UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

CONTRIBUTIONS TO MULTIMEDIA ADAPTATION WITHIN THE MPEG-21 FRAMEWORK

PHD THESIS

Fernando López Hernández

under the supervision of

José M. Martínez Sánchez (EPS UAM)
Narciso García Santos (ETSIT UPM)

Madrid, August 2010

Work financed by the European Commission (IST-IP-001765 - aceMedia, IST-FP6-027685 - MESH), the Spanish Government (TEC2007-65400 - SemanticVideo), the Ministerio de Educación y Ciencia of Spain (through the FPU fellowship grant issued to the student) and the Comunidad de Madrid (S-0505/TIC-0223 - ProMultiDis-CM).

Part of this work was done while visiting the following foreign institutions:

- From July 1, 2007 to September 30, 2007. Multimedia Communication research group (MMC). Klagenfurt University (Austria). C. Timmerer, H. Hellwagner, D. Jannach. Development of decision methods to automatically generate sequences of conversions that adapt multimedia content to the usage environment.
- From September 14, 2009 to December 15, 2009. I-Lab Multimedia and DSP Research Group. University of Surrey (UK). Marta Mrak, Hemantha Kodikara Arachchi. Integration of Scalable Video Coding adaptation in the CAIN-21 framework.

To all those who enjoy multimedia.

Acknowledgments

To begin, I must acknowledge my gratitude to José M. Martínez Sánchez who has been giving his generous support to this research for the last six years. He has aided me in many ways: by helping me obtain financial resources, inviting me to collaborate on very high-level research European projects and reviewing each and every one of my writings. In this regard, I would also like to thank Narciso García Santos for his expert advice, revisions and for believing in and supporting the research presented in this thesis. Thank you both for guiding me during these years. This thesis would not exist without your continuous help, advice, patience and professionalism.

Second, I have to thank the Video Processing and Understanding (VPU) Lab, the research group in which my research interest has progressively grown, and its members: Jesús, Miguel Angel Javi, Luis I, Luis II, Fabrizio, Víctor Fernández, Víctor Valdés, Juan Carlos, Álvaro I, Álvaro II, Álvaro III, Marcos and Virginia. I thank Jesús Bescós Cano in particular for letting me attend his classes and sharing his knowledge, ideas, and resources.

Third, I would like to thank Hermann Hellwagner, Christian Timmerer, and Dietmar Jannach for their support and help during my stay in Klagenfurt. Back in the summer of 2007 I initiated the innovative multimedia adaptation decision method that CAIN-21 uses. I thank them for revising all the versions of the papers in which these approaches have been put forth and explained.

Fourth, I would like to thank Safak Dogan, Gokce Nur, Marta Mrak, Hemantha Kodikara Arachchi for their support during my second stay in their lab. They have expert knowledge of the quality-based decision methods, now incorporated into CAIN-21.

CONTRIBUCIONES A LA ADAPTACIÓN MULTIMEDIA EN EL ENTORNO MPEG-21

Resumen

La investigación actual recoge una gran variedad de métodos de adaptación y tecnologías subyacentes. Después de comparar diferentes propuestas, encontramos que los autores tienden a innovar aspectos individuales de un problema de adaptación en particular. Sin embargo, estos autores no han prestado atención a encontrar métodos genéricos y sistemáticos que se puedan reutilizar para tratar diferentes tipos de problemas de adaptación multimedia. Por ello iniciamos la construcción de un motor de adaptación que incluyera estos métodos genéricos y sistemáticos de decisión.

El objetivo principal de esta tesis es la determinación del alcance de aplicación de la decisión sistemática y automática para la adaptación de contenidos multimedia. La tesis hipotetiza que los metadatatos permiten hacer estas decisiones de forma sistemática y general, es decir, independiente del contenido y del contexto multimedia. Además, la tesis hipotetiza que la descripción de las capacidades de adaptación de los módulos de adaptación permite la selección automática de los módulos de adaptación y parámetros necesarios para hacer la adaptación. Con este propósito, el trabajo de investigación descrito en esta tesis desarrolla un grupo de métodos de decisión sistemática y automática para seleccionar los posibles módulos y parámetros a ejecutar y además describe cómo las preferencias de usuario permiten seleccionar la mejor adaptación y sus parámetros. A continuación se describen los puntos de decisión y se muestra cómo, cuándo y dónde se realizan las decisiones.

Las contribuciones están distribuidas a lo largo de cuatro capítulos. El primer capítulo de contribuciones describe un motor de adaptación de contenidos multimedia llamado CAIN-21 y lo compara con otros motores de adaptación. El segundo capítulo de contribuciones describe las herramientas de descripción usadas para crear el motor de adaptación. El tercer capítulo de contribuciones se centra en el módulo de planificación que selecciona las posibles secuencias de módulos de adaptación y parámetros que se pueden ejecutar. El cuarto capítulo de contribuciones se centra en decidir cuál de las posibles secuencias de adaptación y parámetros maximiza la experiencia de usuario. Después de estos cuatro capítulos de contribuciones, el siguiente capítulo revisa y valida los cuatro capítulos de contribuciones anteriores. El último capítulo recoge las mayores contribuciones, conclusiones, implicaciones y ideas de trabajo futuro.

CONTRIBUTIONS TO MULTIMEDIA ADAPTATION WITHIN THE MPEG-21 FRAMEWORK

Abstract

The current research includes a wide variety of adaptation methods and underlying technologies. After comparing different proposals, we found that authors tend to focus on innovating individual aspects of a particular adaptation problem. However, they have not paid attention to finding generic and systematic methods that could be reused to address different types of multimedia adaptation problems. Therefore, we initiated the construction of an adaptation engine that encompasses these generic and systematic decision methods.

The main objective of this thesis is to determine the reachable scope of systematic and automatic multimedia adaptation decision-making. The thesis hypothesises that metadata allows for the addressing of these decisions systematically and generically, i.e., regardless of the multimedia content and context. In addition, the thesis hypothesises that the description of the capabilities of the adaptation modules allows for the automatic selection of the adaptation modules and parameters necessary to make the adaptation. To this end, the research work described in this thesis develops a group of systematic and automatic decision methods that selects the feasible modules and parameters to execute. It then describes the decision points, shows how, when and where the decisions are carried out and in addition describes how to use the user preferences to select the best adaptation and its parameters.

The contributions are distributed throughout four chapters. The first contributions chapter describes a multimedia adaptation engine named CAIN-21 and compares this engine with other multimedia adaptation engines in the relevant literature. The second contributions chapter describes the description tools used to create the adaptation engine. The third contributions chapter focuses on the planning module that selects the feasible sequences of adaptation modules and parameters that can be executed. The fourth contributions chapter focuses on deciding which of the feasible adaptation sequences and parameters maximizes the user's experience. After these four contribution chapters, the following chapter reviews and validates the previous four chapters' contributions. The last chapter collects the major contributions, conclusions, implications and ideas for future work.

Contents

PART I:

INTRODUCTION AND BACKGROUND

Chapter 1:

Introduction

Synopsis:

This introductory chapter provides a summary of the ideas that have inspired this thesis, and then it goes on to discuss the relevance of multimedia adaptation and to highlight its importance. The chapter also discusses the main objectives of the thesis. The description of the technical details has been kept apart and deferred to the following chapters. The end of the chapter provides an outline of the the remainder of the thesis.

1 Introduction

Multimedia allows the combination of media elements of different modalities (text, audio, images, video, etc.). This way of representing information quickly became popular during the 1980s, mainly because it is a useful instrument to help everyday users to better manage and understand information.

It is easy to understand the predilection of the users for multimedia applications when these do not present technical issues. Surprisingly, however, multimedia can hide many technical glitches inside of it. An important goal in successful *multimedia systems* design is to hide this underlying complexity from the end-users.

Multimedia content providers need to distribute their photos, videos and audio to a wide-range of devices, regardless of the underlying delivery technology. Although an important effort has been made in recent years to find out the best ways to manage this multimedia, presently there is no universal interface for multimedia representation and management. A significant cause of this incompatibility is that multimedia is designed for use in devices produced by many different manufacturers. When multimedia content is conveyed for consumption by unrelated manufacturer's devices, users often experience difficulties utilizing such content. As time goes by, the variety of multimedia formats and devices increases, thus increasing the likelihood that users will have difficulties.

Multimedia adaptation aims to carry out changes to multimedia content in order to fill in the gaps that hamper its consumption by users. In addition, multimedia adaptation offers content providers added value by increasing the range of terminals and networks that can consume their content. Furthermore, multimedia adaptation improves the utility of such content by offering the content provider and its customers the capability of customizing the content to their individual preferences. All too often, the multimedia adaptation modules are intended to adapt only a specific type of media (e.g., audio, video, images), or even a specific media format (e.g., H.264 / *Advanced Video Coding* (AVC)), to a set of constraints imposed by the *usage environment* – namely the terminal, the network and the preferences of both the content provider and its users. *Systematic* multimedia adaptation aims to find common procedures that a machine could execute to adapt dissimilar multimedia content to different usage environments.

2 Motivation

Interoperability refers to the ability of diverse systems to work together (inter-operate). Universal multimedia interoperability is still an open issue. At a first glance, there are two major (and complementary) approaches to fill in the gaps between multimedia content and systems: the *adopt* approach and the *adapt* approach. The adopt approach consists of standardising the formats, profiles and protocols. The industry has reached a certain level of success with the adopt approach (e.g., the video for iPhone must be in a specific set of MP4 video profiles). However, this one-size-fits-all approach hinders interoperability between different devices and frequently ignores users' preferences. On the other hand, the adapt approach proposes modifying multimedia content to work on the constraints of the usage environment. The problem associated with using multimedia content in different devices cannot be solved by standardization only. This is because devices with different features (e.g., screen sizes) or users with different preferences (e.g., a user who prefers content without audio) may demand different representations of the same content.

For these reasons, multimedia adaptation is an inherently important issue both for the present and future multimedia rich world.

Early multimedia systems tended to assign the users the responsibility of adapting multimedia content to their devices. In *manual* multimedia adaptation, the users are responsible for executing the technical operations necessary to prepare and transfer content for consumption into their devices. On the contrary, *automatic* multimedia adaptation aims to liberate the users from the responsibility of having to execute the right operations to adapt the content for consumption. In automatic adaptation systems, the users are responsible for deciding *what* they want and the electronic system decides *how* this content is to be adapted and transferred to the terminal.

Automatic multimedia adaptation does not just cover the *syntactic aspects* of formats and profiles adaptation. The users may wish to modify *semantic aspects* of content adaptation. Video, audio and images are sampled and compressed data. It is difficult for computing machines to interpret the underlying meaning of these data necessary to semantically adapt multimedia. The users, however, are very well aware of multimedia content's abstract view and semantics. For instance, a user can decide to increase the contrast of an image to highlight faces or to crop the important area of a photo and remove the rest. As another example, a user can remove video segments, audio tracks or subtitles in which he/she is not interested. For these reasons, multimedia adaptation has normally been a duty assigned to users. Nonetheless, these types of adaptations are tedious and time-consuming. Therefore, it is worthwhile for the user to automate these adaptation operations as much as possible. The purpose of this thesis is to relieve the users from having to make all these multimedia adaptation decisions by means of automating the decision process.

3 Objectives

The main objective of the work presented in this thesis is to study the reachable scope of systematic and automatic multimedia adaptation decision-making. This overall objective can be divided into four distinct objectives:

1. *Systematic multimedia adaptation*. Current multimedia adaptation decision methods have increased interoperability among heterogeneous multimedia systems. However, to the best of our knowledge, existing multimedia adaptation decision methods are focused on the adaptation of specific types of content. For example, deciding the best layout for the elements of a web page in a mobile device, or choosing the best layer for a scalable visual stream that is going to be delivered to an array of terminals, etc. However, at present time, there are no products addressing multimedia adaptation regardless of the multimedia content and context. One aim of this thesis is to identify generic procedures that can be use to address different kinds of adaptations.

2. *Automatic multimedia adaptation*. The adaptation mechanism must, as much as possible, free the user from making decisions on how to carry out the adaptation. To this end, the automatic adaptation mechanism has to identify the modules capable of adapting the multimedia content and selecting the best adaptation among the feasible ones. The automatic adaptation mechanism can find that the content can be adapted to the usage environment in several ways, all of which fulfil the usage environment constraints. In this case, the criteria to select the best adaptation have to be determined. The user could have been explicitly defined these criteria in his/her preferences (e.g., language preferences). In other cases, these criteria are implicit (i.e.,

the user has not explicitly defined them). In this case there are well-known methods (e.g., quality measures) to decide which feasible adaptation maximises the user's experience.

3. *Interoperability*. Designing, developing and validating a multimedia adaptation engine that further achieve the idea of full universal interoperability. Multimedia diversity hampers interoperability because custom software is required to process each new type of *content* and *context*. To this end, the multimedia adaptation engine has to provide an extensibility mechanism that allows for the progressive integration of pluggable adaptation modules. These adaptation modules facilitate the adaptation of different multimedia content and therefore enlarge the range of multimedia devices able to consume such content.

4. *Storable and repeatable multimedia adaptation validation tests*. Creating a set of adaptation tests that verifies and validates the results of the adaptation engine. These tests must be storable to check that their execution always produces the same results, i.e., the tests must be repeatable. To this end, the description of both the content and of the context has to be formalized. A description standard that harmonises multimedia technologies will be used. During the representation of these multimedia elements, handicaps in the description standards might be identified. In this case, improvements to the current description standard situation have to be proposed.

These four objectives are distributed throughout the four chapters in Part II.

4 Outline

The structure of the rest of this document is as follows:

Chapter 2 gathers the important concepts of the relevant literature on which this work will be based. Each section reviews a different group of background concepts and these are referenced in different parts of the following chapters. The purpose of this chapter is to survey the existing body of knowledge to which this thesis contributes.

Chapter 3 describes the contributions of this work. This chapter identifies some shortcomings in the current proposals to achieve a true universal adaptation engine. After that, the chapter proposes an extensible multimedia adaptation system. The chapter also justifies the selection of the description standards. After that, the chapter describes the description tools and architecture of this engine. Existing adaptation systems are re-used as much as possible, and the subsystems that do not exist are created. Finally, the chapter provides a comparative analysis between the proposed adaptation engine and other multimedia adaptation engines. This chapter contributes to multimedia adaptation by providing an extensible, systematic and automatic multimedia adaptation engine. The following chapters use this engine to evaluate the decision methods thereby proposed.

Chapter 4 starts reviewing the standard description tools used in the proposed adaptation engine. Subsequently, it identifies some limitations and ambiguities in the current description tools. It then justifies the introduction of complementary description tools addressing these limitations and ambiguities. These description tools allow for the description of storable and repeteable tests, the necesary metadata managament and the proper description of the adaptation capabilities.

Chapter 5 proposes a novel method to systematically and automatically select multimedia adaptation modules. The decision is taken according to the metadata that describes the media. This approach makes the decision method generic and independent of the underlying content to be adapted. Specifically, the method allows for the systematic and automatic identification of all the multi-step sequences of such modules that fulfil the usage environment constraints. The method also identifies the parameters that can be used in each step. In addition, the proposed decision method is compared with other multimedia decision methods.

Chapter 6 is based on the results of the previous chapter. This chapter contributes to multimedia adaptation decicion-making by providing a systematic multimedia preferences model and decision methods that identify the best adaptation. The main idea is that, after executing the decision mechanism, one will obtain different ways to execute these modules in order to adapt the multimedia content to the consumption terminal. At this point, the user preferences are added to the selection process. In this way, the chapter describes how to select the modules and parameters that best suit the user preferences. Some methods to assess the utility of the selected modules and parameters are discussed at the end of the chapter.

Chapter 7 describes the tests and experiments that have been undertaken to validate the results of the previous chapters.

Chapter 8 summarizes the main contributions of this research. It also discusses the results and their most important conclusions, which in turn may give rise to future studies.

Chapter 2:

State of the art

Synopsis:

This chapter describes the state of the art of different technologies, methods and approaches related to multimedia adaptation decision-making, which exist at the time of writing this thesis. The background ideas encompass multimedia adaptation techniques, multimedia description tools, comparisons among different multimedia adaptation engines, the idea of multi-step adaptation and the use of preferences to improve the adaptation decision process.

1 Introduction

The purpose of this chapter is to recapitulate the main approaches to multimedia adaptation in the literature. The chapter is specially centred in the areas to which this thesis contributes, and its content will be referenced in the following chapters for comparison purposes.

The description of the state of the art of multimedia adaptation in this chapter focuses on the aspect to which this thesis have contributed, rather than on the overall state of the art of multimedia adaptation. In addition, a rich set of references complement this survey with other notable multimedia adaptation approaches to which this thesis has not made any contributions.

The structure of this chapter is as follows. Section 2 describes the current types of multimedia composition, adaptation and delivery modes. Section 3 describes the evolution and current state of the multimedia description standards. Section 4 describes how semantics can improve multimedia adaptation. Section 5 goes over a group of adaptation engines that relate and can be compared to *Content Adaptation INtegrator in the MPEG-21 framework* (CAIN-21). Section 6 reviews the classical and neoclassical planning techniques as well as the idea of non-deterministic planner. Section 7 analyzes how to represent and manage the user preferences. Section 8 concludes the chapter.

2 Multimedia adaptation and delivery

Multimedia adaptation offers content providers added value by increasing the range of terminals that can consume their content. The paradigm of *terminal-centric* adaptation is intended to guarantee unrestricted delivery of multimedia content to diverse terminal capabilities and networks. Therefore, terminal-centric adaptation deals with two important issues:

1. Addressing technical incompatibilities among devices and media formats.
2. Reducing the size of the media that is delivered to constrained environments. For instance, with mobile phones, media with high resolution increase the delivery time, but due to the spatial resolution of the mobile, the media ends up having a reduced presentation resolution, so it would be optimal to reduce the resolution before transmission.

In addition to terminal-centric adaptation, content providers can improve the quality of their service by offering adaptive and customized content according to the end-users' preferences. The paradigm of *Universal Multimedia Access* (UMA) [1] aims to enable terminal-centric adaptation, but in addition, to take into account the user’s preferences. Therefore, UMA is not just intended to fulfil technical constraints but also to provide a result that maximizes the satisfaction of the end-user. The paradigm of *user-centric* adaptation goes a step further and places the end-user's assessment of the multimedia utility in the centre of the adaptation. User-centric adaptation is also referred to as *Universal Multimedia Experience* (UME) [2].

2.1 Multimedia composition levels

In computer science, *media* is a collection of information prepared for consumption by the human senses. Regular media is represented only in one *modality* (such as video, audio, image or text). *Multimedia* extends the notion of media by combining at least two media elements synchronized for presentation. This document uses the term *resource* to refers to both media or multimedia content. Multimedia may be seen as having different levels of composition (not necessarily unrelated):

1. *Resource level composition*. Isolated and homogeneous media or multimedia content. It can be consumed alone or in conjunction with other resources. Usually (but not always), it is stored in one single file. We can further divide resources at this level into: a) single *media level*, in which a standardization body defines the whole format of a single media modality (e.g., *.jpg* or *.mp3*), and b) *multimedia level*, in which the resource is made up of one or more media modalities (e.g., *.mpg* video[1] files or *.html* web files). Subsection 3.1 further describes the difference between a *media resource* and a *multimedia resource*.
2. *Structure level composition* (also named *system level* or *application layer* [3]). Different kinds of resources (such as audio and video) are gathered together forming high-level structures and concepts (e.g., MPEG-21 *Digital Items* (DIs) [4] or NewsML [5]). Usually this level also comprises metadata describing the resources (e.g., MPEG-7 Part 5 *MediaInformation* [6] or semantic information).
3. *Scene level composition*. Occurs when the multimedia elements (single or composed media elements) form a whole multimedia presentation. This level extends the structure level composition with either or both: a) *layout information*, which provides cues and restricts or completely defines the way in which the content must be rendered onto the screen of the terminal (e.g., *HyperText Markup Language* HTML), and b) *synchronization information*, which helps during the rendering of the content along the timeline as well as during the synchronization of the different elements of the scene (e.g., *Synchonized Multimedia Integration Language* (SMIL), MPEG-4 *Binary Format for Scenes* (BIFS) or Adobe Flash).

2.2 Types of multimedia adaptations

Multimedia adaptation follows the basic principles of meeting the usage environment constraints and, at the same time, preserving the utility (e.g., information, quality) of the adapted media as much as possible. The objective of this principle can be accomplished using different types of multimedia adaptations. In particular, we propose the following taxonomies.

With respect to *the operations that are performed to adapt the media*, multimedia adaptation can be classified into[2]:

1. *Transrating*. The adaptation changes the media sampling rate to a lower sampling rate while maintaining the media content, modality and format. (e.g., frame-dropping, coefficient-dropping).
2. *Transforming*. The adaptation changes the media content while maintaining the media modality and format (e.g., summarization, spatial resolution adjustments, scalable visual or audio streams adaptation).
3. *Transcoding*. The adaptation changes the media format to produce content that the usage environment can consume (e.g., *Tagged Image File Format* (TIFF) to *Portable Network Graphics* (PNG) image format conversion).

[1] In this work, the term video refers to gathering several streams (normally visual and audio streams), although another streams (such as the subtitles or transcoding hints) can also exist.

[2] Different authors propose different taxonomies for the adaptation operations. For instance, [7] divides adaptation operations into static adaptation (i.e., variations) and dynamic adaptation (i.e., transcoding, transforming, transmoding). In [8] the efficiency of scalable video adaptation is stressed by dividing adaptation operations into variations, scalable video and transcoding.

4. *Transmoding*. The adaptation changes the modality of the media or multimedia resource (e.g., transmoding from a visual stream to license plate numbers).
5. *Variation selection*. The adaptation creates a pre-defined set of versions (i.e., variations) of the media with different modalities and fidelities. Then it selects the variation that best fits with the target usage environment (e.g., image thumbnails are variations of the original image with a low delivery cost).

