Palma de Mallorca Meeting, Octubre 2004.

[28] E. P. Vivancos Rubio. "Incorporación de un sistema basado en reglas en un entorno de tiempo real". Tesis Doctoral. Dpto. de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia. 2004

[29] Marriott, Kim. "Programming with constraints, an introduction". Ed. MIT Press.

[30] R.E. Steuer. "Multiple criteria optimization: Theory, computation and application". Krieger, Publishing Company. August 1996.

[31] P. E. Gill, W. Murray, and M. H, Wright "Practical Optimization". New York: Academic. 2003.

Appendix A: Advanced DIA tools

UED is a basic DIA description tool. Nevertheless, more advanced functionalities are also supported within DIA. As state in section 2.5.1, to represent constraints and optimizations, two of the description tools proposed by the MPEG-21 DIA [5] standard are: **Terminal and Network Quality of Service**, represented by the `AdaptationQoS` description scheme, and **Universal Constraints Description (UCD)** tools.

Using the UCD tools, it's possible to describe various types of constraints that impact the adaptation process. End users, or their applications may, for example, constrain the resolution of the rendering device or the display size, in order to satisfy the needs of the application itself. On the other hand, DI providers may want to restrict the way that their resources are adapted, for example, a minimum level of quality may be imposed. Besides specifying such constraints, several optimization criteria may also be specified to guide the adaptation decision.

The adaptations constraints can be specified, not only on the resource as a whole, but also differentiated with respect to individual units of the resource corresponding to logical partitions such as Group Of Pictures (GOP), Regions Of Interest (ROI), titles, frames, an so on. Such units are referred as **adaptation units**.

1. Push and pull mode

The actual values of the variables defined in the UCD model can be related to:

- The DI itself.
- The usage environment of the DI.

In the former case, the possible values to which the constraints are applied can be obtained from appropriate MPEG-7 descriptions, or the `AdaptationQoS` [5] (described in section 3 of this appendix). In the latter case the UCD is used to further constrain the usage environment.

From this observation, it follows that an instance of the UCD tools can travel both upstream from the consumer to the adaptation engine, which is referred as the push mode, and downstream from the content provider to the adaptation engine, which is referred as the pull mode. This observation gives rise two modes in the UDC tools:

- **Push mode** (upstream) that is used by the DI provider to further constrain the usage of the DI that must be satisfied for any adaptation of a resource.

- **Pull mode** (downstream) that cover the case where the DI consumer want to further constrain the usage environment of the DI. Specifically, constrain on the terminal, network and user characteristics may be imposed. The primary advantage of this mode is the full flexibility of the DI consumer to retrieve the DI in a more controller manner. For example, it is possible for the DI consumer to render the visual resource in a screen at a smaller resolution than that of the display, while another DI consumer displays a qualitatively better image in exchange of waiting for the image to be downloaded.

2. Types of constraints

Constraints of the UCD model can be divided in two groups:

- o **Limitation constraints**, which restricts the solution space of a set of possible adaptation decisions. These mathematical expressions are proposed by the standard to be described using a reverse polish notation. According to the standard, this kind of constraints has to be evaluated to either *true* or *false*.

- o **Optimization constraints**, which provide optimal decision for the adaptation of DIs from the utility point of view.

Limitation constraints prune the solution space of decisions. In practice, however, this solution space could be still very large. Therefore, one or more optimization constraints can be formulated within the UCD in order to provide an optimal decision.

3. Terminal and network quality of service

Terminal and network quality of service (QoS) address the problem of media resource adaptation to constraints imposed by the terminal and/or the network. The `AdaptationQoS` description scheme DIA tool specifies the relationship between constraints, feasible adaptation operations satisfying this constraints, and associated utilities (qualities) [18]. Therefore, the `AdaptationQoS` tool lets an adaptation engine know what adaptation operations are feasible for satisfying the given constraints, and the quality resulting from each adaptation. In this way terminal and network QoS management is efficiently achieved by adaptation of media resources to constraints. Note that QoS is used loosely and does not correspond to network level guarantees.

The DIA tools, `AdaptationQoS` and UCD used in combination support the decision taking adaptation engine proposed in section 2.5.2. The `AdaptationQoS` description consist of two main components:

- o **IOPins** that represent variables, and provide an interface for input and output values, which allows the interconnection of different modules.