The simple way to transcode or transform is to fully decode and re-encode the media in a different format or with a scaled down version of the content. Recently, a great effort has been made in efficient transrating, transforming or transcoding audio or video by avoiding fully decoding the compressed media streams (see, for instance, [8][9]). Variation selection permits storing several variations of the media stream, which greatly reduces the computational cost, but has a higher storage cost. Scalable media is a promising approach, which allows for the efficient adaptation of resources without the cost-intensive encoding/decoding steps. Van Deursen et al. [10] have proposed a two-step media resource customization method that combines variation selection and scalable video. First, variation selection obtains a scalable resource roughly suited for the usage environment. Next, the authors perform scalable layer selection and semantic adaptation that further tailor the media resource. Currently, the main limitation of scalable video is that few terminals in the market support it.

In reference to transmoding, Li et al. have described the classical *InfoPyramid* multimedia adaptation model [11]. In this model, multimedia variations are represented with different modalities (e.g., video, images, text), at different levels of abstraction (e.g., raw media, features, semantic, metadata), and with different resolutions (e.g., thumbnails, key-frames, video). In this pyramid, the variations with higher levels of detail include other variations extracted from the former ones. Furthermore, the variation selection process decides which variation best fits the usage environment constraints. The classical *InfoPyramid* model does not include scalable media, but this could be easily added.

With respect to *the level of understanding applied to the media*, multimedia adaptation can be performed in two different ways:

1. *Signal level adaptation*. This kind of adaptation is committed to transrating, transforming or transcoding media resources without understanding the meaning of the content.
2. *Semantic level adaptation*. This kind of adaptation modifies the media supposing that there exists some knowledge about the meaning of the content.

With respect to the *location where the adaptation is performed*, we can classify adaptation techniques into:

1. *Centralized adaptation*. The media processing is performed in one location that can be a client, a proxy or a server. Content adaptation in the server is more common because of a server's comparatively less constrained capability for content processing.
2. *Distributed adaptation*. The adaptation process is distributed over various locations, usually referred to as *nodes*.

With respect to *the way in which multimedia is composed*, adaptation can be performed at the three different composition levels:

1. *Resource level adaptation*. Adaptation of the resource through transrating, transforming or transcoding.
2. *Structure level adaptation*. Adaptation of a resource that is composed of (or has references to) other media resources (e.g., DI, MPEG-4 BIFS, HTML, etc.).
3. *Scene level adaptation*. Adaptation of the spatial or temporal information of the media resources in the scene, as well as that of the relationships. This adaptation preserves the consistence and meaning of the elements in the scene. To this end, often the semantics of the element in the scene are annotated. It can be divided into: a) *layout level adaptation*, which changes the arrangements of the constituent elements in the scene (e.g., HTML or SMIL), and b) *synchronization level adaptation*, which modifies the timeline of the constituent resources (e.g., video summarization or SMIL images serialization).

This thesis focuses on structure level adaptation in the MPEG-21 framework. Scene level adaptation (layout and synchronization level adaptation) along with the rendering of the DIs in the scene is out of the scope of this work. For further information about scene level adaptation of DIs, the reader can consult [12].

2.3 Multimedia adaptation decision methods

Usually multimedia adaptation engines perform the adaptation in two phases, each executed in a sequential manner (see, for instance, [13][14]). Firstly, a *decision phase* is used to decide which adaptation best suits the constraints of the usage environment. Secondly, in the *execution phase*, these adaptation actions are performed on the media. For instance, in [14] the *Adaptation Manager* module is in charge of the decision phase and the *Content Adaptor* module carries out the execution phase.

For the decision phase, two different methods have been widely investigated in the literature:

1. *Utility-based methods* [13][15] aim at finding the adaptation parameters that maximise the end-user's satisfaction with the adapted media. Frequently, these methods operate by solving an optimisation problem. The MPEG-21 Part 7 DIA *AdaptationQoS* description tools (examined in the next section) have been proposed to point out these relationships between the adaptation parameters and the corresponding utilities.

2. *Knowledge-based methods* [16] have been used primarily to determine whether a conversion can be executed and which parameters must be supplied to produce adapted content. The term *adapted content* refers to multimedia content that fulfils the terminal and network constraints and hence can be consumed in the user's terminal. These methods usually analyze the concatenation of several conversions in a sequence. They have also been referred to as *multi-step adaptation*.

In general, the utility-based methods aim to maximize the *utility*, which refers to the end-user's satisfaction. The utility can be increased by adapting the media according to the user's preferences (e.g., summarizing the parts of the media in which the user is interested) or by maximizing the *quality* (fidelity of the adapted media with respect to the original media). In this latter case, authors distinguish between *subjective quality assessments* and *objective quality assessments* (see, for instance, [17]). Subjective quality assessments are concerned with how the end-user perceives the video. Subjective quality assessments are expensive in terms of time and test preparation. To reduce the inherent cost associated with subjective quality assessments, objective quality assessments are often used. Objective quality assessments are mathematical models that approximate results of subjective quality assessments. They are based on criteria and metrics that a computer program can obtain automatically. To make this *assessment*, objective quality metrics use standard *metrics* such as *Peak Signal-to-Noise Ratio* (PSNR), *Video Quality Metrics* (VQM) or *Structural SIMilarity* (SSIM). See [18], for a further description of these metrics.

In [19] we proposed combining both methods in sequence. Firstly, the knowledge-based methods use the media format to decide which conversions have to be carried out in order to adapt the content to the usage environment. Chapter 5 describes in detail this knowledge-based method. Secondly, certain “intelligent” conversion modules incorporate the capability to select the parameters that maximize the utility of their output. Chapter 7 demonstrates these utility-based methods.

2.4 Delivery and adaptation modes

From the standpoint of the media client, there are two main media *delivery modes* [20]: *download*, in which the client starts to play the media content after completely receiving the media from the server, and *streaming* in which media content is played while data reception is in progress. *Streaming servers* usually send the users two types of media streams, namely visual stream, audio stream or both:

- *Live media*. Broadcast of live events in real time. This streaming is useful when the client expects to receive the media stream as soon as it is available. Live events, video conferencing, and surveillance systems are commonly streamed over the Internet as they happen with the assistance of streaming software. The media recording software encodes a live source in real time and transfers the resulting media to the streaming server. The streaming server then serves, or "reflects", the live stream to clients. Regardless of when different customers connect to the stream, each sees the same point in the stream at the same time.
- *Media On Demand* (MOD). This category includes *Video On Demand* (VOD) and *Audio On Demand* (AOD) [20]. Each customer initiates the reception of the media from the beginning, so no customer ever comes in "late" to the stream. For instance, TV channels can use this mode to distribute movies to users who play those movies at different times.

With respect to the moment at which the adaptation takes place, media adaptation can be divided into three *adaptation modes*:

- *Offline Adaptation mode* (*OffA mode*). The adaptation is performed in the background and before the media is available to the user. This mode is adequate for on demand media delivery. However, this mode is not suitable for live media because the user is expecting to watch the event as soon as it occurs. This adaptation requires a predefined set of usage environments (i.e., terminals, networks and user's preferences). Variations of the media can be prepared for each of these predefined usage environments. For this purpose, MPEG-7 Part 5 [6] has proposed to create a group of variations (described with instances of the *mpeg7:VariationDescriptionType* description tool). The main limitation of the *OffA* mode is that the user’s preferences and natural environment constraints cannot be completely taken into account. In particular, creating a repository of adapted resources for each usage environment is possible, but it becomes unmanageable from a practical point of view when the number of feasible usage environments notably increases. Therefore, at most, the *OffA* mode can address a limited and predefined set of user preferences. Note also that for *OffA* mode with MOD, the encoder complexity is less of an issue. However, if the media stream is encoded from a live source, then the encoding complexity will become a significant constraint.

- *On Demand Adaptation mode* (*OdA mode*). Adaptation takes place at the same time the user requests the resource. In this mode, the client's characteristics, preferences and natural environment can be fully taken into account. However, if the resource adaptation process is time consuming, the user has to wait until the whole resource is adapted. Therefore, this adaptation turns out to be useful for small resources (e.g., images), but may be impractical for long resources (e.g., video or speech).
- *Online Adaptation Mode* (*OnA mode*). The adaptation begins as soon as the user asks for the resource. However, in contrast to the *OdA* mode, in the *OnA* mode the resource begins to be delivered to the user before the whole resource has been adapted. As with the *OdA* mode, the user's characteristics, preferences and natural environment can be taken into account. This adaptation is appropriate for long resources (and perhaps also for small resources). The drawback of this approach is that, in general, implementing this solution efficiently is difficult. In *OnA* mode, we need to ensure that media data fragments are delivered to the client in time to maintain playback continuity. The advantage is that, once implemented, the *OnA* mode can be reused to simulate the *OffA* and *OdA* modes.

3 Multimedia description standards

3.1 Background and evolution

In the last two decades, at least two different communities have worked on multimedia description standards: the coding community and the metadata community. The main objective of the first community is to represent multimedia content compactly and efficiently with standard multimedia formats. One of the important aims of standardization is to reduce the manufacturing costs of terminals capable of consuming multimedia. For instance, the *Joint Photographic Experts Group* (JPEG) is well known for image compression standards. Examples of audio compression standards are *MPEG-1 Audio Layer 2* (MP2), *MPEG-1 Audio Layer 3* (MP3) and *Advanced Audio Coding* (AAC). The *Moving Picture Experts Group* (MPEG) has defined widely used video compression standards such as MPEG-1, MPEG-2 or MPEG-4 [21]. The MPEG standards divide video formats into profiles and levels. A *profile* defines how complex the encoding is. Technically, a profile defines a subset of the syntax of the specification. For each profile, there are a series of levels. A *level* defines a set of constraints on the values that may be taken by the parameters of the specification within a profile. For instance, *Digital Versatile Disk* (DVD) uses the MPEG-2 *MainProfile@MainLevel* visual format. Frequently, uncompressed video is referred to as RAW video and the *Waveform Audio Format* (WAV) has been widely used to represent uncompressed audio.

Video *containers* such as *Audio Video Interleaved* (AVI), *MPEG-4 File Format* (MP4), or *Windows Media Format* (WMF) usually consist of one *visual stream* and one *audio stream*. For instance, the iPhone and the Nokia N810 mobile phones use the MP4 video container with an H.264/AVC visual stream and an AAC audio stream. Additional audio streams are frequently used to provide support for different languages (such as in DVDs). In this document, the term *media resource* is used to refer to both media (e.g., MP3 audio) and multimedia resources (e.g., MP4 video container with MP3 audio stream and AVC visual stream). In the same way, the term *multimedia resource* is used to stress that the resource groups several media resources.

Further work initiated by the coding community includes semantic multimedia analysis (such as voice or face recognition) to extract information not explicitly represented in the media. Roughly speaking, the low-level multimedia features are automatically obtained by means of signal analysis techniques. Subsequently, inference techniques are used to obtain high-level descriptions. Currently, multimedia researchers are trying to fill the semantic gap between the low-level and high-level multimedia descriptions (see for instance [22]).

As not all the multimedia semantics can be automatically extracted, the metadata community has proposed the semi-automatic or manual annotation[3] of the semantics of the content. In addition, storing these semantics is more efficient that obtaining them on-demand. The MPEG-7 [23] standard provides descriptions for the multimedia content, which can be obtained automatically or manually. This standard has chosen the *eXtensible Markup Language* (XML) [24] as the base technology to represent all this meta-information.

In the MPEG-1/2/4 coding standards the way in which the data is stored and decoded is clearly defined, but the way in which the coder obtains this data is open so that different coders can compete encoding this data more or less precisely or efficiently. MPEG-7 also follows this principle: it defines how the metadata is stored and accessed, but it does not define how to obtain these values. MPEG-7 formalizes a wide set of *description tools* in order to represent metadata of the multimedia content. For this reason MPEG-7 is formally named *Multimedia Content Description Interface*. The base technology that represents the description tools is XML Schema [25] (i.e., the description tools correspond to XML Schema description types). MPEG-7 uses the terms *Descriptors*[4] (Ds) and *Description Schemes* (DSs) to refer to these description tools. The term *Description* refers to an instance of one or more description tools (i.e., XML elements).

In summary, MPEG-1/2/4 aims to encode video (i.e., visual and audio streams) and MPEG-7 aims to describe multimedia. The next standard in this series is MPEG-21, which aims to describe all the elements of a multimedia system consistently with the idea of UMA. The MPEG-21 *framework* describes how the different elements of this multimedia system fit together during the life cycle of multimedia [26]. To harmonize multimedia systems, MPEG-21 has proposed a set of normative description tools (to which the MPEG-21 standard usually refers to as *tools*). These tools do not define how to implement a multimedia system, but rather identify the information necessary to manage multimedia.

3.2 MPEG-21 Part 2: Digital Item Declaration

The second part of MPEG-21 (i.e., MPEG-21 Part 2) focuses on standardizing multimedia content. This subsection reviews the element of this standard that this thesis utilizes.

MPEG-21 defines two important concepts, the *User* and the *Digital Item* (DI). A *User* is any entity that interacts within the MPEG-21 environment: content creator, content provider, content consumer, etc. A DI is a representation of any *asset* (multimedia intellectual or artistic creation) along with its metadata (e.g., MPEG-7 media description, intellectual property license, etc). The

[3] In this document, we use the term *annotation* to refer to information that the content creator manually adds for the end-user's consumption (e.g., movie title), and the term *metadata* to refer to both information about the media that the content creator manually adds or information that the machine automatically extracts.

[4] MPEG-7 and MPEG-21 capitalise and italicise description elements. This document follows this rule.

DI is the fundamental unit of transaction and distribution within the MPEG-21 framework, and DIs are used throughout the consumption and delivery chain.

3.2.1 Model and representation

MPEG-21 Part 2 focuses on formalizing:

- An *abstract model*, which has to be generic in order to cover all kinds of multimedia content, including metadata and relationships among the multimedia elements.
- A *representation*, which is an interoperable format that represents the elements of the abstract model.

To survey the abstract model, the left side of Fig. 1 shows an example of a music container. This example is an MPEG-21 compliant representation similar in purpose to the one used in music application such as iTunes or Songbird. The right side of Fig. 1 shows their corresponding elements in the MPEG-21 Part 2 abstract model.

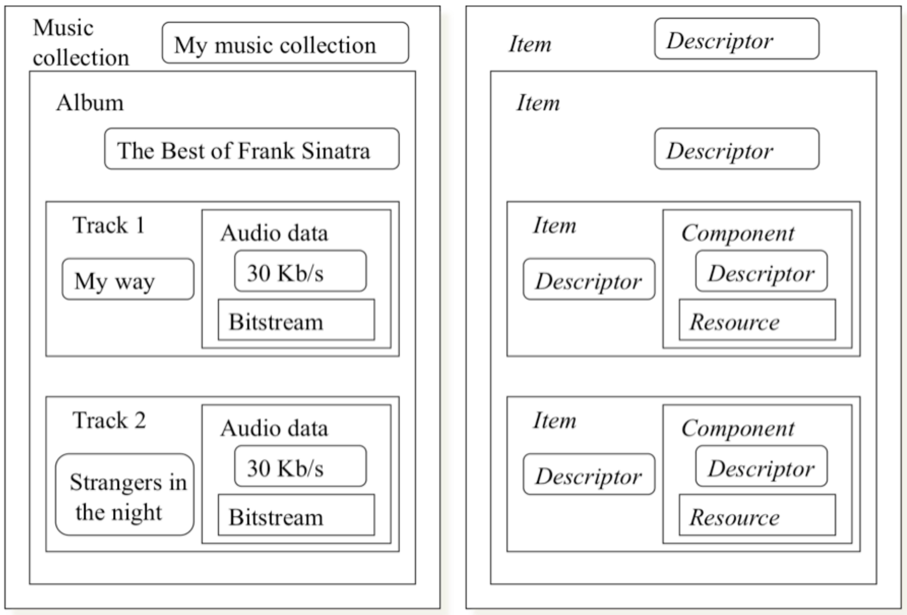


Fig. 1: Example of DI abstract model

An *Item* allows for grouping and organizing sub-*Item* and *Component* elements. The difference between a DI and an *Item* is that the DI corresponds to the wider notion of multimedia asset with standard representation whereas the *Item* just corresponds to the syntactic grouping of sub-*Item* and *Component* elements.

MPEG-21 uses the *Extended BNF* notation (EBNF) [27] to represent the relationships between the elements of the abstract model. In particular, for the *Item* element, the EBNF notation is:

Item := Condition Descriptor* Choice* (Item* | Component*)*

This means that the *Item* element includes zero or more *Condition* elements (described in Subsection 3.2.2 below), followed by zero or more *Descriptor* elements, followed by zero or more *Choice* elements and then followed by zero or more *Item* or *Component* elements.

In our example, the *Item* has a *Descriptor* element labelled "My Music Collection" and a sub-*Item* labelled "Album", which in turn has a *Descriptor* element labelled "The Best of Frank Sinatra".

A *Component* is an association between just one *Resource* element and a set of *Descriptors*. Its EBNF notation is:

Component ::= *Condition* Descriptor* Resource*

A *Resource* is an individual asset, which can be binary (e.g., audio or image file) or textual (e.g., lyrics). The *Resource* element has a *Uniform Resource Identifier* (URI) that identifies the media or multimedia resource. MPEG-21 has proposed the term *resource* to refer to any individual asset, which may be media or multimedia resource. This thesis is consistent with this proposal, but makes a distinction between *media resource* and *multimedia resource*, when necessary. If we intentionally want to avoid specifying the composition level or the existence of one or more media resources in the multimedia asset, we use the term *content* (see Subsection 2.1 of this chapter).

In reference to the representation of the elements of the abstract model, the standard uses the term *Digital Item Description* (DID) to refer to an XML document describing the DI. This document includes one or more *Item* elements. The grammar for the valid elements in the DID document is named *Digital Item Description Language* (DIDL) and defined using XML Schema.

3.2.2 Conditional elements

DIs are configurable through the so-called *Choice/Selection* mechanism that this subsection describes. This mechanism will be used to explain the DIA *Configuration* tool in Subsection 3.3.2 below.

In MPEG-21, there are *optional*, *alternative* and *conditional* elements. An optional element is an element that may or may not appear in the DI. Elements followed by * in the EBNF notation correspond to optional elements. Alternative elements are a set of elements in which only one of them has to appear. In the EBNF notation, alternative elements appear separated by |. A conditional element is an element that appears only if certain conditions are met.

MPEG-21 Part 2 allows the existence of runtime configurable DIs. In this case, the elements of the DI depend on the value of some *Predicates* elements. Each *Predicate* can take only three values: *true*, *false* or *undecided*. The *Condition* element was introduced in the previous section. A *Condition* is a conjunction (AND operator) of one or more *Predicate* elements. Its EBNF notation is:

Condition ::= *Predicate+*

A *Choice* element describes the set of related selections that can affect the configuration of the DI. Then, the *Choice* element is the "menu" that assigns values to the *Predicate* elements. The way to obtain the predicate values of the *Choice* is not defined by the standard. The *Predicate* may proceed from the end-user, content creator, content provider, hardware or software. It is even possible for the value of the *Predicate* to have not been defined at runtime. In this case, the value of the *Predicate* is *undefined*.

To clarify this idea, Fig. 2 shows an example of a DI that can be configured at run-time. This DI has three *Predicates*: LYRICS, MP3_FORMAT and WMA_FORMAT. Arrows indicate the container to which the *Condition* of the *Predicate* is associated (i.e., the Track 1 and Track 2 *Component* elements).

Once the *Condition* and *Component* elements have been defined, we need to create a *Choice*, which usually appears in the *Item* (i.e., Album in our example). Its EBNF notation is:

Choice ::= *Descriptor* Selection+*

A *Choice* must have at least one *Selection* element, each of which corresponds to a menu option (i.e., LYRICS, MP3_FORMAT and WMA_FORMAT in our example). The optional *Descriptor* in the *Choice* indicates the menu name (e.g., "Please choose the media format that your prefer").

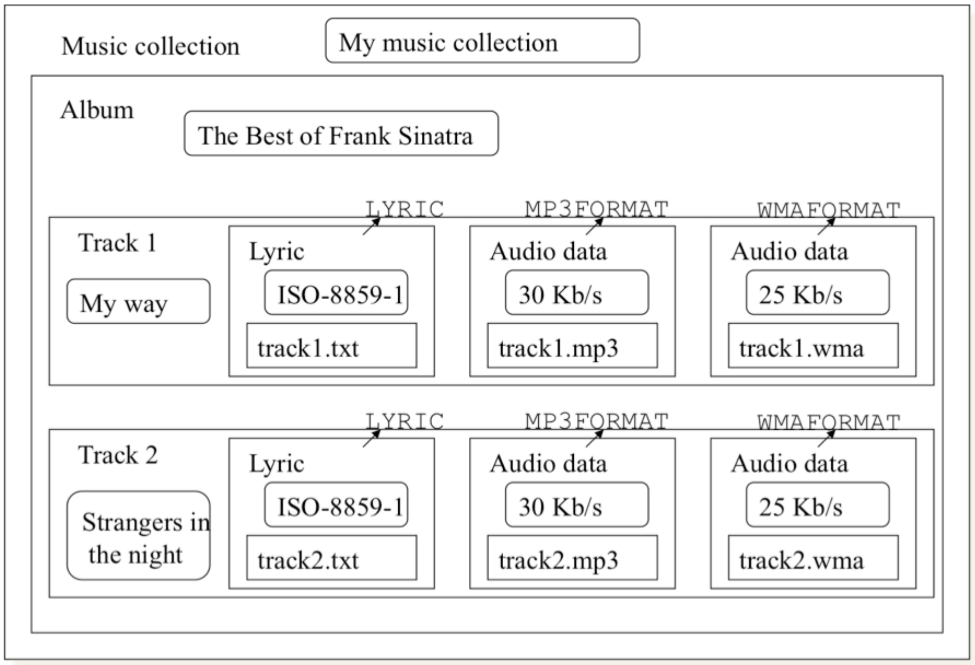


Fig. 2: Example of configurable DI

Listing 1 shows the representation of this configurable DI. The *Choice* element appears at the beginning of the album and starts with a *Descriptor* element that label the menu. After that, there are thee *Selection* elements that also use *Descriptor* elements to label the options of the menu. In particular, the *select_id* attributes provide the feasible values for the predicates. Before the *Choice* has been configured all the predicates are *undefined*. The *min_selections* and *max_selections* elements of the *Choice* limit the minimum and maximum number of predicates that can be selected (i.e., assigned to *true*). By default, these attributes take the value 0 and *unbounded*, respectively. In our example, both attributes have the value 1 indicating that only one predicate can be selected. The remaining elements can be both *undefined* or receive the *false* value.

The subsequent elements of the DI use the value of the predicates to dynamically decide which subelements are parts of the configurable DI. The *Condition* elements can precede most of the elements of the DI. In our example, the *Condition* elements precede the *Component* elements to indicate which *Component* is part of the Configurable DI. In particular, the *Condition* elements may have two attributes:

- *require* activates the condition when the predicate is *true*.
- *except* activates the condition when the predicate is *false*.

The *Component* whose *Condition* element is activated is the *Component* that will be part of the Configurable DI. Therefore, *Condition* elements in which the predicate is undefined never will be activated.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS">
<Container>
 <Descriptor>
  <Statement mimeType="text/plain">
   My music collection
  </Statement>
 </Descriptor>
 <Item>
  <Descriptor>
   <Statement mimeType="text/plain">
    The best of Frank sinatra
   </Statement>
  </Descriptor>
  <Choice minSelections="1" maxSelections="1">
   <Descriptor>
    <Statement mimeType="text/plain">
     What format do you want?
    </Statement>
   </Descriptor>
   <Selection select_id="LYRIC">
    <Descriptor>
     <Statement mimeType="text/plain">
      Lyrics
     </Statement>
   </Descriptor>
   </Selection>
   <Selection select_id="MP3FORMAT">
    <Descriptor>
     <Statement mimeType="text/plain">
      MP3 format
     </Statement>
    </Descriptor>
   </Selection>
   <Selection select_id="WMAFORMAT">
    <Descriptor>
     <Statement mimeType="text/plain">
      WMA format
     </Statement>
    </Descriptor>
   </Selection>
  </Choice>
  <Item>
   <Descriptor>
    <Statement mimeType="text/plain">
     My way
    </Statement>
   </Descriptor>
    <Component>
     <Condition require="LYRIC"/>
     <Resource mimeType="text/plain"
      ref="http://www.youtube.com/sound/sinatra/track1.txt"/>
    </Component>
   <Component>
    <Condition require="MP3FORMAT"/>
    <Resource mimeType="audio/mpeg"
```

```
     ref="http://www.youtube.com/sound/sinatra/track1.mp3"/>
   </Component>
   <Component>
    <Condition require="WMAFORMAT"/>
    <Resource mimeType="audio/wav"
     ref="http://www.youtube.com/sound/sinatra/track1.wav"/>
   </Component>
  </Item>
 <!-- More songs -->
 .....
 </Item>
</Container>
</DIDL>
```

Listing 1: Representation of a configurable DI

3.3 MPEG-21 Part 7: Digital Item Adaptation

MPEG-21 Part 7 uses the term *Digital Item Adaptation* (DIA) to refer to a set of XML documents describing the DI adaptation process. Fig. 3 shows the DIA abstract model. While DIA standardizes the white elements, it does not standardize grey boxes, but rather transfers its implementation to the *Digital Item Adaptation Engine* (DIAE). In this model, the DIAE receives a DI and produces an adapted DI ready for consumption.

MPEG-21 Part 7 distinguishes between the *Description Adaptation Engine* (DAE) and the *Resource Adaptation Engine* (RAE), see, for instance, [28]. The DAE is responsible for adapting the *Descriptor* elements within DIs, while the RAE performs adaptation on the corresponding *Resource* elements.

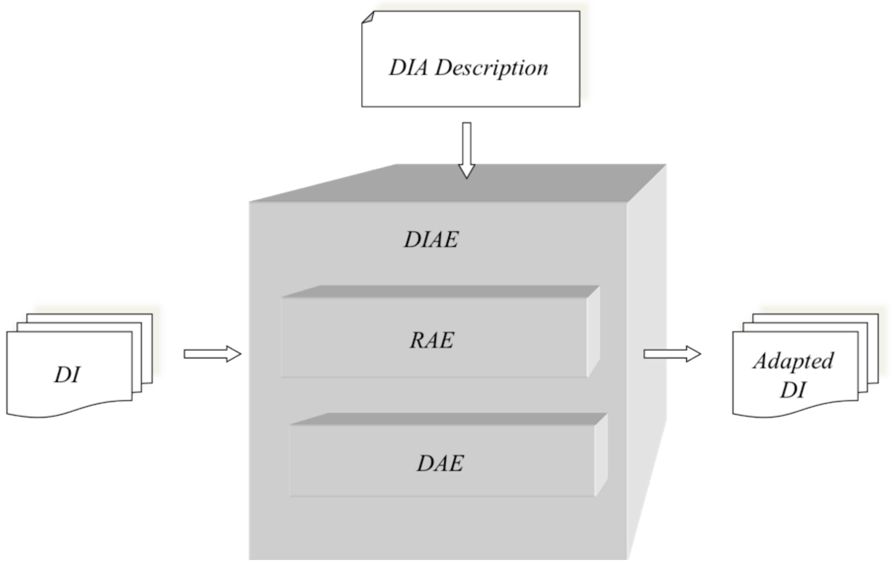


Fig. 3: Architecture of the DIAE

To make this adaptation, the DIAE utilizes a set of description tools named *DIA tools* (represented using XML Schema). A *DIA description* is an instance of a DIA tool (represented using XML). Frequently the DIA descriptions are collected in a DI, but the standard also allows for keeping them separated. In the figure, the DIA descriptions are used to drive the adaptation. These DIA descriptions may represent the usage environment, the content provider constraints, the quality of the adaptation, etc. The rest of this subsection focuses on describing the DIA tools to which this thesis has contributed.

3.3.1 Usage environment

This subsection describes the *Usage Environment Description* (UED) tools. These description tools address the UMA problem of describing the heterogeneous existing devices. In this tool, the term *terminal* refers to the physical or logical device (e.g., iPhone, Web browser) in which the user consumes multimedia content. The UED tools enable the description of different terminals, data networks and user characteristics. Listing 22 of Appendix A presents UED descriptions for the terminals, network and user preferences that we utilize, and Listing 21 of Appendix A presents the corresponding XML Schema. This thesis contributes to the user preferences description tools, which the next paragraphs further describe.

```
<DIA>
 <Description xsi:type="UsageEnvironmentType">
  <UsageEnvironmentProperty xsi:type="UsersType">
   <User>
    <UserCharacteristic xsi:type="UsagePreferencesType">
     <UsagePreferences>
      <mpeg7:FilteringAndSearchPreferences>
       <mpeg7:ClassificationPreferences>
        <mpeg7:Genre>
         <mpeg7:Name>Sports</mpeg7:Name>
        </mpeg7:Genre>
        <mpeg7:Genre>
         <mpeg7:Name>Entertainment</mpeg7:Name>
        </mpeg7:Genre>
        ................
       </mpeg7:ClassificationPreferences>
      </mpeg7:FilteringAndSearchPreferences>
     </UsagePreferences>
    </UserCharacteristic>
   </User>
  </UsageEnvironmentProperty>
 </Description>
</DIA>
```

Listing 2: *UsagePreferencesType* example

MPEG-21 UED proposes using the *UsagePreferencesType* and the *ConversionPreferenceType* description tools to describe the user's preferences. Listing 2 shows a usage example for the *User-PreferencesType* description tool. This description tool reuses the MPEG-7 Part 5 DSs. In this example, it refers to the MPEG-7 Part 5 *UserPreferences DS* and *FilteringAndSearchPreferences DS*. These DSs have been frequently used to describe the user's preferences relating the consumption of multimedia content [29][30].

The *ConversionPreferenceType* description tool describes the user's preference for conversion (i.e., media formats and modalities). In particular, the *order* attribute is a non-negative integer number representing the qualitative preference of the user for that conversion. Conversions with a smaller *order* are preferable to conversions with a higher *order*, except when the *order* is 0, in which case the conversion is not allowed. The *weight* attribute is a non-negative real integer number representing the quantitative preference of the user for that conversion. The *order* attribute is required while the *weight* attribute is optional, and its default value is 1.0. Listing 3 shows a usage example of the *ConversionPreferenceType* description tool.

```
<DIA>
 <Description xsi:type="UsageEnvironmentType">
  <UsageEnvironmentProperty xsi:type="UsersType">
   <User>
    <UserCharacteristic xsi:type="ConversionPreferenceType">
     <GeneralResourceConversions>
```

```
      <Conversion order="1" weight="1.0">
       <From href="urn:mpeg:mpeg7:cs:ContentCS:2001:4.2">
        <mpeg7:Name>Video</mpeg7:Name>
       </From>
       <To href="urn:mpeg:mpeg7:cs:ContentCS:2001:4.2">
        <mpeg7:Name>Video</mpeg7:Name>
       </To>
      </Conversion>
      <Conversion order="3" weight="1.0">
       <From href="urn:mpeg:mpeg7:cs:ContentCS:2001:4.2">
        <mpeg7:Name>Video</mpeg7:Name>
       </From>
       <To href="urn:mpeg:mpeg7:cs:ContentCS:2001:4.1">
        <mpeg7:Name>Image</mpeg7:Name>
       </To>
      </Conversion>
      <Conversion order="2" weight="1.0">
       <From href="urn:mpeg:mpeg7:cs:ContentCS:2001:4.2">
        <mpeg7:Name>Video</mpeg7:Name> </From>
       <To href="urn:mpeg:mpeg7:cs:ContentCS:2001:1"> <mpeg7:Name>Audio</mpeg7:Name>
       </To>
      </Conversion>
     </GeneralResourceConversions>
    </UserCharacteristic>
   </User>
  </UsageEnvironmentProperty>
 </Description>
</DIA>
```

Listing 3: *ConversionPreferenceType* example

3.3.2 Configuration of the adaptation

The *DIA Configuration* (DIAC) tools allow for configuring the adaptation while taking into account the DI author's intentions. This information is conveyed into a *Descriptor* element of the DI. The following subsections describe the two DIAC methods available to make this configuration (i.e., resource selection and criteria suggestion).

Resource selection

The first method is the *UserSelection/BackgroundConfiguration* elements of the DIAC. Listing 4 shows a configurable DI that uses the *Choice/Selection* elements (see Subsection 3.2.2 above) to provide different modalities of the DI.

```
<?xml version="1.0" ?>
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
      xmlns:diac="urn:mpeg:mpeg21:2003:01-DIA-DIAC-NS">
 <Item>
  <Choice choice_id="modality" minSelections="1" maxSelections="1">
   <Descriptor>
    <Statement mimeType="text/xml">
     <diac:UserSelection/>
    </Statement>
   </Descriptor>
   <Selection select_id="Audiovisual">
    <Descriptor>
      <Statement mimeType="text/plain">
       Audio and video streams
      </Statement>
    </Descriptor>
   </Selection>
   <Selection select_id="Visual">
    <Descriptor>
     <Statement mimeType="text/plain">
      Only visual stream
     </Statement>
    </Descriptor>
   </Selection>
  </Choice>
  <Component>
   <Condition require=" Audiovisual "/>
    <Resource mimeType="video/mpeg"
     ref="rtsp://www.server.com/audiovisual_movie.mpg"/>
  </Component>
  <Component>
   <Condition require=" Visual "/>
   <Resource mimeType="video/mpeg"
    ref="rtsp://www.server.com/visual_movie.mpg"/>
  </Component>
 </Item>
</DIDL>
```

Listing 4: Configurable DI

Fig. 4 shows the typical resource selection steps between a multimedia client and a multimedia server. MPEG-21 does not standardize how to transfer the DI, UED or DIAC. A typical option is to transfer these elements over *HyperText Transfer Protocol* (HTTP) [28][31]. The selection steps are as follows. 1) The multimedia player requests a DI from the multimedia server. 2) The server returns a DI along with a DIAC to indicate that this DI can be dynamically configured. 3) The multimedia player or the end-user make a decision and request the proper resource, which is returned in step 4).

During the second step, there are only two feasible and alternative elements: *UserSelection* or *BackgroundConfiguration*. The first element indicates that the end-user has to specify which resource to request. The second element indicates that the media player has to make this decision. In our example in Listing 4, the *UserSelection* element is inside the *Choice* element meaning that the end-user has to decide which resource he/she prefers.

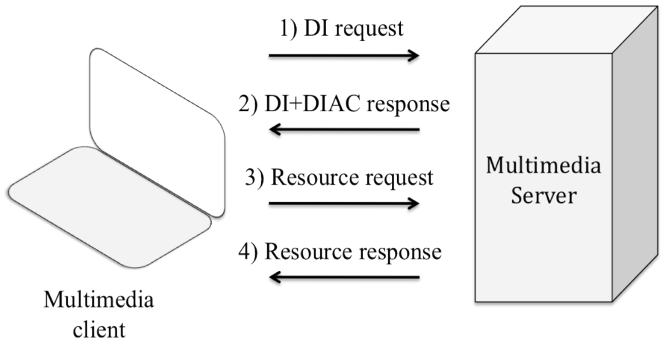


Fig. 4: Usage example of the DIA Configuration

Criteria suggestion

In this case, the *SuggestedDIADescriptionType* description tool is provided inside a *Descriptor* element of the configurable DI (usually stored in the server). The information provided is a suggestion, that is, it is not mandatory to take into account this element of the DI.

For instance, the DI's author can advise the adaptation engine to use the *Bitrate* of an instance of the *VideoCapabilitiesType* description tool of the UED. In this example, an instance of the *SuggestedDIADescriptionType* description tool would contain an XPath [32] pointer to the *Bitrate* element.

The *SuggestedDIADescriptionType* tool also allows the DI author to suggest where the adaptation may be performed: at the receiver side, at the server side or at either sides.

3.3.3 Coding format-independent adaptation

Subsections 3.3.1 and 3.3.2 introduced description tools that address *device independence*, i.e., the adaptation is independent of the terminal, network and user's preferences. This subsection introduces the *Bitstream Syntax Description* (BSD) tools, which produce *coding format-independence*. The paradigm of coding format-independence enables the adaptation engine to adapt the media regardless of the underlying coding format. An adaptation module is coding format-independent if it is deployed once and used for multiple coding formats.

Traditional video and audio transcoders require partially or completely decoding and re-encoding the compressed bitstream. In contrast, scalable content transcoders (typically visual and audio stream transcoders) perform simple bit truncations and modifications in the bitstream. The BSD tools can represent these truncations and modifications, so they are especially useful for transcoding scalable content.

To achieve coding format-independence, the BSD tools allow the description of the high-level syntactic structure of the bitstream (i.e., how the stream is organized in term of frames, layers or packets) using XML. Once this structure is represented in XML, one can define a standard *eXtensible Stylesheet Language Transformation* (XSLT) [33] to transform the *original BSD* into an *adapted BSD*. As the adaptation is executed in the XML domain, the adaptation process can be generalized thus removing dependence on the specific bitstream format. Coding format-independence is achieved because the adaptation module transforms the high level description of the original BSD (following the abovementioned XSLT) into the adapted BSD. Subsequently, a generic processor will use the adapted BSD to generate the adapted bitstream.

Some authors [34] have studied extending the coding format-independence idea to achieve delivery format-independence (i.e., engines independent of the underlying delivery format). MPEG-21 Part 18 has standardized the *Bitstream Binding Language* (BBL) tools that map the media resource and metadata to the delivery format [35][36].

3.3.4 Quality of the adaptation

Often, adaptation engines do not systematically analyze the utility (introduced in Subsection 2.3) that each set of parameters yields; instead, the selection of the adaptation parameters is made in an ad-hoc manner. Consequently, these adaptation parameters do not produce the maximum utility.

The MPEG-21 Part 7 *Adaptation Quality of Service* (*AdaptationQoS*) description tools aim to improve the parameter selection by analyzing the utility of different adaptation parameters. In this way, the adaptation engine can select the adaptation parameter configuration that, meeting the usage environment constraints, maximises the utility of the adapted media resource.

Computing the *AdaptationQoS* of the media resource is computationally intensive; however, once calculated, this information can be stored in an *AdaptationQoS* description (which in turn can be stored in a *Descriptor* of the DI). In this way, the adaptation engine can efficiently initiate the adaptation with an optimal parameter configuration [13]. However, pre-calculating the *AdaptationQoS* is only useful for MOD applications (see Subsection 2.4). In the case of live video, Wang et al. [37] have proposed that, instead of pre-calculating the adaptation utilities, their utility prediction function allows for making this real-time adaptation decisions efficiently. In this case, the values of the *AdaptationQoS* are not pre-stored, but rather calculated on-demand.

An *AdaptationQoS* description has three parts:

- *IOPins*, which act as variables (or constants).
- *Modules*, which relate the input values of one or more *IOPins* to the output values also defined through *IOPins*. These output values can be pre-calculated in the *Module*, calculated by means of a stack function that the *Module* provides, or the *Module* can even indicate that an algorithm has to be executed to compute these values.
- *Constraints*, which are optional and use the MPEG-21 Part 7 *Universal Constraints Description* (UCD) tools. The UCD tools allow for further restriction of the usage environment constraints by taking into account the DI author and DI provider constraints.

In this research, we do not use the *Constraints* description element or the UCD tools. Therefore, we are going to further describe only the *IOPins* and the *Modules* descriptions.

The IOPins

The *IOPins* are variables and constants that stand for the inputs and outputs of the *Modules*. Each *IOPin* has a unique ID (*xsd:ID* attribute). In addition, the *IOPin* can have a semantic label that represents the semantic meaning of its value.

The *IOPin* domain can be continuous or discrete. In the former case, the domain is represented by a minimum and a maximum value. In the latter case, the sampling values are provided in the *IOPin* description.

The Modules

Modules provide a mechanism to select and output value given one or more imput values. Mathematically, a *Module* corresponds to the idea of *function* in which the relationship between the arguments (input) and the evaluation of the function (output) is described by means of *IOPins*. From the point of view of computer science, a *Module* corresponds to the idea of *software operation* in which the relationship between the parameters (input) and the return value (output) is represented by means of *IOPins*. In both cases, it is important to remember that a *Module* is not software, but a description of the relationships among input and output values. The adaptation engine uses these values to make an adaptation decision. The standard defines three types of *Modules*:

- *UtilityFunctionType*, which provides information about a limited set of adaptations and their utility.

- *LookUpTableType*, which represents a matrix in which each discrete value of the input *IOPins* is mapped to an output *IOPin* value. If the input values are continuous, they can be interpolated.
- *StackFunctionType*, which enables the representation of the adaptation quality as a function of the input *IOPins*.

This research utilizes the *UtilityFunctionType* description tool to describe the relationships among constraints, adaptation operations and utilities. Typical examples of constraints are the bandwidth or the spatial resolution. Among the typical operations, we find the frame dropping or coefficient dropping transrating operations. The distortion index and the PSNR are typical examples of the utility metrics. To describe these relationships we have to create *Vector* elements with a list of values in which the *i*-th value represents:

- For constraints, the constraint value (e.g., BANDWIDTH<12000).
- For adaptation operations, the adaptation to perform (e.g., an adaptation operation labelled CHANNELS can take three values S(Stereo), M(Mono) or N(None) indicating that both channels have to be preserved, that only one channel have to remain or that all the channels have to be removed).
- For the utility, the assessment of the utility for the adaptation.

3.3.5 Conversion description tools

The MPEG-21 Part 7 Amendment 1 specifies the conversion description tools that are intended to convey *steering descriptions*. The steering descriptions provide information related to the adaptations that can be performed and instructions on how to conduct these conversions. The conversion tools include the *BSDLink* tools and *ConversionLink* tools [38]. The former enable the linking of steering descriptions to the BSD tools (i.e., scalable media bitstreams). The latter enables the linking of steering description to general-purpose conversions (i.e., transrating, transforming transcoding, transmoding and variations). Some contributions in this thesis both use and are compared to the *ConversionLink* tools, which are further described below.

In a multimedia framework, in which different terminals are capable of making different adaptations, it is important to describe the adaptation capabilities of these terminals[5]. For instance, a streaming server may describe its adaptation services for different end-users' terminals by means of a group of adaptations (grey scaler, spatial scaler, cropper, etc).

MPEG-21 Part 7 recommends using the term *conversion* to refer to a process that changes the characteristics of a resource. A *conversion act* refers to a conversion and its parameters, including the actual name of the parameters. Finally, a *conversion tool* refers to a hardware and/or software module that implements a conversion act in order to perform the conversion.

Describing the conversion capabilities

The standard proposes three different tools to describe the conversion capabilities:

1. *ConversionCapabilitiesType*, whose instances provide the conversion capabilities of a terminal.

[5] While this work normally uses the term *terminal* to refer to the end-user's terminal, the MPEG-21 standard frequently uses the term *terminal* to refer to any multimedia processor. In the case of the *ConversionLink* tools, the term *terminal* refers to any multimedia processor.

2. *ConversionLinkType*, whose instances describe an individual conversion act (i.e., conversion tool and parameters).
3. *ConversionCompositeType*, whose instances describe a multi-step conversion, i.e., composed of two or more instances of the *ConversionLinkType*.

This thesis contributes to the *ConversionCapabilitiesType* tool. This tool can contain several *ConversionCapabilityType* tools. The label *ConversionCapability* represents an instance of the *ConversionCapabilityType*. This description tool uses the following XML schema to enable any proprietary description of the conversion capability:

```
<complexType name="ConversionCapabilityType">
 <complexContent>
  <extension base="dia:ConversionDescriptionBaseType">
   <sequence>
    <any namespace="##other" processContents="lax" minOccurs="0"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

Listing 5 provides a usage example that shows how to associate conversion capabilities to a terminal. The particular elements that describe adaptation capabilities are not standardized. MPEG-21 Part 6 standardizes a dictionary of terms (mainly devoted to rights and permissions management) for use within the MPEG- 21 Framework [39]. In this example, the *ConversionActUri* element provides the semantic using a URI that references a specialized MPEG-21 Part 6 term for the rectangular cropping operation of bitmap image.

```
<?xml version="1.0" ?>
<DIA xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <Description xsi:type="TerminalsType">
  <Terminal>
   <TerminalCapability xsi:type="ConversionCapabilitiesType">
    <ConversionCapability>
     <ConversionActUri uri="urn:mpeg:mpeg21:2003:01-RDD-NS:CropRectangularBitmapImage">
      <!-- Propietary description -->
     </ConversionActUri>
    </ConversionCapability>
   </TerminalCapability>
  </Terminal>
 </Description>
</DIA>
```

Listing 5: Usage example of the *ConversionCapabilitiesType* tool

3.4 Related usage environment description standards

In the multimedia adaptation literature, there are two mainstream usage environment description standards: *User Agent Profile* (UAProf) from the Open Mobile Alliance [40], previously named *Wireless Access Protocol* (WAP) Forum, and the MPEG-21 Part 7 UED tools described in Subsection 3.3.1 of this chapter.