- o **Modules**, which provide a means to select an output value given one or several input values. In mathematics, a module is comparable to a function with several input variables represented by the IOPins. In a programming language, a module is similar to a function returning a scalar value.

In order to enable linking of the UCD to the right IOPins in `AdaptationQoS`, DIA create a number of dictionaries named **classification schemes**. The `AdaptationQoS` associated the IOPins it defines with the classification scheme terms that are closest in semantic, while the UCD creator uses the same classification scheme terms to specify the problem, rather that using identifiers of the IOPins directly. The ADTE simply performs a textual match of the classification scheme terms to know how the variables specified in the UCD map to the IOPins.

4. UCD-based decision taking

UCD-based decision taking is performed on variables that are related to the resource or the usage environment characteristics. In this context, the set of variables is denoted by the vector $I=\{i_0,i_1,...i_n\}$. The purpose of the UCD is to specify a constraints satisfaction with optimization problem involving these variables.

The UCD has o functions with the form $O_j(I)$ $j=1,..,o$ called **optimization constraints**, along with l boolean functions $L_k(I)$ $k=1,..,l$ called **limitation constraints**. Both are used together to specify the following constraints satisfaction with optimization problem involving I:

Maximize $O_j(I)$ $j=1,..,o$ subject to $L_k(I)=true$ $k=1,..,l$

Let I^* represent a solution to the above problem, the syntax of the UCD allows any number of optimization constraints to be specified:

- For $n=0$, that is, no optimization constraints are specified, any solution I^* in the feasible solution space where the limitation constraints evaluate to *true* is acceptable.

- For $n=1$, that is, a single optimization constraint is specified, a single objective optimization problem is defined, thus we regard as acceptable any solution that maximized the single optimization metric within the feasible solution space.

- For $n>1$, that is, a multicriteria (also knows as multiobjective) optimization problem is defined [30][31]. Any solution I^* in the Pareto optimal set included in the feasible region is acceptable. In multicriteria optimization literature, a set of point in the feasible region is said to be Pareto optimal if in moving from one point in the feasible region to another point in the feasible region, any improvement in one of the optimization metrics from it current value would cause at least one of the other optimization metrics to deteriorate from it current value. In other words, this is the set of best solutions that could be achieved without affecting another metric.

The above approach is applied on a single adaptation unit. However, if multiple adaptation units need to be considered, this approach is repeated for each adaptation unit.