UAProf is based on the *Composite Capabilities/Preference Profiles* (CC/PP) representation standard. The CC/PP standard was initiated by the W3C to standardise descriptions based on the *Resource Description Framework* (RDF) [41]. RDF is a general-purpose standard that can represent directed graphs constituted of triples (subject, predicate, object). CC/PP restrict the representation to a two level hierarchy made up of *components* and *attributes*. CC/PP is independent of any particular vocabularies and therefore does not define which components and attributes must be used in a profile. UAProf is a specific implementation aimed at mobile devices (it comprises hardware, software, browsing and network capabilities).

On the other hand, the UED tools define an extensible set of properties of the usage environment (terminal, network and user's preferences), and they are based on XML Schema.

Each standard has its pros and cons. RDF provides a rich semantic description. The UED tools provide better support to specify constraints, cardinalities and data types. For a further comparison of these standards, please refer to Timmerer et al. [42]. The rest of this work utilizes the UED tools.

4 Semantic technologies

The semantic is the meaning of the symbols. At least two different technologies have addressed the problem of understanding the multimedia symbols: the Semantic Web and semantic multimedia adaptation technology. This section summarizes the state of the art of these technologies.

4.1 Semantic Web for multimedia

The Semantic Web [43] aims to represent knowledge in a format that can be automatically processed without human intervention. To this end, the machine must be capable of understanding the concepts and relationships thereby described. In this way, the machine can process knowledge, instead of multimedia symbols.

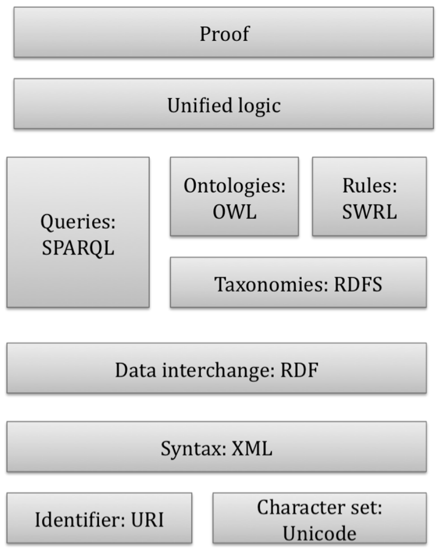


Fig. 5: The Semantic Web stack

The *Semantic Web Stack* [43] defines a stack of languages in which each layer uses the description capabilities of the layer under it (see Fig. 5). In this stack, upper levels provides a higher level of *expressiveness* (i.e., capability of describing knowledge). Specifically, the technologies up to RDF, *Web Ontology Language* (OWL) and *Simple Protocol and RDF Query Languege* (SPARQL) have been standardized and accepted (in Fig. 5 these technologies are named after a colon). However, it is not clear how to implement the technologies on the top of the stack (i.e., unified logic and proof, whose technology is not named in the figure). The term *ontology* is used to refer to the concepts (usually defined with a formal *vocabulary*) and relationships of a specific domain. The ontology is frequently represented with OWL creating a *semantic graph*.

Recall from Subsection 3.1 of this chapter that the term *metadata* refers to both information that the content creator manually adds and information that the machine automatically extracts, and the term *annotation* refers only to information manually added. The computer uses the ontology to automatically extract knowledge and therefore the ontology is a kind of metadata.

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

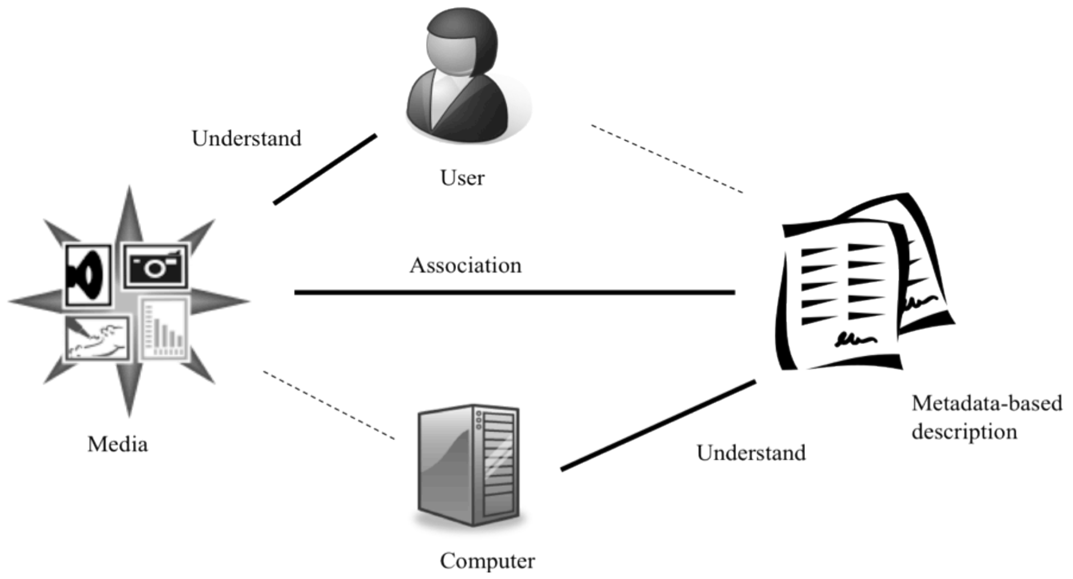


Fig. 6: Semantic description of multimedia content

The MPEG-7 and MPEG-21 standards make use of metadata to achieve a better understanding of multimedia. Particularly, these standards propose several informal vocabularies to represent a detailed description of the multimedia elements. The W3C Consortium has initiated a project to represent multimedia ontologies called *Multimedia Vocabularies on the Semantic Web* [44]. Even though this standard fully exploits the Semantic Web technologies, which have a higher level of expressiveness and ability to represent knowledge, at the time of writing, the majority of multimedia research relies on MPEG-7 and MPEG-21.

4.2 Reasoning and inference rules

Ontologies allow the formal representation of knowledge by means of description languages such as OWL. Specifically, the description languages allow the representation of a semantic graph, which contains concepts and their relationships (subclass, cardinalities, inverse relations, etc).

Ontology editors and a knowledge acquisition system (such as Protégé[6]) facilitate the management of this graph.

The technologies on the top of the Semantic Web stack use the semantic graph to obtain additional knowledge from existing knowledge. However, as the previous subsection stated, currently it is not clear how to implement the technologies on the top of the stack. To this end, AI *reasoners* [45] have found a promising new field of application. A reasoner is software that allows the extraction of new knowledge from existing knowledge. A multimedia ontology along with a reasoner have been used to make the machine capable of understanding both the content and the relationships among the multimedia descriptions (see, for instance, [46]). Once the knowledge is formally represented, the reasoner systematically applies *inference rules* to extract new knowledge. However, the results of reasoners are still insufficient to achieve the ultimate aim of the Semantic Web: the sharing, processing and understanding of data by automatic systems in the same manner as humans.

4.3 Semantic multimedia adaptation

The *multimedia semantic gap* is the lack of coincidence between the low level features in the media record (e.g., audio samples, video pixels, *Virtual Reality Modelling Language* tags, etc.) and the interpretation that a human would make of this media. The term *semantic multimedia adaptation* refers to a group of techniques that extract the semantic of the media in order to increase the user's satisfaction with the adapted media.

In [10], adaptation approaches are divided into *structural adaptation* and *semantic adaptation*. Structural adaptation exploits the scalability properties of the encoded bitstreams to create content that fulfils the usage environment constraints, but it does not take into account the meaning of this content. Metadata such as the one provided in MPEG-7 Parts 3 and 4, and in cooperation with the MPEG-21 Part 7 BSD tools, can be used to drive the structural adaptation.

Semantic adaptation focuses on extracting specific fragments that are of interest to the user. The usual result of a semantic-based adaptation is some kind of content summarization. In the case of video summarization, *Segments of Interest* (SOIs) can be used to indicate or select the more relevant *temporal* parts of the visual or audio stream [47][48]. In the case of image and video summarization, *Regions of Interest* (ROIs) can be used to crop or highlight the more important *spatial* parts of the video or image [49].

For instance, De Bruyne et al. [47] use the BSD tools to annotate the relevance of the different segments of the video content to create a tailored video, based on the user's preferences. Van Deursen et al. [10] have realized that in the standard XML-driven adaptation (i.e., BSD tools adaptation), the integration of semantic adaptation operations and semantic metadata standards is done in an ad-hoc manner. Therefore, they propose that RDF-driven content adaptation provide a high abstraction level for the definition of adaptations that best fits with semantic adaptation. This distinction between XML-driven adaptation and RDF-driven adaptation can be also found in [29], but these adaptations are named syntactic adaptation and semantic adaptation, respectively.

Another difference between structural and semantic adaptation is that, in general, structural adaptation does not summarize the content. Conversely, summarization is the most frequent application of semantic adaptation. In [48] semantic video summarization is seen as a special case of

[6] Publicly available at *protege.stanford.edu*

structural adaptation in which the meaning of the content is used to summarize the scalable bit-stream. In particular, the authors analyze the video in order to remove redundancies. In [50] semantic adaptation is classified into temporal, spatial and scene summarization.

Scene level adaptation (see Subsection 2.2 of this chapter) is another type of multimedia adaptation. For instance, Hori et al. [51] include *semantic labels* in their web pages. The semantic labels can be added manually or automatically. In their proposal, web pages have images and a text related to the images. If, due to the terminal constraints, an image is removed, then the related text is also removed. In [12], the author adds semantic labels to her MPEG-21 DIs in order to conduct semantic content selection.

In addition to the plentiful literature on multimedia adaptation by means of semantic summarization, some authors consider that content search and content retrieval are other kinds of semantic adaptation [52][53]. In this case, the lists of items that result from this search correspond to the adapted content.

5 Multimedia adaptation engines comparison

This section reviews eight multimedia adaptation engines that to some extent follow the representation schema proposed in the MPEG-21 framework. In Section 4 of Chapter 3 and Section 6 of Chapter 5, we provide comparisons between these engines and CAIN-21, the adaptation engine that we have developed during this work.

5.1 Existing multimedia adaptation engines

ConversionLink

Kimiaei [12] has studied the applicability of the standard *AdaptationQoS* description tool (see Subsection 3.3.4 above) to drive generic (scalable and non-scalable) resource adaptation. This investigation concludes by developing the ConversionLink[7] adaptation engine together with the *ConversionLink* description tool (see Subsection 3.3.5 above). The *ConversionLink* tool was later standardized in [38].

VRT

Soetens and Geyter proposed the VRT[8] adaptation engine in [54]. This engine allows for the composition of Semantic Web services in order to alter the original document to the needs of the target device. They use *OWL-Services* (OWL-S) to describe the adaptation chain and CC/PP to describe the target terminal capabilities. OWL-S is an ontology that builds on top of OWL to describe Semantic Web Services [55]. Specifically, they specify the transformations in terms of *Inputs, Outputs, Preconditions and Effects* (IOPE).

koMMa

[7] Note that the symbol ConversionLink is not italicized to refer to the adaptation engine, whereas it is italicized to refer to the description tool.
[8] The authors do not name their system. VRT (Vlaamse Radio- en Televisieomroep) is the name of the Belgian radio and television broadcaster in which the authors studied these ideas.

Jannach et al. [16] developed the *Knowledge-based Multimedia Adaptation* (koMMa) framework. As is the case with VRT, they also have exploited the Semantic Web Services to address interoperability. The main innovation is that koMMa demonstrates the use of AI planning (see Section 6 below) in multistep multimedia adaptation. They also have proposed an extensibility mechanism by means of pluggable Web Services.

MAGG

Berhe et al. [56] published their *Multimedia Adaptation Graph Service* (MAGG) a few months after koMMa. They have also proposed using an AI planner for multi-step adaptation. Their approach is different from koMMa because they focus on creating a *Direct Acyclic Graph* (DAC) that represents all the adaptation steps. They also show how to use Dijkstra's algorithm to find the optimal path.

BSD

The BSD description tools have been introduced in Subsection 3.3.3 of this chapter. Different authors have developed adaptation engines that study the use of the BSD tools. Timmerer has a complete and didactic description of the generic adaptation of scalable media resources [57]. A. Hutter et al. [58] have focused on the dynamic and distributed adaptation of bitstreams. De Bruyne et al. [47], Van Deursen et al. [10] and Zufferey et al. [50] have focused on semantic video adaptation and personalization of the bitstream. Van Deursen et al. [59] have also developed a description of the high-level structures to steer the adaptation of a binary media resource, using an efficient and format-independent parser named gBFlavor.

DCAF

Sofokleous et al. describe in [7] the *Dynamic Content Adaptation Framework* (DCAF). DCAF is the same adaptation engine in which Berhe et al. have tested their MAGG algorithm. In their work, Sofokleous et al. do not address multimedia adaptation, but show how to use heuristic genetic algorithms to identify the parameters of the *AdpatationQoS* description tool. The UED and UCD description tools are used to represent the context of the adaptation. The notion of Pareto optimality (see Subsection 7.5.3 below) is also introduced to rank the possible decisions.

NinSuna

Van Deursen et al. [34] have built on the *AdaptationQoS*, BSD and UED tools to develop the NinSuna adaptation engine. This engine provides both coding format-independence and packaging format- independence. The major innovation of this engine is its ability to leverage Semantic Web technologies to accomplish semantic adaptation. The semantics are explicitly represented by means of RDF tuples. In this way, the authors introduce formal semantics in the existing MPEG-21 adaptation description tools.

Early CAIN

Martinez et al. [60] developed the early version of *Content Adaptation and INtegrator* (CAIN) adaptation engine. The results of this thesis are demonstrated with CAIN-21, which extends CAIN description capabilities and algorithms.

Early CAIN received a media resource, an MPEG-7 description of this resource and an MPEG-21 description of the usage context. After its execution, CAIN produced adapted media along with its MPEG-7 media description. The version of CAIN published in [60] developed its modular architecture as shown in Fig. 7. In [60], CAIN comprised a *Decision Module* (DM) and a set of adaptation operations called *Content Adaptation Tools* (CATs), *Encoders* and *Decoders*.

In response to an external invocation, the multimedia resource, MPEG-7 description of this resource and an MPEG-21 description of the context were parsed. Then, the DM performed three main steps in sequence: (1) selection of the target media parameters, (2) selection of an adaptation operation capable of performing the adaptation and lastly, (3) the launch of the selected CAT/Encoder/Decoder.

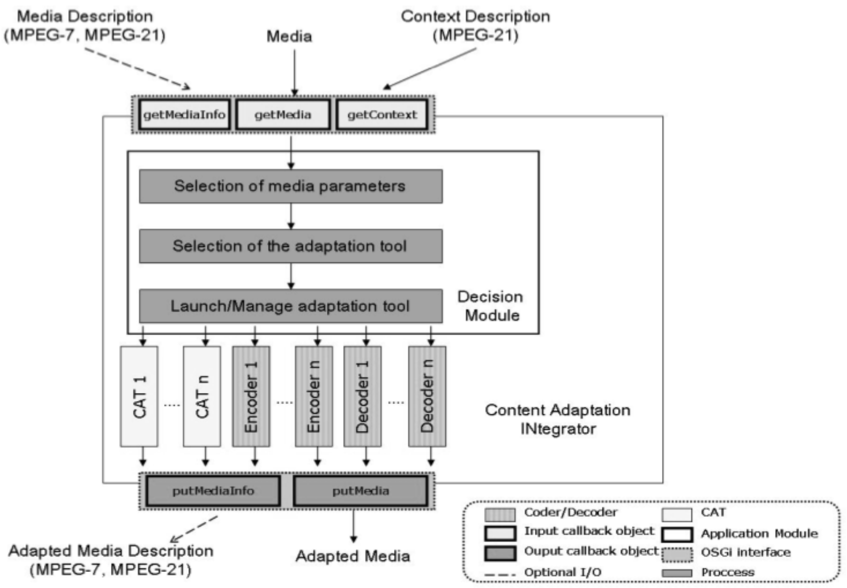


Fig. 7: Early CAIN

The next step in the development of CAIN (carried out at the beginning of this thesis) was in two parts [61]: (1) the development of an extensibility mechanism and (2) the development of a formal decision process capable of dealing with this extensibility. The extensibility mechanism proposes the use of a *CAT Capabilities* document to describe the adaptation capabilities of each pluggable CAT. The automatic decision mechanism was intended to select a CAT capable of performing the adaptation and the parameters to use.

5.2 Comparison

In summary, koMMa and MAGG focus on demonstrating how an AI planner obtains multi-step sequences taking into account the terminal constraints. However, their solutions do not elaborate on how to include the user's preferences or the utility of the adaptation (the user's experience). On the contrary, BSD, DCAF and NinSuna do not pay attention to the multi-step problem (some approaches for BSD multi-step adaptation can be found in [57]), but they rather focus on analysing the inclusion of the user's preferences and the signal level (e.g., objective quality-based) or semantic level (e.g., face detection) utility of the adaptation. CAIN-21 will combine and further elaborate on both techniques.

6 AI planning techniques

The koMMa and MAGG adaptation engines use AI planners to systematically identify the actions that adapt content to the constraints of the terminal. This thesis contributes to multi-step adaptation with AI planning techniques. This section subsequently surveys these standard AI planning techniques.

6.1 Planning

In AI, *planning* is the decision-making process that precedes acting [62]. Formally, a planning problem is made up of a finite and recursively enumerable *set of states* $S=\{s_1, s_2, ...\}$, a finite and recursively enumerable *set of actions* $A=\{a_1, a_2, ...\}$, and a *state transition function* $\gamma(s,a)$: $S\times A\rightarrow S$, which, given a specific state s_i and a specific action a_j, take us to a different state $s_{i+1}\in\gamma(s_i,a_j)$. The result of $\gamma(s_i,a_j)$ can be an empty set (i.e., $s_{i+1}=\phi$) if, for the given a_j, there is no subsequence state. In addition, each action a_j is associated with a *set of preconditions* (shortened to *pre*(a_j)) that must be true before the action can be executed and a *set of effects* (shortened to *effects*(a_j)) that describes how the state changes when the action is executed. Under such conditions, planning algorithms commit to finding the cumulative effects of these actions to search for sequences of actions that lead from an initial state to a goal state.

Moreover, *effects*(a_j) can be further divided into a *set of postconditions* (shortened to *post*(a_j)) that represent changes in properties of the state and a *set of invariants* (shortened to *invariants*(a_j)) that represent properties of the state that must not change, i.e., *effects*(a_j) = *post*(a_j) ∪ *invariants*(a_j). Traditionally, preconditions are represented as predicates that must be true before the action starts, postconditions are represented as predicates that must be true when the action terminates, and invariants are represented as predicates that keep their true value from the beginning to the end of the executing action.

6.2 Neoclassical planners

In the 1980s, the computational costs needed to solve the above-described planning problem using classical planners apparently could not be further reduced. However, in the 1990s the computational costs of planning systems were reduced with the rise of techniques that have been qualified as *neoclassical planners* [62]. The most remarkable approach was Graphplan [63]. The main difference between classical planning and neoclassical planning is that in classical planning every state of the search space represents to a single outcome in a partial plan, whereas in neoclassical planning states represent to the union of a set of outcomes that can be seen as a set of partial plans. In classical planning, actions were analyzed individually and fully instantiated. Conversely, neoclassical planning analyzes a partially defined set of actions. To this end, the *planning graph* serves to gather similar actions forming a partially defined *set of actions*. Additionally, Graphplan proposes building a reachability graph instead of a reachability tree to reduce the number of states that the planning algorithm has to expand. Even though the first implementation of this idea used forward search, further advances in this area included backwards search, i.e., with the goal state evaluated first (see for instance [64]).

6.3 Non-deterministic planning

This subsection introduces the notion of *non-deterministic planning* and focuses on the difference between bounded and unbounded non-deterministic planners.

Classical and neoclassical planners make two restrictive assumptions [62]:

- *Deterministic actions*: In a given state, actions always produce the same effects. That is, for each state s_i action a_j, if the action is applicable for the s_i state, it will lead to no more than a single state s_{i+1}, i.e., $|\gamma(s_i,a_j)|\leq 1$. The planner terminates the expansion of search paths in those states s_i in which for every action a_j it holds that $|\gamma(s_i,a_j)|=0$.
- *Full observability*: The planner can monitor all the relevant features of the world, meaning that it can recognize all the properties of the states.

Non-deterministic planning relaxes these assumptions. Specifically, non-deterministic planning introduces:

- *Non-deterministic actions:* Actions that under the same conditions (receiving the same input state) produce dissimilar outcomes, i.e., the exact outcome that is going to be produced is unknown before executing the action, i.e., $0\leq|\gamma(s_i,a_j)|\leq n$, where n is any natural number. For instance, during a manufacturing process the equipment may fail, or throwing a dice has several possibilities, none of them are certain. Thus, deterministic actions are a particular case of the non-deterministic actions in which $|\gamma(s_i,a_j)|\leq 1$.
- *Partial observability*: In some applications the state of the world is only partially observable, and as a consequence different states of the system become indistinguishable. Full observability is a specific case of partial observability in which all the states of the world are distinguishable.

Probably the main problem of non-deterministic planning is that the plan may result in different execution paths. The usual way to address this uncertainty follows three basic rules:

- *Outcome probabilities.* Non-deterministic actions are modelled by associating probabilities with the outcomes of the actions. This rule allows taking into account that some outcomes are more probable than others.
- *Belief states.* States are replaced by *belief states*, which associate a probability distribution across the state space.
- *Utility function.* Goals are represented via a *utility function*, i.e., numeric values that indicate the level of preference of each possible goal state. Under these circumstances, planning under uncertainty can be seen as an optimization problem where the objective of the planner is to maximize the utility function.

A non-deterministic planner can be *unbounded* or *bounded*. A *bounded non-deterministic planner* is one that can control the parameter of the action to limit the outcome to a subset of its potential instantiations. This thesis will focus on the implementation of a neoclassical non-deterministic bounded planner. To this end, Subsection 2.3 of Chapter 5 proposes the notion of *conversion states* and how preconditions and postconditions partially define a group of conversion states. Subsection 2.4 of Chapter 5 describes how such conversion states are bounded using so called *source and target parameters* and how uncertainty is handled using non-deterministic conversion states.

7 Preferences management

To manage preferences for the adaptation decision-methods proposed in this work, this section surveys the state of the art of preferences representation and elicitation methods.

7.1 Qualitative preference model

Game theory [65] has dealt with the representation of preferences in the following way. Let X define a *variable* over a finite *domain* $D(X)=\{x_1,x_2,...,x_n\}$ (also named *list of options* or *list of preferences*), where the x_i *values* (also named *options*) are *exclusive* between them (only one value must be selected). In such a framework, the term *outcome* is used to refer to the set of options selected by all the users (e.g., end-user, system administrator, system implementer, etc.). B. L. Slantchev [66] proposes the example of representing the user's preferences with a single variable X, where $D(X)$ represents the list of candidates in an election and the user has to choose which one to vote for. The outcome is the x_i candidate that the user has selected. The standard way to model the user's wishes is with a *preference relation*, also known as *priority list* or *ranking*. The preference relation on X represents the relative preference between the choices offered to the user (e.g., merit of the different candidates). In mathematics, a recognizable preference relation defined on the set of all real numbers is "≥" where "$x_1 \geq x_2$" is interpreted as "number x_1 is at least as big as number x_2". Similarly, the relation "L", interpreted as "is more liberal than", can be defined on the list of candidates where "$x_1 L x_2$" can be interpreted such as "candidate x_1 is more liberal than candidate x_2".

Generalizing the above notion of preference relations, the notation $x_1 \sim x_2$ is used to represent an *indifferent preference relation*, and the notation $x_1 \succ x_2$ is used to represent a *strict preference relation*. Both relations can be combined in a disjunctive form $\vee$ (i.e., or operator) within a *weak preference relation* represented as $x_1 \succcurlyeq x_2$. In this matter, often the literature draws on two basic assumptions [66]:

Assumption 1: Preference relations are *asymmetric*, that is, there is no pair of options x_1 and x_2 from the list of options $D(X)$ such that both preference orders hold. Under this assumption it can be easily inferred that $x_1 \succcurlyeq x_2 \Leftrightarrow (x_1 \sim x_2) \vee (x_1 \succ x_2)$ and $x_1 \succcurlyeq x_2 \Leftrightarrow \neg (x_2 \succ x_1)$.

An important characteristic of preference relations is rationality. A preference relation over a list of options $D(X)$ is *rational* if it is:

1. *Complete*: the user can determine whether he likes one option at least as much as any other, i.e., he knows all the preference relations between all the options x_1, x_2, ... $x_n \in D(X)$. In a more formal way: $\forall\ x_1,x_2 \in D(X)$, either $x_1 \succ x_2$ or $x_2 \succ x_1$ or $x_1 \sim x_2$.

2. *Transitive*: the user never introduces inconsistencies in preference relations of the options of the list of options, i.e., the user is aware of the implications of the $\succ$ and $\succcurlyeq$ operators. In a more formal way: $\forall\ x_1,x_2,x_3 \in D(X)$, if $x_1 \succcurlyeq x_2$ and $x_2 \succcurlyeq x_3$ then $x_1 \succcurlyeq x_3$.

Assumption 2: The *user is rational*, that is, capable of providing a rational preference relation.

7.2 Quantitative preference model

In economics, utility is a measure of the relative happiness or satisfaction of an economic agent. Given this measure, one may speak meaningfully of increasing or decreasing utility, and thereby explain the economic behaviour in terms of an economic agent attempting to increase its utility function.

Let's consider a set of alternatives $D(X)=\{x_1,x_2,...,x_n\}$. The *utility function* $u(x_i)$ is defined as a function that assigns a numerical value to every option $x_i \in D(X)$, so that the ranking order of these alternatives is preserved [66].

The term *ordinal* or *qualitative* preference is used to refer to a preference relation, i.e., to specify a ranking of alternatives. However, an ordinal preference relation says nothing about how far apart one option is from the others (the intensity). Conversely, the term *cardinal* or *quantitative* preference is used to refer to the distance between preferences. A utility function $u: X \rightarrow \mathbb{R}$ *rationalises* a preference relation $\succcurlyeq$ on $D(X)$ if the following equivalence holds: $\forall\ x_1,x_2 \in D(X)$, $x_1 \succcurlyeq x_2 \Leftrightarrow u(x_1) \geq u(x_2)$.

It is important to highlight that utility functions convey more information than preference relations. In fact, an infinite number of utility functions can represent the same preference relation. Consider, for example, a client that assigns a utility of 300 to a car, a utility of 200 to a bicycle, and a utility of 100 to a skateboard. When speaking about quantitative preference one can conclude that: (1) the client prefers a car to a bicycle or to a skateboard, and (2) that the car has the same utility as a bicycle and a skateboard altogether. When speaking about its corresponding preference relation, it would only be possible to say that the car is preferred to the bicycle and to the skateboard, but no more. At this point, it is worth emphasizing that human users do not have utility functions. Rather they have preference relations, which they can represent (for analysis purposes) by means of utility functions.

Note that a utility function can always be transformed into a preference relation (losing its intensity information), but instead the reverse is more difficult. There is an important theorem (developed by J. V. Neumann and O. Morgenstern [65]) that states that given a rational preference relation $\succcurlyeq$, it always can be represented by a utility function. Obviously, in such case the utility function cannot accurately represent the intensity of the options.

Another important conclusion [65] is that utility functions are unique up to positive affine transformations. Therefore, if $u_1(x)$ represents a preference relation, then so does $u_2(x) = a\ u_1(x) + b$, with $a \in \mathbb{R}^+$, $b \in \mathbb{R}$. Hence, without loss of generality, one can normalise any utility function to lie between 0 and 1.

7.3 Threshold preference model

A third way to represent preferences is with *threshold preferences* in which a specific level of the value of the preference must be fulfilled. For instance, *frame_width*≥176 is a threshold preference. Threshold preferences are Boolean preference variables; once the threshold has been achieved, the value of the variable is no longer considered. Therefore, threshold preferences can be transformed into a binary list of options with two values: *satisfied* and *not satisfied*.

Still another class of preferences is the *maximisation (minimisation) preferences* in which the objective is to maximise the value of the preference. Fig. 8 shows an example with two multimedia adaptation preference variables: X_1 with the number of frames per second (fps) and X_2 with the bitrate. The example gathers two preferences for these preference variables. The first preference is a threshold preference in which the user has indicated that he/she prefers an fps number of 15 or greater. The second preference is a minimisation preference in which the adaptation engine states its preference for minimising the bitrate. In this example, (as the figure shows) the optimum is the point where the threshold preference is satisfied and the minimisation preference reaches a minimum value.

There are well known numerical methods in the literature (e.g., Simplex [67]) for solving these kinds of problems using multiple lists of preferences represented as utility functions[9]. If one can assign numerical values to a list of options, then the preference variable can be represented as an utility function and then as a maximisation preference.

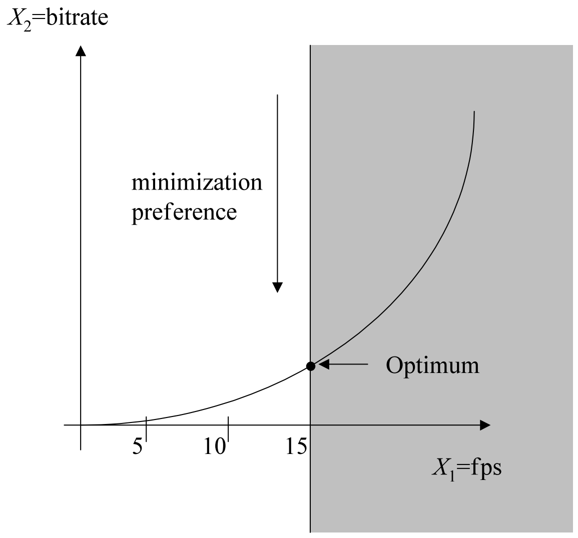


Fig. 8: Threshold and minimisation multimedia preferences

7.4 User preferences elicitation methods

In the fields of human recommendation and decision support systems, a great deal of research has been done on the subject of *user preferences elicitation methods* [68]. This subsection brings these elicitation approaches to the field of multimedia adaptation systems.

The ultimate goal of the user preferences elicitation methods is to guide the user through a sequence of queries that gather the user's real wishes. This information allows the computer system to produce better results than would be obtained without any knowledge of the user's preferences. In the context of the preferences representation methods (examined in the previous subsections), these elicitation methods correspond to the process of extracting the user's wishes, which are represented as preference relations or utility functions.

Extracting preferences from the user – especially with unskilled users – is generally an arduous process. The key for effective preferences extraction is the construction of tools that automate preferences elicitation, either partially or fully. To this end, a rich set of sophisticated preferences elicitation support tools has been researched [68].

With respect to the preferences representation methods (surveyed in the previous subsections), the kind of queries with which the elicitation system prompts the user, and the ideas developed in [68], we divide the elicitation queries into:

[9] See *http://plato.asu.edu/* for an exhaustive list of optimization methods, including a decision tree for choosing the more appropiate optimization method and software according to the nature of your problem.

1. *Outcome queries*. This type of queries presents the user with global queries (i.e., a set of options), and asks his/her preference for this set of options. This type of queries is applicable to two situations: 1) When the set of options does not present any structure and 2) when the set of options has a multiattribute structure, but its structure is ignored and only full or global outcomes are considered. Braziunas and Boutilier [69] explain that, in practice, people cannot meaningfully compare outcomes with more than five or six options. When there are more attributes, it becomes necessary to break down the outcomes into lists of options and to ask the user for each list of options separately. For example, consider the following query to the user about his/her preferences, "What do you prefer: A film in English about politics, a film in Spanish about soccer, or a podcast in English about technology?" Even this uncomplicated query with only three attributes (language, media and topic) seems more effective when one breaks down those outcomes according to their three constituent preference variables, "Which language do you prefer: English or Spanish?", "Which media do you prefer: video or audio?", and "Which topic do you prefer: politics, soccer or technology?".
2. *Ranking queries.* In this case, the outcomes have been divided (according to their attributes) into lists of options (also named lists of preferences). Usually each list of preferences represents the values (options) of an attribute. For example, the attribute topic could take the values politics, soccer or technology. After that, the user is asked to provide a preference relation for the options in each list of preferences. The simple way to obtain a ranking is to ask the user to *compare two options* of the list of options and to indicate his/her preference relation between each two options. This query usually requires little cognitive effort from the user. Unfortunately, they are not very informative. More complicate comparison queries ask the user to *pick the preferred option* from a set of *k* options. This rather easy task actually produces *k*-1 preference relations (the selected option is preferred to all remaining ones). At the most extreme, a *total ranking* query expects the user to rank all specified alternatives; answering such a query would provide preference information relating all the pairs of alternatives.
3. *Utility queries*. This type of queries presents the user with a list of options and asks the user to provide an assessment of the utility of each option. Frequently, if the query is a continuous variable X_i, it is simplified by asking the user to provide the estimated utility for a small set of values and, after that, the utility function can be generated over the whole range of the variable using interpolation techniques. For example, asking the user to assess the utility for three different frame sizes and after that, interpolating the utility of other frame sizes.

7.5 The preference graph

This section describes how preference graphs can be used to implement preference-based decision methods. A preference graph represents preference relations between outcomes. This representation allows the selection of the outcome that best suits the preferences of the user. These preferences can be represented according to the qualitative or quantitative models described above.

If we have only one preference variable X_i with several options $x_1,x_2,...,x_n \in D(X_i)$ the easy way to describe the preferences is with a linked list like the one shown in Fig. 9 (a).

However, in practice, decision problems are endowed with a multidimensional structure, that is, there is more than one preference variable, and their combination gives rise to *a space of variables* (also named *set of theoretical outcomes*) $\mathbf{X}=(X_1,X_2,...,X_M)$. In this case, it is necessary to represent the Cartesian product of the options of each variable, and then to represent the preference relations between the different outcomes $\mathbf{x} \in \mathbf{X}$. Under these conditions, a preference graph is a directed graph that represents the relationships among the outcomes. In particular, the outcomes correspond to the instance of the Cartesian product of the variables, and the term *outcome relation* refers to a directed arc between outcomes of a preference graph. The arrow of each arc means that the target node of the arc is preferred to the source node of the arc. Fig. 9 (b) shows an example of a preference graph where there are two variables X,Y with $x_1,x_2,x_3 \in D(X)$ and $y_1,y_2 \in D(Y)$.

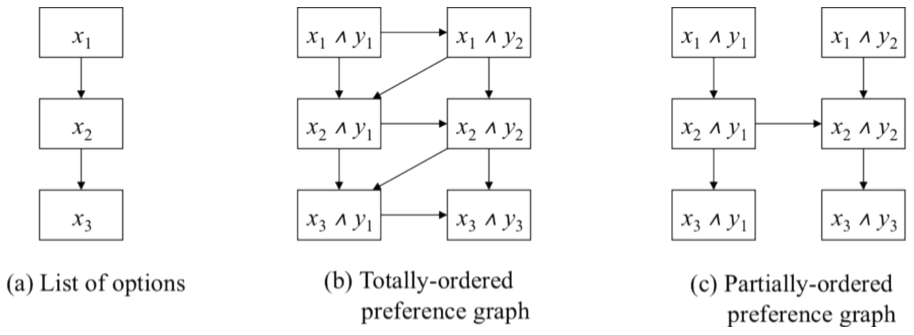


Fig. 9: Preference graphs

Please note that outcome relations must not be confused with preference relations. The former refers to relationships between the outcomes. The latter term refers to relationships between the options x_j of a list of options $D(X_i)$ of a single preference variable. For example, $(x_2 \wedge y_1) \succcurlyeq (x_1 \wedge y_2)$ is an outcome relation, whereas $x_1 \succcurlyeq x_2$ is a preference relation.

A preference graph is called a *totally-ordered preference graph* if there are outcome relations among all the outcomes, i.e., there is a forward or backwards path from each outcome to any other outcome. For instance, Fig. 9 (b) is a totally-ordered preference graph. Conversely, a preference graph is called a *partially-ordered preference graph* if there is no outcome relation among two or more outcomes, i.e., there is at least one pair of nodes with neither a forward nor a backwards path between them. Fig. 9 (c) shows a partially-ordered preference graph.

7.5.1 Building a preference graph

The preferences graph can be computed from the preference relations of a group of M variables according to the following algorithm:

1. Build a node for each outcome so that each outcome is a conjunction of M atoms $\mathbf{x} = a_1 \wedge a_2 \wedge \ldots \wedge a_M$. Each atom represents an option from its corresponding variable.
2. For each preference relation $a_i \succ b_i$ where $1 \leq i \leq M$, trace an arc from the outcome $a_1 \wedge \ldots a_i \ldots \wedge a_n$ to the outcome $b_1 \wedge \ldots b_i \ldots \wedge b_n$ so that their atoms only differ in a_i and b_i, that is, $a_1=b_1, \ldots, a_i \neq b_i, \ldots, a_M=b_M$.

Note that even though the user could provide all the preference relations of the entire set of variables in a qualitative way; such information is not enough to compute a totally-ordered preference graph. This is because what the graph contains is outcome relations (which are calculated from the preference relations). For example, in Fig. 10 (a) there is a total order with respect to the variables ($x_3 \succ x_2 \succ x_1$ and $y_2 \succ y_1$), but a partial order with respect to the outcomes. For instance, there is no path from $(x_1 \wedge y_2)$ to $(x_2 \wedge y_1)$. For the sake of completeness, Fig. 10 (b) provides an example of a preference graph with a total order with respect to both the preference relations and the outcome relations.

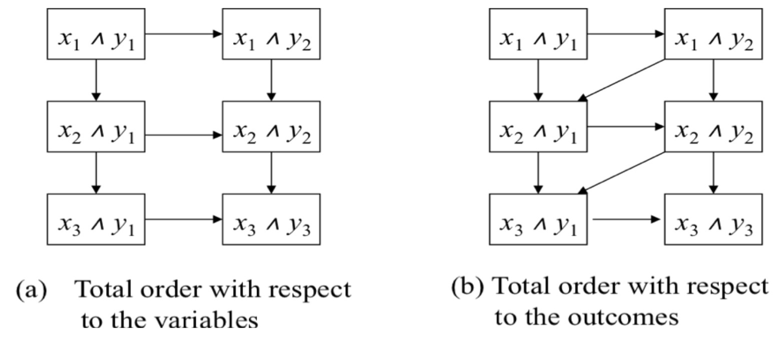


Fig. 10: Types of total order in a preference graph

In general, when the preferences are provided in a qualitative way, it is difficult to compute some outcome relation between non-adjacent nodes. The numerical utility value of the quantitative model allows for dealing with this partial order more effectively. In this case, the numerical preference values of each outcome are summed and the preference graph computation process decides that the outcome with a higher value is best.

7.5.2 Theoretical and feasible outcomes

At the beginning of Subsection 7.5, the term *set of theoretical outcomes* $\mathbf{X}=(X_1,X_2,...,X_M)$ was proposed for representing the Cartesian product of all the options that the preference-based decision system is considering. Hereafter, the term *theoretical outcome* $\mathbf{x}\in\mathbf{X}$ will refer to each element of the set of theoretical outcomes.

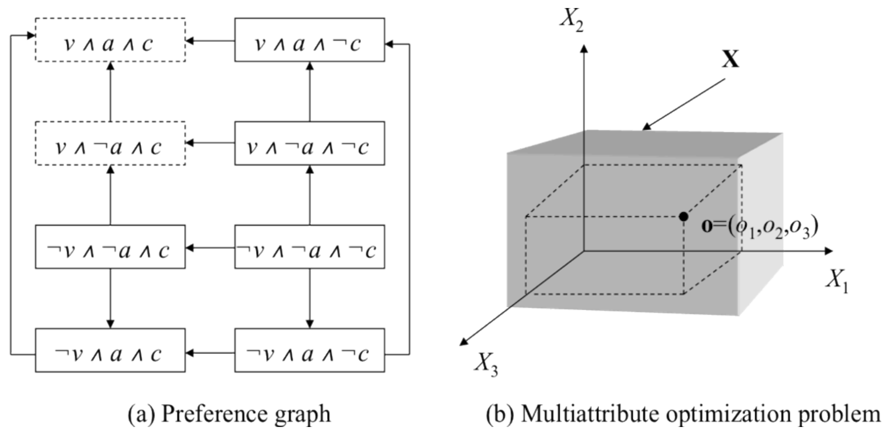

$v \wedge a \wedge c$
$v \wedge a \wedge \neg c$
$v \wedge \neg a \wedge c$
$v \wedge \neg a \wedge \neg c$
$\neg v \wedge \neg a \wedge c$
$\neg v \wedge \neg a \wedge \neg c$
$\neg v \wedge a \wedge c$
$\neg v \wedge a \wedge \neg c$
X_2
$\mathbf{X}$
$\mathbf{o}=(o_1,o_2,o_3)$
X_1
X_3
(a) Preference graph
(b) Multiattribute optimization problem

Fig. 11: Theoretical and feasible outcomes in a multiattribute problem

In practice, due to empirical constraints in the real world, often only a part of such theoretical outcomes is permitted. The term *set of feasible outcomes* ($\mathbf{O}$) will be used to refer to these permitted elements. Therefore, the $\mathbf{O}\subseteq\mathbf{X}$ relationship holds. Accordingly, the vector $\mathbf{o}\in\mathbf{O}$ will refer to a specific *feasible outcome*. For example, let us suppose that we have three preference variables for visual stream V, audio stream A and captions C, and that their domains (list of options) are respectively $D(V)=\{v, \neg v\}$, $D(A)=\{a, \neg a\}$ and $D(C)=\{c, \neg c\}$. If, for instance, the multimedia terminal can render visual stream, audio stream and captions, but visual stream and captions cannot be rendered at the same time, then Fig. 11 (a) shows the corresponding set of theoretical outcomes. The outcomes that appear in solid line boxes are the feasible outcomes. Arcs show the preference relations. This is a partially-ordered preference graph because there are no outcome relations between some outcomes. In this example, we know that the user would like to have the three variables enabled, that is, his/her preference ranking is $v \succcurlyeq \neg v$, $a \succcurlyeq \neg a$ and $c \succcurlyeq \neg c$; however, this outcome is not a feasible outcome.

The goal of multiattribute utility theory [65] is to investigate numerical representations that reflect the structure of the preference attributes over multiattribute spaces. Often, in order to represent preference-based decision problems, the utility functions are preferred over the preference relations. This is because, given a space of preferences $\mathbf{X}=(X_1,X_2,...,X_M)$, instead of examining conditions under which preference relations produce maximal level of satisfaction, it is easy to specify the numerical representation and to apply standard optimisation techniques to find the maximum. Thus, the "best" options for the space of list of preferences $\mathbf{X}=(X_1,X_2,...,X_M)$ are precisely the options that have maximum utility. Fig. 11 (b) shows the relations between these concepts from the multiattribute optimisation problem point of view. The shadow box represents the set of theoretical outcomes, the dashed box represents the set of feasible outcomes and the $\mathbf{o}$ point represents the optimal outcome from the point of view of the preference variables (X_1, X_2, X_3).