Appendix B: CAT and UED examples

Following listings corresponds to CAT capabilities and UED example files used in section 6.4.

```
<?xml version="1.0" encoding="UTF-8"?>
<cat xmlns="acemedia:cme:cat" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="acemedia:cme:cat     file:./CATCapabilities.xsd">
    <name>ImageCAT</name>
    <description>Resize the input image</description>

    <!--BMP visual elementary stream adaptation capabilities description-->
    <ElementaryStreamFormat type="Image" id="BMP">
              <CommonParameters>
                     <VisualCoding>
                             <Frame>
                                   <height>
                                          <range>
                                          <from>10</from>
                                          <to>2000</to>
                                          </range>
                                   </height>
                                   <width>
                                          <range>
                                          <from>10</from>
                                          <to>2000</to>
                                          </range>
                                   </width>
                             </Frame>
                     </VisualCoding>
              </CommonParameters>
    </ElementaryStreamFormat>

    <!--JPEG visual elementary stream adaptation capabilities description-->
    <ElementaryStreamFormat type="Image" id="JPEG">
              <CommonParameters>
                     <VisualCoding>
                             <Frame>
                                   <height>
                                          <range>
                                          <from>10</from>
                                          <to>2000</to>
                                          </range>
                                   </height>
                                   <width>
                                          <range>
                                          <from>10</from>
                                          <to>2000</to>
                                          </range>
                                   </width>
                             </Frame>
                     </VisualCoding>
              </CommonParameters>
    </ElementaryStreamFormat>

     <!--TIFF visual elementary stream adaptation capabilities description-->
    <ElementaryStreamFormat type="Image" id="TIFF">
              <CommonParameters>
                     <VisualCoding>
                             <Frame>
                                   <height>
                                          <range>
                                          <from>10</from>
                                          <to>2000</to>
                                          </range>
                                   </height>
                                   <width>
                                          <range>
                                          <from>10</from>
                                          <to>2000</to>
                                          </range>
                                   </width>
                             </Frame>
```

```
                </VisualCoding>
            </CommonParameters>
    </ElementaryStreamFormat>

    <!--PNM visual elementary stream adaptation capabilities description-->
    <ElementaryStreamFormat type="Image" id="PNM">
            <CommonParameters>
                <VisualCoding>
                        <Frame>
                            <height>
                                <range>
                                <from>10</from>
                                <to>2000</to>
                                </range>
                            </height>
                            <width>
                                <range>
                                <from>10</from>
                                <to>2000</to>
                                </range>
                            </width>
                        </Frame>
                </VisualCoding>
            </CommonParameters>
    </ElementaryStreamFormat>


     <CommonMediaSystemFormats> <!--common Media System formats: both input
                               and ouput formats-->
            <MediaSystemFormat id="BMP-Media">
                <FileFormat>BMP</FileFormat>
          <Extension>bmp</Extension>
          <VisualCoding>
              <CodingFormatRef idref="BMP"/>
          </VisualCoding>
       </MediaSystemFormat>

       <MediaSystemFormat id="JPEG-Media">
                <FileFormat>JPEG</FileFormat>
          <Extension>jpeg</Extension>
          <VisualCoding>
              <CodingFormatRef idref="JPEG"/>
          </VisualCoding>
       </MediaSystemFormat>

      <MediaSystemFormat id="TIFF-Media">
                <FileFormat>TIFF</FileFormat>
          <Extension>tiff</Extension>
          <VisualCoding>
              <CodingFormatRef idref="TIFF"/>
          </VisualCoding>
       </MediaSystemFormat>

            <MediaSystemFormat id="PNM-Media">
                <FileFormat>PNM</FileFormat>
          <Extension>pnm</Extension>
          <VisualCoding>
              <CodingFormatRef idref="PNM"/>
          </VisualCoding>
       </MediaSystemFormat>
    </CommonMediaSystemFormats>


    <AdaptationModalities>
        <AdaptationModality>
            <Mode href="acemedia:cme:cat:cs:RC-CAT-Modes">
                <Name xmlns="urn:mpeg:mpeg7:schema:2001">Keyframe
                                      Replication Sumarization</Name>
            </Mode>
            <MediaSystemRefInput idref="BMP-Media"/>
            <MediaSystemRefInput idref="JPEG-Media"/>
            <MediaSystemRefInput idref="TIFF-Media"/>
            <MediaSystemRefInput idref="PNM-Media"/>

            <MediaSystemRefOutput idref="BMP-Media"/>
            <MediaSystemRefOutput idref="JPEG-Media"/>
            <MediaSystemRefOutput idref="TIFF-Media"/>
            <MediaSystemRefOutput idref="PNM-Media"/>
        </AdaptationModality>
    </AdaptationModalities>
</cat>
```