7.5.3 Optimal outcomes

Pareto optimality is an important concept in multiattribute utility theory [65]. Given two outcomes $\mathbf{o}_1,\mathbf{o}_2 \in \mathbf{O}$, we say that $\mathbf{o}_1=(o_1,o_2,...,o_M)$ *dominates* $\mathbf{o}_2$ if all the values of the M preference variables that comprise $\mathbf{o}_1$ are at least equal to or (perhaps) greater than the preference variable values of $\mathbf{o}_2$. For a given set of feasible outcomes $\mathbf{O}$, the term *Pareto frontier*, *Pareto set* or *skyline* [70] $\mathbf{O}^*$ is used to refer to the set of outcomes $\mathbf{O}^* \subseteq \mathbf{O}$ that are Pareto optimal. Note that if $\mathbf{o} \in \mathbf{O}$ and $\mathbf{o} \notin \mathbf{O}^*$ then $\mathbf{o}$ cannot be optimal. Therefore, the identification of Pareto frontiers yields all the potential optimal solutions. Once this frontier is obtained, the decision algorithm can focus on the tradeoffs within the outcomes in $\mathbf{O}^*$, rather than having to consider all the initial outcomes in $\mathbf{O}$. In this work the term *optimal outcomes* $\mathbf{O}^*$ will refer to that set of feasible outcomes $\mathbf{O}^* \subseteq \mathbf{O}$ placed within the Pareto frontier. Note that the optimal outcomes $\mathbf{O}^* \subseteq \mathbf{O}$ comprise a partially-ordered preference graph, which includes only the "best" (according to the Pareto optimality principle) outcomes. This work will address the problem of deciding which of these optimal outcomes to choose in the context of the multimedia user's preferences.

8 Conclusions

This chapter has surveyed the state of the art of multimedia adaptation and related technologies. Here ends the description of the current knowledge that the following chapters utilize. The content described in this chapter has paved the way for the contributions of the following chapters. This survey of the multimedia adaptation technologies has included multimedia adaptation and delivery, multimedia description standards and multimedia adaptation engines. The related technologies are the semantic technologies, AI planning and user preferences modelling and representation.

PART II:

CONTRIBUTIONS

Chapter 3: Architecture

Synopsis:

This chapter presents the CAIN-21 multimedia adaptation engine. The engine facilitates the integration of reusable and pluggable multimedia adaptation modules, chooses the chain of adaptations to perform and manages its execution. Evolving from CAIN, CAIN-21 complies better with the MPEG-21 framework. Its new features and improvements are discussed in this chapter. In addition, the pros and cons are explained with respect to other multimedia adaptation engines, including early CAIN.

1 Introduction

An adaptation engine is necessary for the study of multimedia adaptations. In this research, this platform is a middleware that allows for the creation, evaluation and validation of adaptation tests. In addition, to evaluate and validate the results it is important to make these adaptation tests storable and repeatable.

Frequently authors make several assumptions about the content to be adapted (e.g., in [71] the authors assume the existence of news items in the content). To generalize and systematise the adaptation scope, this work bases its decisions on media format descriptions. In this way, the scope of the content is not limited by the semantic of the content. In addition, the modality of the content can be also different (i.e., in our current version it can be video streams, images, audio or a combination of them).

The metaphor of the Babel fish[10] universal translator for multimedia [72] has been used to explain the idea of UMA. This metaphor also can be used to explain the objective of this chapter. As stated in Subsection 3.1 of Chapter 2, the MPEG-21 standard addresses the construction of a generic multimedia system that is consistent with the idea of UMA. This standard allows for the incorporation of multimedia content resources and their metadata. This standard gathers an exhaustive group of description tools, enjoys a significant level of acceptance by the multimedia community, and furthermore there is solid and ample background research literature on this description model. This chapter formalizes both the concepts to be managed and the description of these concepts. For these reasons, the MPEG-21 standard was chosen for this work.

Specifically, this work was initiated with the *Early CAIN* platform (described in Section 5 of Chapter 2). To accomplish the objectives of this work, this software has been extended producing the CAIN-21 platform (CAIN in the MPEG-21 framework) [73]. This software acts as a middleware in which adaptations can be easily integrated. The CAIN-21 platform is open-source. The *CAIN-21 software* together with a *CAIN-21 demo* is publicly available at *cain21.sourceforge.net*.

This chapter explains CAIN-21. The main objectives of this platform are:

- To perform systematic and automatic multimedia adaptation decisions. These decisions have been integrated in the *Planner* module. This module is introduced in this chapter and the underlying novel algorithms are examined in detail in Chapter 5 and Chapter 6.
- To support interoperability by means of an extensibility mechanism. It does not seem realistic that one single adaptation module will be able to perform every kind of multimedia adaptation. Therefore, this work researches on an extensibility mechanism in which reusable and pluggable tools are incorporated to progressively address wider ranges of adaptations.
- To support multi-step adaptation. The range of feasible adaptation increases if the reusable and pluggable adaptation modules can be combined and executed in several steps. The *Planner* is in charge of identifying the availability of multi-step adaptation solutions.
- To formalize the representation of the multimedia elements in order to make the adaptation tests storable and repeatable. The results of these adaptation tests will be compared with the results obtained in other adaptation decision methods in the literature.

[10] The Babel fish is a fictitious leech-like, which simultaneously translates from one spoken language to another.

- To determine the set of MPEG-21 elements that describes the multimedia system. The MPEG-21 framework is very comprehensive, but also complex. The elements of CAIN-21 have been chosen following the *Keep It Short and Simple* (KISS) design principle. This principle recommends to avoid unnecessary complexity and construct systems as simple as possible, but no simpler.
- To create a content agnostic decision mechanism that will be based mainly on the multimedia format description features.
- To enable static and dynamic decisions. *Static decisions* are format-independent and generic, rely on metadata, and do not access the content. Conversely, *dynamic decisions* are designed for specific media format and access the content to make additional decisions. In our platform, dynamic decisions are made in the pluggable tools.

CAIN-21 can be integrated within large-scale multimedia systems with the purpose of providing multimedia adaptation services. It has been successfully put into practice in the *Integrating Knowledge, Semantics and Content for User-centred Intelligent Media Services* (aceMedia) and *Multimedia Semantic Syndication for Enhanced News Services* (MESH) European projects, which are referenced in Appendix D.

In the remainder of this chapter, Section 2 explains the main features and elements of CAIN-21. This section also explains the new features and advantages that the new multimedia adaptation engine has incorporated into Early CAIN. Section 3 summarizes the evolution from Early CAIN to current CAIN-21. Section 4 provides a comparative analysis between CAIN-21 and other multimedia adaptation engines. Finally, Section 5 concludes the chapter and summarizes the advantages of the adaptation techniques explained in this chapter. The demonstrations and experiments involving this adaptation engine are in Chapter 7.

2 CAIN-21: functionalities and improvements

This section sequentially describes the CAIN-21 software interfaces, the interior architecture and the control flow.

2.1 External interfaces

CAIN-21 serves adaptation requests through two external software interfaces (see Fig. 12 below):

1. *Media level adaptation interface*. This performs *signal adaptation* (i.e., semantic-less) adaptation of a resource level composition. In addition to adapting a media level composition, this interface can also adapt a multimedia level composition, i.e., videos composed of one or more audio and visual streams. The adaptation operations are implemented in the *Tlib* module. This includes conventional software libraries such as *ffmpeg*, *imagemagick* as well as *Java Native Interface* (JNI) custom libraries.
2. *DI level adaptation interface*. This performs structure level (semantic or signal level) adaptations. In this case, metadata is used during the adaptation.

The DI level adaptation interface complies with the MPEG-21 representation schema. The *Content DI*[11] is a DI that conveys the media resource together with its metadata to be adapted. To drive the adaptation, CAIN-21 uses four *DIA* description tools. Only the *Content DI* and *DIA* de-

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

scription tools fully follow the MPEG-21 recommendations. Fig. 12 provides a view of CAIN-21 consistent with the idea of an MPEG-21 Part 7 adaptation engine (described in Subsection 3.3 of Chapter 2). From the point of view of these interfaces, CAIN-21 is a replaceable black box.

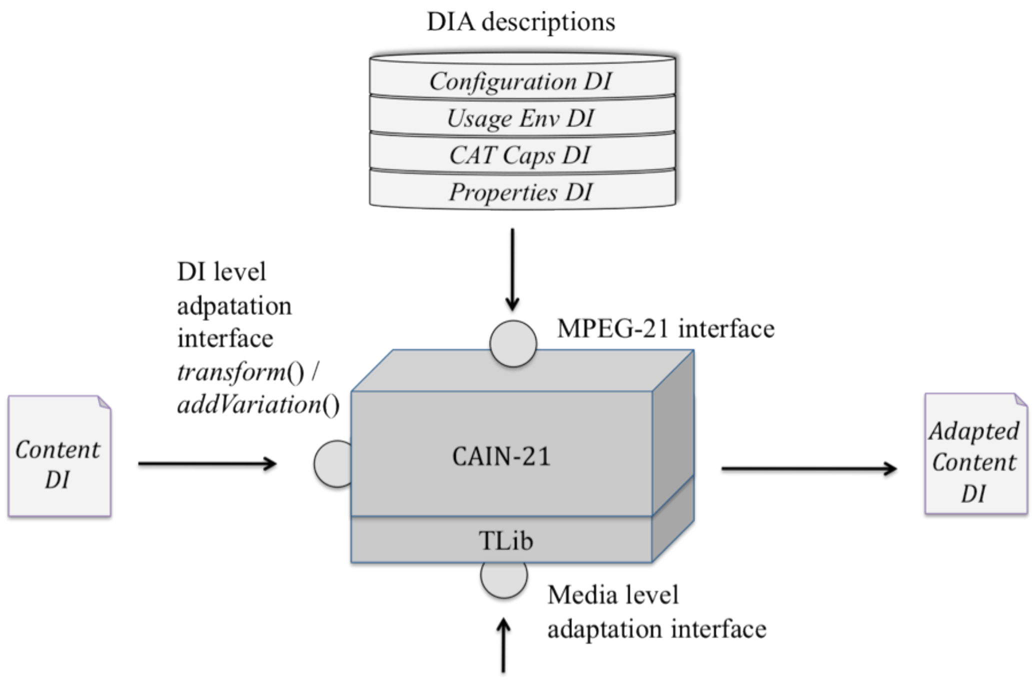


Fig. 12: Software interfaces of CAIN-21

In CAIN-21, metadata-based adaptation is performed through the DI level interface and at the *Component* level. An MPEG-21 *Component* includes a media resource (in the *Resource* element) and its metadata (in the *Descriptor* element). The *Descriptor* elements use MPEG-7 Part 3, Part 4 and Part 5 to describe the multimedia content. The DI level adaptation interface provides two different operations. The first one modifies the existing *Component* and the second operation adds a new *Component* element to the DI. More specifically:

1. The *transform*() operation takes a *Component* from the *Content DI* and modifies its media resource and metadata in order to adapt it to the usage environment.
2. The *addVariation*() operation takes a *Component* from the *Content DI* and creates a new *Component* ready to be consumed in the usage environment. At the end of the *addVariation*() operation, CAIN-21 adds this adapted *Component* to the *Content DI*.

2.2 Architecture

This subsection provides a detailed description of CAIN-21's modules, description tools in and control flow Fig. 13.

Manager module

The *Manager* module is responsible for coordinating the entire DI level adaptation process. The modules depicted below the *Manager* perform different tasks initiated by the *Manager*.

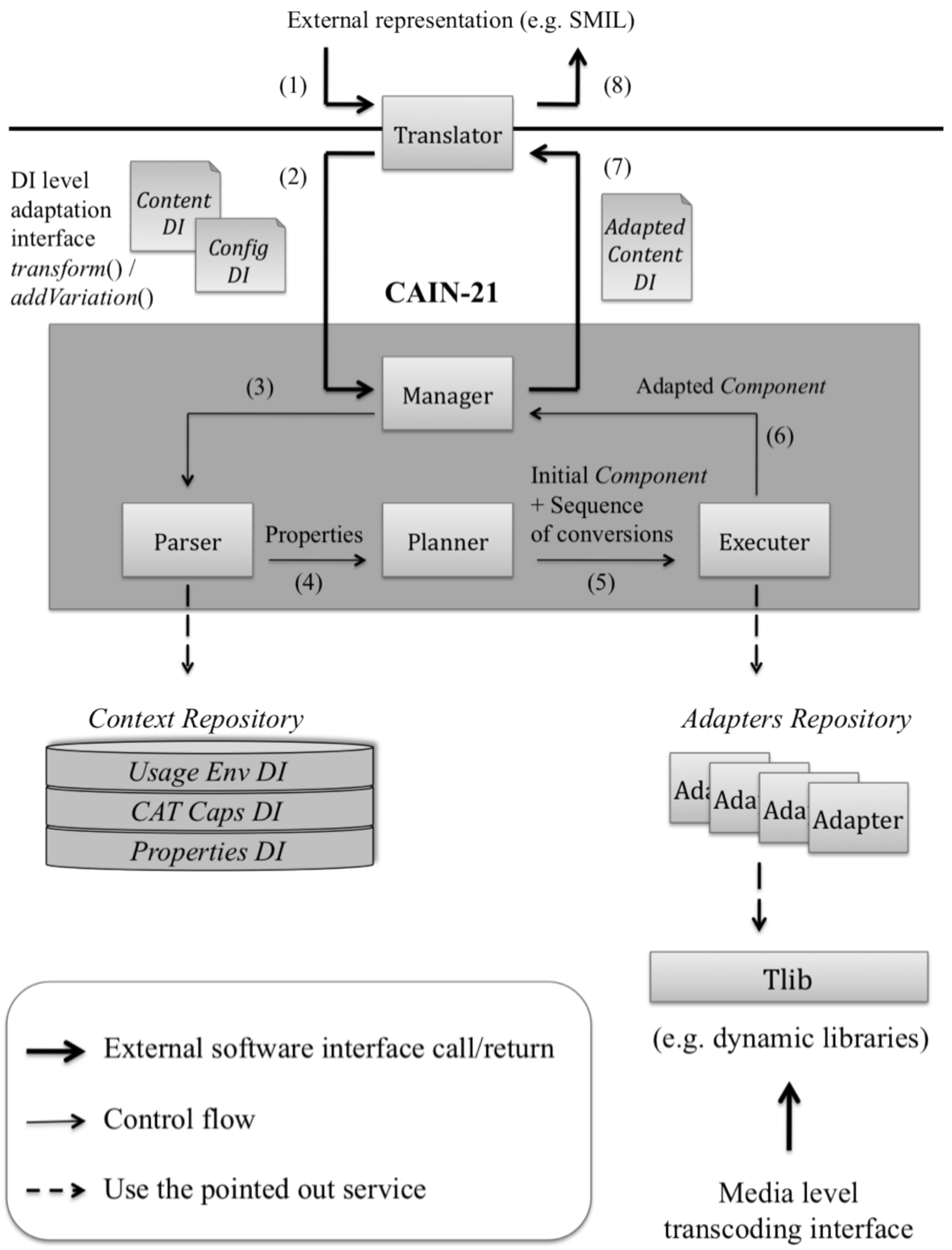


Fig. 13: Modules and control flow within CAIN-21

The *Planner* and *Executer* modules

Subsection 2.3 of Chapter 2 explained that, frequently, adaptation engines divide the adaptation process into two different phases. CAIN-21 also includes this distinction implemented in the *Planner* and the *Executer* modules, respectively.

Adapters and conversions

As explained in Subsection 3.3.5 of Chapter 2, MPEG-21 Part 7 defines a *conversion* as the process that changes the characteristics of a resource. In general, a conversion performs the act as defined by the MPEG-21 Part 6 term *adapt*. In CAIN-21, an *Adapter* is a reusable and pluggable adaptation module[12] that implements one or more conversions. For the sake of homogeneity, this work uses the term *tool* to refer to description tools and the term *conversion module* (instead of the MPEG-21 Part 7 term *conversion tool*) to refer to software or hardware processors. In this way, an *Adapter* implements one or more conversion modules. Multi-step adaptation allows for the sequential execution of conversions implemented in one or more *Adapters*. The *Planner* uses metadata to determine the sequence of conversions and parameters that should be executed over a *Component* element of the *Content DI*. Subsequently, the *Executer* is responsible for executing the corresponding sequence of *Adapters* on the initial *Component*. When an *Adapter* is executed, both the conversion to execute and the parameters of the conversion have to be provided. If CAIN-21 receives multiple requests to adapt the same content to the same usage environment, a caching mechanism speeds up this process through bypassing the execution of the *Planner* and *Executer* several times.

During their execution, *Adapters* have the option to append information to the *Descriptor* element of the *Component* so that subsequently *Adapters* can use it. As stated in Section 1 of this chapter, the term *static decisions* refers to metadata-based decisions. Static decisions do not depend on the resource content (only the *Descriptor*) and the *Planner* is responsible for these decisions. On the contrary, the term *dynamic decisions* refers to adaptation decisions that perform operations over the resource content. The *Planner* cannot make dynamic decisions because during the planning phase only metadata (and not the resource) is available and therefore dynamic decisions are transferred to the *Adapters*. These dynamic decisions usually increase the utility of the adapted content by means of semantic decisions or quality-based decisions[13]. Frequently, semantic decisions assume particular content (e.g., faces, soccer, news items, violent scenes in the movie). For example, in Subsection 4.2.1 of Chapter 7 it is assumed the existence of faces in the images. Utility-based decision methods have been introduced in Subsection 2.3 of Chapter 2, in depth described in Section 4 of Chapter 6 and demonstrated in Section 4 of Chapter 7.

Context Repository

As further explained in Subsection 2 of Chapter 4, CAIN-21 defines a type of DI referred to as *Context DIs.* These *Context DI* elements store *DIA* descriptions with information concerning the context in which the adaptation takes place.

The *Context Repository* in Fig. 13 includes three *Context DIs*. The *Usage Environment DI* describes the available usage environments using standard UED elements (i.e., instances of the UED tools). Each *Adapter Capabilities DI* describes the different *conversions* that an *Adapter* is able to perform. Each conversion has a set of valid input and output properties along with its corresponding values. The relationships among these elements are explained in more detail in Section 4 of Chapter 4.

[12] This work uses the term *adaptation module* to refer to a general group of conversion modules and the term *Adapter* to refer to its implementation in CAIN-21.

[13] Subsection 2.3 of Chapter 2 states that, in this work, *quality* refers to sampling fidelity, whereas *utility* refers to the end-user satisfaction towards the media content.

CAIN-21 includes an addressing mechanism in which changes in the metadata descriptors will not imply changes in the underlying source code. This mechanism is explained in detail in Subsection 6 of Chapter 4. The mechanism represents all the multimedia information by means of *properties*. Each property has one *key* and one or more *values*. The advantage of this representation is that it suits the decision mechanism explained in Chapter 5. The *Properties DI* is intended to store a set of keys and corresponding *xpointer*() [74] expressions providing access to the actual values. That is, the *xpointer*() expressions in the *Properties DI* are stored in the other DIs.

Configuration DI

The *Configuration DI* is a *DIA* description indicating which description of the terminal, network and user – from the ones available in the *Usage Environment DI* – to use during an adaptation request. Subsection 3.3.2 of Chapter 2 explained that MPEG-21 recommends using the *Choice* descriptor and *DIA Configuration* description tool to specify the adaptation to perform. CAIN-21 does not use this standard mechanism; instead it uses the *Configuration DI* to indicate the parameters of the adaptation to perform. Subsection 2 of Chapter 4 justifies this change and explains the advantages that this proposal yields.

***Parser* module**

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

***Translator* module**

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

2.3 Control flow

The numbers in Fig. 13 indicate the control flow of the tasks in the adaptation process, which is as follows:

1. When interacting with external systems, the *Translator* transforms the external multimedia representation into a *Content DI* that CAIN-21 can process.

2. The *Content DI* and *Configuration DI* arrive via the DI level interface *transform()* or *addVariation()* operations. The *Manager* is in charge of coordinating the whole DI level adaptation process. Although Fig. 13 does not explicitly show it, the *Manager* is in charge of transferring control to the *Parser*, *Planner* and *Executer*.
3. The *Manager* initiates the adaptation by transferring the control to the *Parser* so that all the metadata is collected as properties. Although other modules use the *Parser* (e.g., to create the adapted *Content DI*), for simplicity, the figure shows the *Parser* used only to extract the relevant properties.
4. The *Planner* receives these properties in order to select the adaptation to perform.
5. Subsequently, the *Executer* receives the initial *Component* to adapt and the sequence of conversions (together with the corresponding parameters). The *Executer* uses the *Adapters'* services to execute the sequence of conversions. In turn, the *Adapters* may use the *TLib* services to adapt the media resource. The *Adapters* may also change or append information to the *Descriptor* element of the *Component* so that the subsequent *Adapters* may use it.
6. Once all the conversions of the sequence have been executed, the *Executer* returns the adapted *Component*.
7. The *Manager* replaces or appends the *Component* to the adapted *Content DI,* depending on the DI level interface operation used (i.e., *transform()* or *addVariation()* operations).
8. Frequently, the adapted *Content DI* may need to be transformed to an external representation and in this case, the *Translator* performs this transformation.

3 From CAIN to CAIN-21

Subsection 5 of Chapter 2 described Early CAIN. CAIN-21, the adaptation engine that demonstrates the results of this research, is an evolution of Early CAIN. This section goes over the major changes.

3.1 Parsing

The first set of changes relates to the *Parser*. In Early CAIN, the *Parser* was in charge of parsing all the metadata: the instances of the MPEG-7 Part 5 *MediaDescriptionType* description tool that describe the media along with the usage environment (using the MPEG-21 Part 7 UED tools). The changes to the *Parser* have been in two areas:

1. The multimedia elements are now represented by means of the *Content DI*, *Configuration DI and Context DIs* as explained in Subsection 2.1 of this chapter. These elements are further decribed in Section 2 of Chapter 4.
2. The *Parser* required changes in the source code of CAIN if a new description was added or modified. CAIN-21 has introduced a mechanism in which metadata is dealt through properties stored in the *Properties DI*. This mechanism has been introduced in Subsection 2.2 of this chapter and further explained in Subsection 6 of Chapter 4. With this mechanism, changes in the metadata managed by the adaptation engine imply only changes in the *Properties DI*. Additionally, this is an on-demand mechanism, which means that only the values of the properties used by the decision-making process are evaluated.

3.2 Configuration of the adaptation

Early CAIN assumed the existence of only one *Terminal*, *Network* and *User* element in the UED. Different UED documents represented different adaptation environments. CAIN-21 has gathered and placed the UED in the *Usage Environment DI*. The *Usage Environment DI* is the first type of

Context DI in Fig. 13. With CAIN-21, more than one *Terminal*, *Network* and *User* elements can be stored in the *Usage Environment DI*. Once CAIN-21 is deployed in a multimedia system, the target *Terminal*, *Network* and *User* elements can be addressed by means of the *Adaptation Request Configuration* (ARC) description tools. This description tool will be explained in detail in Section 3 of Chapter 4.

As explained in Subsection 5 of Chapter 2, Early CAIN incorporated three types of adaptation modules (see Fig. 7): *Content Adaptation Tools* (CATs), *Encoders* and *Decoders*. The third major change in CAIN-21 gathered all of these modules under the concept of *Adapter*. In CAIN-21, adaptations are always performed at *Component* level. An MPEG-21 *Component* includes a media resource and its metadata. Early CAIN included only one operation named *adapt*() with four parameters [61]: the *input content*, the *output format*, the *output folder* and a *set of properties* reserved for future functionalities. CAIN-21 divided this operation into two different operations, *transform*() and *addVariation*() (already explained in Subsection 2.1 of this chapter). Additionally, in CAIN-21, the number of parameters is variable and determined as a subset of the properties gathered from metadata.

3.3 Description of the adaptation capabilities

Further advances in the development of the systematic and automatic decision mechanism motivated changes in the *AdapterCapabilities* description. These changes are explained in detail in Subsection 4 of Chapter 4. In a nutshell, to express disjunction in the adaptation capabilities, each *Adapter Capabilities DI* has to be divided into several instances of the *ConversionCapabilitiesType* description tool. To allow for the existence of multi-valued properties, the *AdapterCapabilities* description mechanism was changed. In this way the properties of the *Adapter Capabilities DI* may be single-valued (e.g., *format* = {*mpeg2*}), multi-valued (e.g., *color_space* = {*rgb*, *grayscale*}), ranges (e.g., *bitrate*=[100..400000]), or compound values (e.g., *frame_size*= {144x176, 288x352}). Ranges are also allowed in compound properties (e.g., *frame_size* = [10..5000]x[10..5000]). In the current *AdapterCapabilities* description model, each instance of the *ConversionCapabilitiesType* description tool contains preconditions and postconditions used by the systematic and automatic decision-making process explained in Chapter 5.

3.4 Adaptation modes

The adaptation of large media resources such as videos may imply long delays if the resource needs to be adapted before being delivered. Early CAIN only supported the *OdA* mode (explained in Subsection 2.4 of Chapter 2). With this mode, adaptation takes place as soon as the user requests the resource. The client characteristics, preferences and natural environment can be taken into account. However, if the resource adaptation process is time-consuming, the user has to wait until the whole resource is adapted. Hence, this type of adaptation is useful for small resources (e.g., images), but undesirable for long resources (e.g., video or audio). CAIN-21 introduces the *OnA mode* by which the media resource can start its delivery before the whole media resource has been adapted.

Lastly, in CAIN-21 a great deal of research has been done in order to automatically construct multimedia adaptation plans. As explained in [61], Early CAIN was, "*not truly extensible in the sense it is currently, that is, it was possible to add additional Adapters, but it was needed to code or recode some parts in the core of CAIN.*" In order to automate the decision-making process, the method explained in Chapter 5 was integrated into the *Planner* of CAIN-21. This approach uses a description of the input and output parameters of the *Adapters* as preconditions and postconditions, respectively. The method computes a sequence of zero or more *Adapters* together with their parameters in order to adapt the media to the usage environment. CAIN was only capable of selecting and executing one CAT, *Encoder* or *Decoder* in order to perform the adaptation. Now CAIN-21 can perform multistep adaptation, i.e., CAIN-21 can find and execute sequences of *Adapters* of any length. Another difference between Early CAIN and CAIN-21 is that the former was only capable of finding one of the feasible solutions to address the adaptation problem, whereas the latter computes all the feasible adaptation solutions (sequences of *Adapters*). Subsequently, only one of these sequences needs to be executed in order to adapt the content. Chapter 6 develops the methods to select the sequence best suited to the usage environment.

4 Multimedia adaptation engines comparison

This section provides a comparative review of six multimedia adaptation engines, which operate in the MPEG-21 framework: ConversionLink [12], koMMa [16], BSD [15], DCAF [7], NinSuna [34] and CAIN-21. These engines have been introduced in Subsection 5 of Chapter 2. As depicted in Table 1, where the publication year is also shown, the comparison is based on six aspects, namely:

1. The automatic *decision-making method* that the engine implements.
2. Whether the engine supports *multi-step* adaptation.
3. Whether the engine provides a *complete-solution*, i.e., finds all the solutions.
4. The *extensibility mechanism* (if any).
5. The *multimedia content* that the engine is prepared to adapt.
6. The *semantic adaptations* (See Subsection 4.3 of Chapter 2) that the engine considers.

4.1 Decision-making method

Subsection 2.3 of Chapter 2 divided automatic decision-making methods into utility-based methods and knowledge-based methods. koMMa and CAIN-21 rely on knowledge-based methods, whereas BSD, DCAF and NinSuna rely on utility-based methods. ConversionLink is a generic description engine that does not specify the algorithms used to make the adaptation decisions. BSD and DCAF engines use the notion of Pareto optimality. CAIN-21 also uses utility-based decisions during a second step (see, Subsection 4.2 of Chapter 6 for a further discussion on how CAIN-21 provides these utility-based decisions). Whereas BSD, Ninsuna and CAIN-21 rely on classical multi-attribute optimisation methods, DCAF exploits genetic algorithms to compute this optimization.

4.2 Multi-step adaptation

Section 2.3 of Chapter 2 introduced the advantages that multi-step adaptation provides. These advantages are frequently studied in knowledge-based methods. The koMMa and CAIN-21 adaptation engines use these methods. BSD is mainly devoted to performing the adaptation of the scalable resource in one step. Nonetheless, the authors have also studied the problem of distributed adaptation, which correspond to the idea of multi-step adaptation in different nodes [58].

4.3 Complete solutions

In reference to completeness, utility-based methods usually obtain a complete solution, i.e., all the feasible solutions are obtained and ranked: this is the case of BSD, DCAF and CAIN-21. More specifically, these engines create a ranking among the available solutions. Well-known quality metrics such as PSNR or VQM [77] are used to create this ranking. Regarding the knowledge-based methods, koMMa only extracts one solution. CAIN-21 analyses all of them using both the knowledge-based and utility-based decision methods. NinSuna and ConversionLink do not specify the completeness of their decisions.

	Conversion-Link	**koMMa**	**BSD**	**DCAF**	**NinSuna**	**CAIN-21**
Year	2005	2006	2008	2008	2010	2010
Decision-making method	Ad-hoc	Knowledge-based	Utility-based	Utility-based	Utility-based	Knowledge-based + Utility-based
Multi-step	No	Yes	Yes	No	No	Yes
Complete solutions	Unspecified	No	Ranking	Ranking	Unspecified	Knowledge-based + Ranking
Extensibility mechanism	Yes	Yes	No	No	Yes	Yes
Multimedia content	Images + Video+Audio	Image + Video	Scalable content	General video	Scalable content	Images + Video + Audio
Semantic adaptations	Scene adaptation	OWL description	BSD + IOPins	BSD + IOPins	RDF + BSD + SOIs	ROI

Table 1: Summary of the comparison for the multimedia adaptation engines

4.4 Extensibility mechanism

The idea of extensibility appears in ConversionLink, koMMa, NinSuna and CAIN-21. Both the *ConversionLink* description tool used in ConversionLink and *AdapterCapabilities* description tool used in CAIN-21 include the standard *ConversionCapabilitiesType* [38] description tool. The differences between these description tools will be discussed in Section 4 of Chapter 4. As [15] and [34] do not discuss about extensibility, it can be assumed that extensibility is not supported in BSD and DCAF. The authors of NinSuna discuss its extensibility, which is based on its format independence.

4.5 Multimedia content

BSD and NinSuna are particularly effective with scalable media, while DCAF deals with general video resources. ConversionLink, koMMa and CAIN-21 are intended to deal with a wider range of media resources. Specifically, ConversionLink and CAIN-21 can manage images, audio and video, whereas koMMa provides adaptation experiments involving images and video. The scalable content adaptation in BSD and DCAF is one of the adaptations that CAIN-21 incorporates. Moreover, Subsection 4.1 of Chapter 7 discusses how scalable video adaptation is carried out inside an *Adapter* called the *SVCAdapter*. The scalable content adaptation corresponds to the idea of resource conversion in the case of ConversionLink.

4.6 Semantic adaptations

In reference to semantic adaptations, ConversionLink allows scene level adaptation. It addresses the question of semantic adaptation of documents based on temporal, spatial and semantic relationships between the media objects. koMMa relies on Semantic Web Services to describe its adaptation capabilities and to identify the sequence of conversions. BSD and DCAF use the BSD and the *AdaptationQoS* with *IOPins* description tools. These tools were examined in Subsection 3.3.3 and 3.3.4 of Chapter 2. *IOPins* are linked to semantics labelling the video stream on a semantic level. NinSuna uses RDF to describe the semantic relationships used during its semantic adaptations. It also uses the BSD tools to provide semantic adaptation for the selection of SOIs as well as for frame-rate reduction. Another research area of NinSuna is extending the coding-format independence ideas to achieve delivery-format independence. CAIN-21 makes use of ROIs inside some *Adapters* such as the *Image2VideoAdapter*. Experiments involving semantic adaptation in CAIN-21 are reported in Subsection 4.2 of Chapter 7.

5 Conclusions

This chapter has described CAIN-21, the extensible and metadata-based multimedia adaptation engine developed for this research. In the rest of this thesis, CAIN-21 will serve the purpose of creating tests and experiments involving automatic multimedia adaptation decisions and extensibility.

Current multimedia adaptation engines [7][13][75][76] make a distinction between the decision phase and the execution phase. The general architecture of CAIN-21 also includes this distinction between the *Planner* and the *Executer*. In particular, the *Planner* is responsible for adapting the metadata and the *Executer* is responsible for adapting the content of the media resource. The distinction between the DAE and the RAE – explained in subsection 3.3 of Chapter 2 – can also correspond to the *Planner* and *Executer*.

An advantage of CAIN-21 is that it enables interoperability by means of a clear extensibility mechanism. This mechanism enables the progressive addition of multimedia adaption tools that allows the adaptation of a wide range of multimedia content. Indeed, CAIN-21 is theoretically able of managing all content that can be represented as a DI. To provide extensibility, CAIN-21 uses the notion of *Adapters*.

A second advantage of CAIN-21 is that it enables systematic adaptation, i.e., it utilizes the same decision methods to manage diverse multimedia content. The semantic-agnostic *Planner* makes the adaptation engine independent of the semantic of the content to be adapted (e.g., soccer, news items, violent scenes in the movie). The independence is achieved by making decisions according to the media format. In a latter step, the *Adapters* can perform semantic adaptation for particular or general content.

A third advantage of CAIN-21 is that it combines two major decision-making methods: knowledge-based and utility-based (see Table 1). In contrast to existing knowledge-based adaptation engines such as koMMa, CAIN-21 includes a complete algorithm, i.e., an algorithm that identifies all the feasible adaptations that produce content satisfying the usage environment constraints. Existing utility-based adaptation engines assume the existence of a specific media format. CAIN-21, however, can manage different kinds of media. In cases in which the utility-based decision methods can be utilized, the *Adapters* execute these utility-based adaptation.

A fourth advantage of CAIN-21 is that it liberates the user from making adaptation decisions. The *Planner* automatically identifies the combinations of existing *Adapters*, in different order or with different parameters, that properly adapt the content; Chapter 5 details this mechanism. The *Adapters* can also make automatic decisions involving the parameters of the adaptation; Chapters 6 details these decisions on the parameters of the adaptation.

A firth advantage of CAIN-21 is that it can adapt both MPEG-21 and non-MPEG-21 multimedia content. Even though CAIN-21 relies on the MPEG-21 framework to manage multimedia, it can also provide adaptations to external multimedia representation systems (non-MPEG-21 compliant multimedia systems) by means of the *Translator*. The MPEG-21-based description of multimedia system facilitates creating storable and repeatable validation tests. The next chapter discusses the innovative ideas that CAIN-21 incorporates to the MPEG-21 framework.

Chapter 4:

Extensions to the MPEG-21 schema

Synopsis:

CAIN-21 uses the description tools that MPEG-21 standardises. This chapter identifies a set of limitations and ambiguities in the description capabilities of MPEG-21. Subsequently, it proposes some extensions to the MPEG-21 description schema. The chapter justifies its addition to CAIN-21 in order to remove these limitations and ambiguities. It also discusses how these extensions make possible to address a new range of multimedia adaptation problems.

1 Introduction

The identification and representation of contextual information in a multimedia system is still an open issue. The evolution of multimedia systems may combine existing description tools with new description tools. The same concept can be represented with different description elements. In this case, one of these description tools will prevail or new description tools will arise for whatever reasons. As explained in Section 3.4 of Chapter 2, in addition to MPEG-21, there are other multimedia description standards. MPEG-21 does not cover 100% of the concepts. The majority of the concepts that MPEG-21 describes are not necessary for a particular instance of a multimedia system. Practice may help in the identification of the most useful description tools and less frequently used description tools would be relegated. The KISS principle (introduced in Section 1 of Chapter 3) recommends selecting the necessary description tools, but no more.

Throughout the design and development of CAIN-21, some problems were encountered using the MPEG-21 description tools. These problems were solved extending the set of description tools that MPEG-21 provides. This chapter is devoted to explain these issues.

The main objectives of this chapter are:

- To describe the practical handicaps in the MPEG-21 framework, which were identified during the development of CAIN-21.
- To formalize description tools addressing the identified problems, which have to be consistent with the KISS principle.
- To construct a multimedia representation mechanism that avoids the encountered ambiguities.

In order to achieve multimedia interoperability, the industry must use the same description tools, however this constraint does not apply during research. This thesis provides solutions to this problem both by creating new description tools and using the XML Schema extensibility mechanism to restrict or extend existing MPEG-21 description tools. In particular, we use standard XML Schema derivation by restriction and derivation by extension [25]. All these extensions have been implemented in CAIN-21 [78].

A different approach to address the expressiveness handicap in description standards has been proposed by [51]. This approach recommends creating an *external document*. The authors create an external document that is attached to the HTML document to be adapted. The main advantage of this approach is that it is backward compatible as web browsers that do not recognize the attached external document can safely ignore it. The XML Schema also provides this backward compatibility. The extensions can be safely ignored by an MPEG-21 browser without these additional functionalities. In addition, we consider that extending the MPEG-21 schema is a best solution for the point of view of future amendments to the standard.

In the remainder of this chapter, Section 2 describes how to represent the multimedia content, the usage environment and the configuration of the adaptation. Section 3 provides details on the configuration of the adaptation. Section 4 explains how to describe the adaptation capabilities of the *Adapters*. Section 5 addresses the problem describing media transferring through the *Adapters* up to the terminal. Section 6 describes a general and systematic mechanism to manage the multimedia properties avoiding ambiguities. Section 7 discusses some handicaps in the description of the usage environment and provides a solution. Finally, Section 8 recaps the most important findings of this chapter. These multimedia representation models are fully available in Appendix A.

2 Content DI, Context DI and Configuration DI

Section 3.2 of Chapter 2 explained that, in MPEG-21 framework, different DIs are used throughout the consumption and delivery chain. The DIs can be classified according to their purposes. One initial approach in the literature has divided the DIs into *Content DI*s and *Context DI*s. The *Content DI* is a DI intended to contain the media or multimedia resource and corresponding metadata. The *Context DI* is intended to contain a description of the usage environment. The notion of *Content DI* and *Context DI* has been surveyed by the MPEG-21 standard [79] although it has not been finally incorporated to the standard. However, some authors have informally used these notions in their systems [80][81].

Particularly, these authors have used the term *Context DI* only to refer to the usage environment [79][80][81]. In [78], this research proposes to extend the idea of *Context DI* to represent the context. Particularly, in CAIN-21 there are three types of context elements: the *Usage Environment DI*, the *Adapter Capabilities DIs* and the *Properties DI*. Subsection 2.2 of Chapter 3 describes in more detail these elements.

Furthermore, we have considered configuring the adaptation using the *DIA Configuration* description tools (described in Section 3.3.2 of Chapter 2). After an adaptation request, the *DIA Configuration* tools can be used to specify the target usage environment. Although there are scenarios in which the *DIA Configuration* tools are applicable, we have identified two limitations in the standard *DIA Configuration* mechanism:

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

This work proposes to overcome the first limitation moving the dependency information to a third DI, and so we propose three DIs:

1. The *Content DI* contains the media or multimedia resource and corresponding metadata.
2. The *Context DI* that acts as a database where usage environment, adaptation capabilities and metadata properties under consideration are stored.
3. The *Configuration DI* that encloses the *DIA Configuration* description.

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

The representation and use of the *Configuration DI* is the innovative part of this work. Next section describes this description tool that the *Configuration DI* conveys. The main aim of our proposal is that the *Content DI* will not be modified when the *Usage Environment DI* changes.

3 The ARC description tool

Section 3.3.2 of Chapter 2 described the two *DIA Configuration* description tools that MPEG-21 Part 7 standardises:

1. The *UserSelection/BackgroundConfiguration* elements indicate whether the *Selections* of the *Choice* have to be presented to the user or the system have to automatically choose one *Selection*.
2. *SuggestedDIADescriptions* indicates the *DIA Description* elements that the DI's author recommends for the adaptation.

Both methods assume the existence of a negotiation mechanism. Authors such as [31][80] have followed this approach incorporating the *DIA Configuration* description in the DI to be consumed. CAIN-21 is not a network agent (as in the *DIA Configuration* usage model developed in [4]) but a middleware providing an *Application Programming Inteface* (API). Section 2 of this chapter introduces the problem of selecting zero or one instance of the standard MPEG-21 Part 7 UED tools (i.e., *Terminal*, *Network* and *User*[14] elements) from the *Usage Environment DI*. If we relax the network agent negotiation assumption, we can use the *DIA Configuration* to specify the usage environment. CAIN-21 extends the *DIA Configuration* to provide this information, i.e., it defines a third *DIA Configuration* tool (not considered in MPEG-21). This extension is called *Adaptation Request Configuration* (ARC) tool. Consider, for instance, two terminals in the *Usage Environment DI*, a mobile and a laptop terminal. In this case, an *ARC* description can be used to indicate the target terminal. The *Content DI* and the *Usage Environment DI* can be deployed before starting the adaptation engine. On the contrary, the *ARC* description is only created when an adaptation is going to be executed.

3.1 ARC driven adaptation example

This subsection demonstrates the way in which the *ARC* tool is used in CAIN-21. When CAIN-21 is deployed in a multimedia system, we assume the existence of a repository of multimedia content represented as a group of *Content DIs*. For example, consider the *Content DI* in Listing 16 of Appendix A, which has only one *Component* element. This element contains a *Resource* element pointing out to the image and a *Descriptor* element. This resource is labelled in MPEG-7. Consider also the *Usage Environment DI* in Listing 22 of Appendix A, which contains standard MPEG-21 Part 7 *Terminal*, *Network* and *User* elements. Before a DI level adaptation could be executed, and *ARC* description must be provided into the *Configuration DI*. Otherwise, CAIN-21 would not be able to perform the adaptation because the *Usage Environment DI* is describing several usage environments. Each element in the *Usage Environment DI* has an associated unique ID. The *Configuration DI* contains an *ARC* description that uses these IDs to specify the usage environment for the adaptation. Listing 6 shows an example of an *ARC* description. In Listing 6, the terminal with ID *mobile_1* has been selected. The description of this terminal is stored in the *Usage Environment DI*.

```
<DIA xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<Description xsi:type="ARCType">
  <Terminal>
   <Id>mobile_1</Id>
```

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

```
    <OperationMode>
    //TerminalCapability[@type='dia:DisplaysType']/Display/
    DisplayCapability[@type='dia:DisplayCapabilityType']/Mode/
    Resolution/[@id="176x144"]
    </OperationMode>
   </Terminal>
   <Network>
    <Id>ethernet</Id>
   </Network>
   <User>
    <Id>reporter</Id>
   </User>
  </Description>
</DIA>
```

Listing 6: *ARC* description example

The *Terminal* description contains information essential to decide the adaptation (such as *Codec-Capabilities*, *Display*, etc). For this reason, the *Terminal* element is marked as mandatory in the XML Schema of the *ARC* tool. The *Network* and *User* descriptions have been left optional. If present, they would contain respectively the ID of the *Network* and the *User* in the *Context DI* (see Listing 6). The whole XML Schema of the *ARC* tool is available in Listing 23 of Appendix A.

3.2 Operation modes

In addition to the ID of the UED tools to use during the adaptation, the *ARC* description can store what we refer to as *operation modes*. An operation mode allows selecting parts of the UED tools whenever several modes could be used. For instance, the *Terminal* might support different spatial resolutions. In this case, the adaptation engine must be capable of selecting the best resolution for each adaptation request. Additionally, the *ARC* description could provide an operation mode indicating the spatial resolution. In this case, the adaptation engine must comply with the constraints of this operation mode.