Listing 1: ImageCAT capabilities

```
<?xml version="1.0" encoding="UTF-8"?>
<cat xmlns="acemedia:cme:cat" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="acemedia:cme:cat     file:./CATCapabilities.xsd">
    <name>VFCCAT</name>
    <description> Turn color image to graylevel image </description>

    <!--BMP visual elementary stream adapation capabilities description-->
    <ElementaryStreamFormat type="Image" id="BMP">
               <InputParameters>
                      <VisualCoding>
                               <Format colorDomain="color">
                               </Format>
                              <Frame>
                                    <height>
                                           <range>
                                           <from>10</from>
                                           <to>2000</to>
                                           </range>
                                    </height>
                                    <width>
                                           <range>
                                           <from>10</from>
                                           <to>2000</to>
                                           </range>
                                    </width>
                              </Frame>
                      </VisualCoding>
               </InputParameters>
                <OutputParameters>
                      <VisualCoding>
                               <Format colorDomain="graylevel">
                               </Format>
                              <Frame>
                                    <height>
                                           <range>
                                           <from>10</from>
                                           <to>2000</to>
                                           </range>
                                    </height>
                                    <width>
                                           <range>
                                           <from>10</from>
                                           <to>2000</to>
                                           </range>
                                    </width>
                              </Frame>
                      </VisualCoding>
               </OutputParameters>
    </ElementaryStreamFormat>

    <!--JPEG visual elementary stream adapation capabilities description-->
    <ElementaryStreamFormat type="Image" id="JPEG">
               <InputParameters>
                      <VisualCoding>
                               <Format colorDomain="color">
                               </Format>
                              <Frame>
                                    <height>
                                           <range>
                                           <from>10</from>
                                           <to>2000</to>
                                           </range>
                                    </height>
                                    <width>
                                           <range>
                                           <from>10</from>
                                           <to>2000</to>
```

```
                                        </range>
                                </width>
                        </Frame>
                </VisualCoding>
        </InputParameters>
         <OutputParameters>
                <VisualCoding>
                         <Format colorDomain="graylevel">
                         </Format>
                        <Frame>
                                <height>
                                        <range>
                                        <from>10</from>
                                        <to>2000</to>
                                        </range>
                                </height>
                                <width>
                                        <range>
                                        <from>10</from>
                                        <to>2000</to>
                                        </range>
                                </width>
                        </Frame>
                </VisualCoding>
        </OutputParameters>
</ElementaryStreamFormat>

 <!--TIFF visual elementary stream adapation capabilities description-->
<ElementaryStreamFormat type="Image" id="TIFF">
        <InputParameters>
                <VisualCoding>
                         <Format colorDomain="color">
                         </Format>
                        <Frame>
                                <height>
                                        <range>
                                        <from>10</from>
                                        <to>2000</to>
                                        </range>
                                </height>
                                <width>
                                        <range>
                                        <from>10</from>
                                        <to>2000</to>
                                        </range>
                                </width>
                        </Frame>
                </VisualCoding>
        </InputParameters>
         <OutputParameters>
                <VisualCoding>
                         <Format colorDomain="graylevel">
                         </Format>
                        <Frame>
                                <height>
                                        <range>
                                        <from>10</from>
                                        <to>2000</to>
                                        </range>
                                </height>
                                <width>
                                        <range>
                                        <from>10</from>
                                        <to>2000</to>
                                        </range>
                                </width>
                        </Frame>
                </VisualCoding>
        </OutputParameters>
</ElementaryStreamFormat>

  <!--PNM visual elementary stream adapation capabilities description-->
<ElementaryStreamFormat type="Image" id="PNM">
        <InputParameters>
                <VisualCoding>
                         <Format colorDomain="color">
                         </Format>
                        <Frame>
                              <height>
                                     <range>
                                     <from>10</from>
                                     <to>2000</to>
                                     </range>
                              </height>
                              <width>
                                     <range>
                                     <from>10</from>
                                     <to>2000</to>
                                     </range>
                              </width>
                        </Frame>
                </VisualCoding>
         </InputParameters>
          <OutputParameters>
                <VisualCoding>
                         <Format colorDomain="graylevel">
                         </Format>
                        <Frame>
                              <height>
                                     <range>
                                     <from>10</from>
                                     <to>2000</to>
                                     </range>
                              </height>
                              <width>
                                     <range>
                                     <from>10</from>
                                     <to>2000</to>
                                     </range>
                              </width>
                        </Frame>
                </VisualCoding>
         </OutputParameters>
</ElementaryStreamFormat>


 <CommonMediaSystemFormats> <!--common Media System formats:
                               both input and ouput formats-->
         <MediaSystemFormat id="BMP-Media">
                <FileFormat>BMP</FileFormat>
       <Extension>bmp</Extension>
       <VisualCoding>
           <CodingFormatRef idref="BMP"/>
       </VisualCoding>
    </MediaSystemFormat>

    <MediaSystemFormat id="JPEG-Media">
                <FileFormat>JPEG</FileFormat>
       <Extension>jpeg</Extension>
       <VisualCoding>
           <CodingFormatRef idref="JPEG"/>
       </VisualCoding>
    </MediaSystemFormat>

    <MediaSystemFormat id="TIFF-Media">
                <FileFormat>TIFF</FileFormat>
       <Extension>tiff</Extension>
       <VisualCoding>
           <CodingFormatRef idref="TIFF"/>
       </VisualCoding>
    </MediaSystemFormat>


   <MediaSystemFormat id="PNM-Media">
                <FileFormat>PNM</FileFormat>
       <Extension>pnm</Extension>
       <VisualCoding>
           <CodingFormatRef idref="PNM"/>
       </VisualCoding>
    </MediaSystemFormat>
</CommonMediaSystemFormats>
```

```
    <AdaptationModalities>
        <AdaptationModality>
            <Mode href="acemedia:cme:cat:cs:RC-CAT-Modes">
                <Name xmlns="urn:mpeg:mpeg7:schema:2001">
                      Turn in graylevel color </Name>
            </Mode>
            <MediaSystemRefInput idref="BMP-Media"/>
            <MediaSystemRefInput idref="JPEG-Media"/>
            <MediaSystemRefInput idref="TIFF-Media"/>
            <MediaSystemRefInput idref="PNM-Media"/>

            <MediaSystemRefOutput idref="BMP-Media"/>
            <MediaSystemRefOutput idref="JPEG-Media"/>
            <MediaSystemRefOutput idref="TIFF-Media"/>
            <MediaSystemRefOutput idref="PNM-Media"/>
        </AdaptationModality>
    </AdaptationModalities>
</cat>
```

Listing 2: VFCCAT capabilities

```
<?xml version="1.0" encoding="UTF-8"?>
<cat xmlns="acemedia:cme:cat" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="acemedia:cme:cat     file:./CATCapabilities.xsd">
    <name>J2KBsd</name>
    <description>
       JPEG2000 image BSD (Bitstream Syntax Description) method adaptation
    </description>

    <ElementaryStreamFormat type="Image" id="input-JP2K">
           <CommonParameters>
                  <VisualCoding>
                         <Frame>
                                <height>
                                       <range>
                                              <from>4</from>
                                              <to>10000</to>
                                       </range>
                                </height>
                                <width>
                                       <range>
                                              <from>4</from>
                                              <to>10000</to>
                                       </range>
                                </width>
                         </Frame>
                  </VisualCoding>
              </CommonParameters>
    </ElementaryStreamFormat>

    <ElementaryStreamFormat type="Image" id="output-JP2K">
           <CommonParameters>
                  <VisualCoding>
                         <Frame>
                                <height>
                                       <percent>100</percent>
                                       <percent>50</percent>
                                       <percent>25</percent>
                                       <percent>12.5</percent>
                                       <percent>6.25</percent>
                                       <percent>6.25</percent>
                                       <percent>3.125</percent>
                                </height>
                                <width>
                                       <percent>100</percent>
                                       <percent>50</percent>
                                       <percent>25</percent>
                                       <percent>12.5</percent>
                                       <percent>6.25</percent>
                                       <percent>6.25</percent>
                                       <percent>3.125</percent>
                                </width>
                         </Frame>
                  </VisualCoding>
              </CommonParameters>
       </ElementaryStreamFormat>

    <InputMediaSystemFormats>
           <MediaSystemFormat id="JPEG2000-input">
                  <FileFormat>JPEG2000</FileFormat>
                  <Extension>jp2</Extension>
                  <VisualCoding>
                         <CodingFormatRef idref="input-JP2K"/>
                  </VisualCoding>
           </MediaSystemFormat>
    </InputMediaSystemFormats>

    <OutputMediaSystemFormats>
           <MediaSystemFormat id="JPEG2000-output">
                  <FileFormat>JPEG2000</FileFormat>
                  <Extension>jp2</Extension>
                  <VisualCoding>
                         <CodingFormatRef idref="ouput-JP2K"/>
                  </VisualCoding>
           </MediaSystemFormat>
    </OutputMediaSystemFormats>

    <AdaptationModalities>
        <AdaptationModality>
            <Mode href="acemedia:cme:cat:cs:J2KBsd-Modes">
                <Name xmlns="urn:mpeg:mpeg7:schema:2001">
                     Bitstream Syntax Description based adaptation</Name>
            </Mode>
            <MediaSystemRefInput idref="JPEG2000-input"/>

            <MediaSystemRefOutput idref="JPEG2000-output"/>
        </AdaptationModality>
    </AdaptationModalities>
  </cat>
```

Listing 3: J2KBsdCAT capabilities

```
<?xml version="1.0" encoding="UTF-8"?>
<DIA xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS" xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:mpeg:mpeg21:2003:01-DIA-NS">
<Description xsi:type="UsageEnvironmentPropertyType">
<UsageEnvironmentProperty xsi:type="TerminalsType">
<Terminal>
<TerminalCapability xsi:type="CodecCapabilitiesType">
<Display id="primary_display">
<DisplayCapability xsi:type="DisplayCapabilityType" colorCapable="false">
<Mode>
<Resolution horizontal="64" vertical="48"/>
</Mode>
<Mode>
<Resolution horizontal="80" vertical="60"/>
</Mode>
</DisplayCapability>
</Display>
<Decoding xsi:type="ImageCapabilitiesType">
<Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:4">
<mpeg7:Name xml:lang="en">JPEG</mpeg7:Name>
</Format>
</Decoding>
<Decoding xsi:type="ImageCapabilitiesType">
<Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:4">
<mpeg7:Name xml:lang="en">GIF</mpeg7:Name>
</Format>
</Decoding>
<Decoding xsi:type="ImageCapabilitiesType">
<Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:4">
<mpeg7:Name xml:lang="en">TIFF</mpeg7:Name>
</Format>
</Decoding>
<Decoding xsi:type="VideoCapabilitiesType">
<Format
href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.1.2">
<mpeg7:Name xml:lang="en">MPEG-4</mpeg7:Name>
```

```
</Format>
</Decoding>
<Encoding xsi:type="ImageCapabilitiesType">
<Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:4">
<mpeg7:Name xml:lang="en">BMP</mpeg7:Name>
</Format>
</Encoding>
<Encoding xsi:type="AudioCapabilitiesType">
<Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:6">
<mpeg7:Name xml:lang="en">AMR</mpeg7:Name>
</Format>
</Encoding>
<Encoding xsi:type="VideoCapabilitiesType">
<Format
href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.1.2">
<mpeg7:Name xml:lang="en">MPEG-4</mpeg7:Name>
</Format>
</Encoding>
</TerminalCapability>
</Terminal>
</UsageEnvironmentProperty>
</Description>
</DIA>
```

Listing 4: Greyscale terminal example

```
<?xml version="1.0" encoding="UTF-8"?>
<DIA xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS" xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:mpeg:mpeg21:2003:01-DIA-NS">
<Description xsi:type="UsageEnvironmentPropertyType">
<UsageEnvironmentProperty xsi:type="TerminalsType">
<Terminal>
<TerminalCapability xsi:type="CodecCapabilitiesType">
<Decoding xsi:type="AudioCapabilitiesType">
<Format
href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.4">
<mpeg7:Name xml:lang="en">MP3</mpeg7:Name>
</Format>
<Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:6">
<mpeg7:Name xml:lang="en">AMR</mpeg7:Name>
</Format>
</Decoding>
<Decoding xsi:type="ImageCapabilitiesType">
<Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:4">
<mpeg7:Name xml:lang="en">JPEG</mpeg7:Name>
</Format>
</Decoding>
<Decoding xsi:type="VideoCapabilitiesType">
<Format
href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.1.2">
<mpeg7:Name xml:lang="en">MPEG-4</mpeg7:Name>
</Format>
</Decoding>
<Encoding xsi:type="AudioCapabilitiesType">
<Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:6">
<mpeg7:Name xml:lang="en">AMR</mpeg7:Name>
</Format>
</Encoding>
<Encoding xsi:type="VideoCapabilitiesType">
<Format
href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.1.2">
<mpeg7:Name xml:lang="en">MPEG-4</mpeg7:Name>
</Format>
</Encoding>
</TerminalCapability>
</Terminal>
</UsageEnvironmentProperty>
</Description>
</DIA>
```

Listing 5: Colour terminal example

Appendix C: Acronyms and symbols

Acronyms	**Description**
ADTE	Adaptation Decision Taking Engine
ARU	Adaptation Resource Utility
BAE	Bitstream Adaptation Engine
BSD	Bitstream Syntax Description
CAIN	Content Adaptation INtegrator
CAT	Content Adaptation Tool
CPOPL	Content Provider Optimization Priority List
CSP	Constraints Satisfaction Problem
CMP	Constraints Matching Problem
DCPL	Desirable Constraints Priority List
DI	Digital Item
DIA	Digital Item Adaptation
DM	Decision Module
DS	Description Scheme
gBSD	General BSD
GOP	Group Of Pictures
JPEG	Join Pictures Expert Group
MDS	Media Description Scheme
MPEG	Moving Pictures Expert Group
ROI	Region Of Interest
PSNR	Peak Signal to Noise Ratio
QoS	Quality of Service
SVC	Scalable Video coding
UCD	Universal Constraints Description
UED	Usage Environment Descriptor
UF	Utility Function
UMA	Universal Multimedia Access
XML	eXtensible Markup Language

Table 1: Acronyms

Symbols	**Description**
$\neg$	Logical not
$\wedge$	Logical and
$\vee$	Logical or
$\rightarrow$	Implication
$\Leftrightarrow$	Equivalence
$\in$	Belonging relation
$\cap$	Set intersection
$\cup$	Set conjunction

Table 2: Symbols