4 Adapter Capabilities DI

The large quantity of multimedia adaptations that could be envisioned makes it unfeasible to implement all of them. Subsection 2.2 of Chapter 3 has introduced the notion of reusable and pluggable *Adapters*. Their adaptation capabilities are described in *Adapter Capabilities DIs*. One *Adapter* can be used as soon as this *Adapter* and its corresponding *Adapter Capabilities DI* are plugged into CAIN-21.

4.1 AdapterCapabilities and ConversionCapabilities

The notion of *AdapterCapabilities* was first introduced in [82]. Since then, a number of enhancements have been incorporated into CAIN-21. This work has carried out two main enhancements with respect to the original *AdapterCapabilities* model [82]. The following paragraphs describe these changes.

Subsection 3.3.5 of Chapter 2 explained that MPEG-21 Part 7 Amendment 1 defines a *conversion* as an (software or hardware) element capable of performing multimedia adaptation. The original *AdapterCapabilities* only allowed describing one conversion, while the final *AdapterCapabilities* can incorporate several conversion elements. During the development of CAIN-21, we observed the practical fact that conversion capabilities are not always easy to describe with only one conversion. With some types of adaptations, we need to divide the capabilities of an individual *AdapterCapabilities* element into several *ConversionCapabilities* elements. Consider, for example, an *Adapter* that is capable of accepting JPEG and PNG images, but PNG images are accepted only in greyscale, whereas JPEG images are accepted in both colour and greyscale. In this case, the *Adapter Capabilities DI* must be split into two separate *ConversionCapabilities*. The first *ConversionCapabilities* element states that PNG images are accepted in greyscale. The second *ConversionCapabilities* element states that JPEG images are accepted in both colour and greyscale.

The second major change implies the description of the values that properties can take. Section 2.4 of Chapter 5 explains that during the decision process, preconditions, postconditions and parameters can take several possible values (e.g., *format* = {*mpeg*-1, *mpeg*-2, *mpeg*-4}). We have modified the description of the conversions so that each input and output property can take multiple values.

The XML Schema of the *AdapterCapabilities* description tool that we propose is available in Listing 24 of Appendix A. An instance of the *AdapterCapabilitiesType* represents the capabilities of an *Adapter*. The instances of the *ConversionCapabilitiesType* description tool represents each conversion that the *Adapter* is capable of performing. Listing 7 shows a fragment of one of the *Adapter Capabilities DIs* fully available in Appendix A. The *Adapter* comprises two *ConversionCapability* elements named *ondemand_mpeg_transcoder* and *ondemand_mp4_transcoder*. The *Preconditions* and *Postconditions* elements contain information related to the media format that each conversion accepts and produces. These description elements are inspired in those from MPEG-7 Part 5 [6]. Note that the values of these descripton elements can be single-valued or multi-valued by means of the *ValueSet* element. The *RangeValueSet* element enables the description of ranges. The *AnyValue* element represents a placeholder whenever the value of the parameter must be provided, but for this element every value is acceptable.

```
<dia:DIA xmlns="urn:vpu:cain21-cat-capabilities"
         xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS"
         xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <dia:Description xsi:type="AdapterCapabilitiesType" id="video_transcoder_cat">
  <AdapterClassName>es.vpu.cain21.cats.OnDemandVideoTranscoderAdapter</AdapterClassName>
  <Platform>
   <ValueSet>
    <Value href="Windows XP">Windows</Value>
    <Value href="Linux">Linux</Value>
    <Value href="Mac OS X">Mac OS X</Value>
   </ValueSet>
   </Platform>
   <!-- Ondemand MPEG conversion using the ffmpeg command -->
   <ConversionCapability xsi:type="ConversionCapabilityType"
                         id="ondemand_mpeg_transcoder">
    <ContentDegradation>0</ContentDegradation>
    <ExecutionCost>1.0</ExecutionCost>
    <Preconditions>
     <URL>
      <AnyValue/>
     </URL>
     <Binding>
      <ValueSet>
       <Value href="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:HTTP">HTTP</Value>
       <Value href="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE">FILE</Value>
      </ValueSet>
     </Binding>
     <Content>
```

```
      <ValueSet>
       <Value href="urn:mpeg:mpeg7:cs:ContentCS:2001:2">Audiovisual</Value>
       <Value href="urn:mpeg:mpeg7:cs:ContentCS:2001:4.2">Video</Value>
      </ValueSet>
     </Content>
     <FileFormat>
      ............
     </FileFormat>
     <Bitrate>
      <RangeValueSet from="5000" to="1000000"/>
     </Bitrate>
     ..............
    </Preconditions>
    <Postconditions>
     ................
    </Postconditions>
   </ConversionCapability>
   <!-- On Demand MP4 conversion using the ffmpeg command -->
   <ConversionCapability xsi:type="ConversionCapabilityType"
                         id="ondemand_mp4_transcoder">
    ..................
    ..................
  </ConversionCapability>
 </dia:Description>
</dia:DIA>
```

Listing 7: *Adapter Capabilities DI* example

4.2 Comparison with the standard ConversionCapabilities

Subsection 3.3.5 of Chapter 2 explains that MPEG-21 Part 7 Amendment 1 has standardised the *ConversionLink* description tool. This tool provides a means for linking steering description parameters and conversion capabilities description. *ConversionLink* uses the *mpeg21*:*ConversionCapabilities* description tool to describe the adaptation capabilities. Each capability is represented with an instance of the *mpeg:21ConversionCapability* description tool (the instance is named as *ConversionCapability*).

The *catc*:*AdapterCapabilitiesType* description tool of CAIN-21 is defined as a derivation by restriction of the standard *DIADescriptionType* description tool of MPEG-21. Therefore, the *catc*:*AdapterCapabilitiesType* description tool can be seen as a (non-MPEG-21 standardised) *DIA* description tool. The *catc:AdapterCapabilitiesType* description tool conceptually corresponds to the *mpeg21:ConversionCapabilitiesType* description tool. The main difference is that the *catc:AdapterCapabilitiesType* description tool derives from the *mpeg21*:*DIADescriptionType* description tool whereas the *mpeg21:ConversionCapabilitiesType* description tool derives from the *mpeg21:TerminalCapabilityBaseType*. We chosed *mpeg21*:*DIADescriptionType* as the base for the *catc:AdapterCapabilitiesType* description tool to remove any dependence on the terminal.

Subsection 3.3.5 of Chapter 2 explained that the *mpeg21:ConversionCapabilityType* description tool is a generic container that can contain any proprietary description. CAIN-21 defines its own *catc:ConversionCapabilityType* description tool by means of a derivation by extension of the standard *mpeg21:ConversionCapabilityType* description tool. Therefore, the *catc:ConversionCapabilityType* description tool can be seen as a particular case of the generic *mpeg21:ConversionCapabilityType* description tool that MPEG-21 provides. In particular, CAIN-21 describes the conversions by means of preconditions and postconditions. This description model suits the automatic decision mechanism that Chapter 5 introduces. Listing 24 in Appendix A provides the XML Schema of the *catc:ConversionCapabilityType* description tool.

Authors such as [12] also use the *ConversionCapability* description tool[15] together with the *ConversionLink* description tool. In this case, the author makes use of RDF tuples to describe the adaptation capabilities and their semantics. CAIN-21 instead uses preconditions and postconditions that best suit its decision-making mechanism. The authors of the adaptation engine in [14] use the term *ADME Profile* to refer to the adaptation capabilities description document.

5 Binding modes

Subsection 2.4 of Chapter 2 has highlighted the difference between adaptation and delivery. CAIN-21 supports *OffA*, *OdA* and *OnA* adaptation modes. In addition, to support media delivery, CAIN-21 introduces the *binding modes*. In particular, delivery can be envisioned as a type of adaptation. The binding modes indicate the delivery mechanism that the conversion uses to receive and transmit the media (such as *FILE*, *HTTP* or *RTSP*). This work proposes to use the *mpeg21:Handler* description tool of the BBL (see Subsection 3.3.3 of Chapter 2) to represent the binding modes. The binding modes are used with two purposes: (1) to transfer the media between *Adapters* in a sequence of *Adapters* and (2) to transfer the media from the last *Adapter* in the sequence to the consumption terminal. Table 2 shows the binding modes currently supported in CAIN-21. The *INPROCESS* binding mode allows efficient transfer of the media resource between *Adapters*.

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

Table 2: Binding modes proposed by CAIN-21

In CAIN-21 each *ConversionCapabilities* element must provide in its preconditions and postconditions the available *binding modes*. For instance, in Listing 7, the first *ConversionCapabilities* element supports *FILE* and *HTTP* in its preconditions (i.e., in the input of the corresponding conversion). The *Terminal* element of the *Usage Environment DI* must also indicate the delivery modes that it supports to receive media. Listing 9 below will show how the binding modes of a terminal are provided into its terminal description. Listing 7 and Listing 9 will show that the binding mode, of both the *ondemand_mpeg_transcoder* and the terminal, can take more than one

[15] The author of [12] uses the name *ConversionDescription* to refer to the notion of *ConversionCapability*.

value. In these examples, both the conversion described in Listing 7 and the terminal described in Listing 9 support the *FILE* and *HTTP* binding modes.

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

6 Properties DI

In early versions of CAIN, the *Parser* (introduced in Subsection 2.2 of Chapter 3) was in charge of parsing the UED and *AdapterCapabilities* (the idea of DI did not exist in early versions). For each document, it produced an in-memory hierarchical tree of objects. After that, the *Planner* went through those hierarchies searching for the values needed to make decisions about the adaptation to perform. This course of actions proved tedious to implement and difficult to maintain. Consider, for instance, that a change in the position of an element in the document implies identifying the corresponding positions in the source code where this element is accessed and updating the algorithm at all these points.

Those troubles motivated the idea of gathering all the information required by the multimedia adaptation process following a declarative approach. Specifically, this information is described consistently using so called *multimedia properties*. These multimedia properties include the *Content DI*, the *Usage Environment DI* and the *Adapter Capabilities DI*. The *Properties DI* contains all these properties. Each property is represented as a label with an associated XPath [32] expression.

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

The expression of each property points out to the part of the DI where its values are located. XPath expressions are relative to the document. Therefore, the *Properties DI* stores only the XPath of the property. The document that contains these properties is determined during the execution of the adaptation. The *Configuration DI* (introduced in Section 2 of this chapter) is used to identify these documents. Furthermore, properties are only resolved on-demand. In this way, properties that are never used are not extracted from de DIs. Internally, CAIN-21 uses *xpointer*() [74] expressions to reference both the document and the XML element or attribute to be accessed. The standard Xalan processor [83] is used in this work to gather all these properties.

The proposed *Properties DI* schema is available in Listing 27 of Appendix A. The *PropertiesDIType* description tool is defined as a derivation by restriction of the MPEG-21 standard *DIADescriptionType* description tool. This type includes four important elements that correspond to the five groups of properties: *DIProperties*, *ComponentProperties*, *AdapterProperties*, *ConversionProperties* and *UsageEnvProperties*. Listing 8 shows the most relevant parts of the current *Properties DI* of CAIN-21 (the whole document is available in Listing 28 of Appendix A).

```
<dia:DIA xmlns="urn:vpu:cain21-properties-di"
         xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS"
         xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <dia:Description xsi:type="PropertiesDIType">
  <DIProperties>
    <Property name="genre" required="false"
              xpath="/Item/Descriptor/Statement/Mpeg7/DescriptionUnit/Genre/@href"/>
  </DIProperties>
  <ComponentProperties>
   <Property name="id" required="true" xpath="/@id"/>
   <Property name="url" required="true" xpath="/Resource/@ref"/>
   <Property name="mime_type" required="false" xpath="/Resource/@mimeType"/>
    ..................
   <ComposedProperty name="visual_frame" required="false">
    <Value xpath="//Mpeg7/Description/MediaInformation/MediaProfile
                 //MediaFormat/VisualCoding/Frame/@width"/>
    <Value xpath="//Mpeg7/Description/MediaInformation/MediaProfile
                 //MediaFormat/VisualCoding/Frame/@height"/>
   </ComposedProperty>
  </ComponentProperties>
  <AdapterProperties>
   <Property name="id" required="true" xpath="/@id"/>
   <Property name="cat_class_name" required="true" xpath="/AdapterClassName"/>
   ....................
  </AdapterProperties>
  <ConversionProperties>
   <Property name="id" required="true" xpath="/@id"/>
   <Property name="content_degradation" required="true" xpath="/ContentDegradation"/>
   <Property name="computational_cost" required="true" xpath="/ExecutionCost"/>
   <!-- Input properties -->
   <Property name="pre_url" required="true" xpath="/Preconditions/URL"/>
   <Property name="pre_binding" required="true" xpath="/Preconditions/Binding"/>
   <Property name="pre_content" required="true" xpath="/Preconditions/Content"/>
    ..............
   <!-- Output properties -->
   <Property name="post_url" required="true" xpath="/Postconditions/URL"/>
   <Property name="post_binding" required="true" xpath="/Postconditions/Binding"/>
    .................
  </ConversionProperties>
  <UsageEnvProperties>
   <TerminalProperties>
    <Property name="id" required="true" xpath="/@id"/>
    <Property name="binding" required="true"
   xpath="/TerminalCapability[@type='cde:HandlerCapabilitiesType']/Handler/@handlerURI"/>
     ..........
   </TerminalProperties>
   <NetworkProperties>
    <Property name="id" required="true" xpath="/@id"/>
    <Property name="max_capacity" required="false"
              xpath="/NetworkCharacteristic/@maxCapacity"/>
    <Property name="min_guaranteed" required="false"
              xpath="/NetworkCharacteristic/@minGuaranteed"/>
   </NetworkProperties>
   <UserProperties>
    <Property name="id" required="true" xpath="/@id"/>
     ..............
    <Property name="pref_focus_of_attention" required="false"
              xpath="/UserCharacteristic/ROI/@uri"/>
   </UserProperties>
  </UsageEnvProperties>
 </dia:Description>
</dia:DIA>
```

Listing 8: *Properties DI* example

For instance, in Listing 8 the *ConversionProperties* element contains the property *pre_url* whose XPath expression is "*/Preconditions/URL*". On resolving this XPath expression in Listing 7, *AnyValue* is obtained indicating that the conversion accepts any value on this property. As another example, on resolving the *pre_binding* property for Listing 8, the *FILE* and *HTTP* binding modes are obtained from Listing 7.

7 Extensions to the UED

Subsection 2.1 of Chapter 3 described multimedia content representation in CAIN-21. The multimedia content is represented with a *Content DI* including one or more *Component* elements. The properties of the *Content DI* and of the *Component* elements are represented with one or more *Descriptor* elements. This *Descriptor* element uses MPEG-7 Part 3, Part 4 and Part 5 to describe the multimedia content. Subsection 4.1 of this chapter explained that the *Preconditions* and *Postconditions* elements of the *Adapter Capabilities DI* describe media formats and are inspired from MPEG-7 Part 5. The *Usage Environment DI* relies on the MPEG-21 Part 7 UED tools to describe the usage environment. Listing 28 in the Appendix A gathers the 107 properties that, at the time of writing, CAIN-21 uses. The relevant properties from the standpoint of this discussion are summarized in Table 3 and Table 4. Specifically, Table 3 gathers MPEG-7 properties used in the *Content DI* and *Adapter Capabilities DI*. Table 4 gathers MPEG-21 properties used in the *Usage Environment DI*.

Property key	**Description**
url	Resource file
content	Modality of the resource: *Video*, *Audio* or *Image*.
format	System format of the resource
bitrate	Bit rate of the system resource
visual_format	Visual stream format of the resource
visual_bitrate	Visual stream bit rate
audio_format	Audio coding format of the resource
audio_bitrate	Audio stream bit rate

Table 3: Properties of the multimedia content (MPEG-7)

Property key	**Description**
binding	Delivery mechanism: *INPROCESS*, *FILE*, *HTTP*, *RTSP*
transport_decoding_format	Container formats supported by the terminal
image_decoding_format	Image formats supported by the terminal
video_decoding_format	Visual formats supported by the terminal
visual_bitrate	Maximum bit rate in the visual stream
audio_decoding_format	Audio formats supported by the terminal
audio_bitrate	Maximum bit rate in the audio stream

Table 4: Properties of the usage environment (MPEG-21)

7.1 Semantic gaps between MPEG-7 and MPEG-21

Chapter 5 will develop an automatic decision process that obtains a sequence of *Adapters*. In this sequence, the properties of the *Adapters* are described with MPEG-7 (see Table 3). Conversely, the capabilities of the terminal are described in MPEG-21 (see Table 4). This change in the description format gives rise to the following semantic gaps:

1. The *mpeg7:Content* description indicates the modality of the multimedia content (images, video, audiovisual, audio, etc). This property is used both in the description of the *Component* and in the postconditions of the *AdapterCapabilities*. The *mpeg7:ContentCS* classification scheme standardises the available values for this element. The *content* property in Table 3 corresponds to this value. The standard *mpeg21:TerminalType* description tool, however, does not includes any reference to the modality of the content that the terminal consumes.
2. The standard *mpeg21:TerminalType* description tool does not provide any description of the binding modes (such as *HTTP* or *RTSP*), i.e., the delivery mechanism used to consume content explained in Section 5 of this chapter.
3. The standard *mpeg21:TerminalType* description tool indicates the terminal capabilities, but does not specify whether the properties of the terminal are mandatory or optional. For instance, if the terminal is defined using the *mpeg21:AudioCapabilitiesType* description tool, does it mean that the adapted media must include audio? Or does it mean that this audio format could be consumed if present?

7.2 Inferred properties and ambiguity

Some missing properties in the MPEG-21 Part 7 UED tools can be inferred from existing properties. In this case, the implicit semantics of the UED tools and reasoning come into play. Other missing properties cannot be solved just by means of reasoning and therefore give rise to ambiguities. In this case, to remove these ambiguities, the elements of the standard UED tools must be extended. Specifically:

1. The media content (image, video, audiovisual, audio) can be inferred from the standard *mpeg21:TransportCapabilitiesType* description tool, which provides the container format (e.g., 3GPP).
2. The binding modes of the terminal cannot be inferred from the elements of the standard *mpeg21:TerminalType* description tool. Media can be delivered to the terminal using different delivery mechanisms (such as *FILE*, *HTTP* or *RTSP*) whilst the terminal may not necessarily support all of them.
3. Mandatory and optional properties of the terminal cannot be inferred if this information is not provided in the terminal description.

Therefore, these semantic gaps include both properties that can be inferred and properties that cannot be inferred (ambiguities). In particular, in the first gap the media content can be inferred from the *mpeg21:TransportCapabilitiesType* description tool (illustrated in Listing 9). The second and third gaps demand an extension of the standard *mpeg21:TerminalType* description tool and are addressed in the next subsection.

7.3 Proposal for extension

To address the previous semantic gaps, this subsection proposes to extend the current *mpeg21:TerminalType* description tool. Listing 9 shows the description of the terminal labelled *id="iphone"* in the CAIN-21 demo. The extensions that this subsection discussed are marked in bold. The XML Schema with these changes is available in Listing 21 of Appendix A.

In particular, this work proposes two extensions to the standard *mpeg21:TerminalType* description tool:

1. The delivery mechanism is fundamental in deciding the adaptation, as different *Adapters* may implement different delivery mechanisms. As explained in Section 5 of this chapter, the delivery mechanism is indicated in the binding mode. The BBL has inspired all of these modes. This section proposes to represent the binding modes of the terminal in the *cde:HandlerCapabilitiesType* description tool. This element makes reference to the *mpeg21:Handler* description tool. Listing 9 shows how to describe that the iPhone terminal supports the *FILE* and *HTTP* binding modes.
2. To describe whether a particular capability of the terminal is mandatory or optional further description is necessary. Mandatory and optional constraints are instances of the hard and soft constraints model developed in Section 2 of Chapter 6. To make this description, this section proposes to extend the *mpeg21:TerminalCapability* description tool with the *optional* attribute. Listing 9 shows how to signal that the audio stream is optional using the *optional* attribute in the *cde:AudioCapabilitiesType* description tool. If this attribute is absent, CAIN-21 considers the terminal description as a mandatory constraint.

```
<Terminal id="iphone" xsi:type="cde:TerminalType">
 <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
  <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
  <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:HTTP"/>
 </TerminalCapability>
 <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
  <cde:Decoding xsi:type="cde:TransportCapabilitiesType">
   <cde:Format href="urn:vpu:cs:FileFormatCS:2009:3gpp">
    <mpeg7:Name xml:lang="en">
     3GPP file format
    </mpeg7:Name>
   </cde:Format>
  </cde:Decoding>
  <cde:Decoding xsi:type="cde:VideoCapabilitiesType">
   <cde:Format href="urn:vpu:cs:VisualCodingFormatCS:2007:1">
    <mpeg7:Name xml:lang="en">
     H.264 Baseline Profile @ Level 1.1
    </mpeg7:Name>
   </cde:Format>
   <cde:CodecParameter xsi:type="CodecParameterBitRateType">
    <BitRate >32000</BitRate>
   </cde:CodecParameter>
  </cde:Decoding>
  <cde:Decoding xsi:type="cde:AudioCapabilitiesType" optional="true">
   <cde:Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.3.1">
    <mpeg7:Name xml:lang="en">
     MPEG-2 Audio AAC Low Complexity Profile
    </mpeg7:Name>
   </cde:Format>
   <cde:CodecParameter xsi:type="CodecParameterBitRateType">
    <BitRate>7950</BitRate>
   </cde:CodecParameter>
  </cde:Decoding>
 </TerminalCapability>
  ....................
</Terminal>
```

Listing 9: Extended *mpeg21:TerminalType* description tool

Subsection 3.2 of Chapter 7 develops several tests that demonstrate and validate these extensions.

8 Conclusions

At the time of writing this thesis, the industry has not opted for any multimedia description standard. Ad-hoc multimedia formats and systems description is the prevalent approach. If multimedia description is not standardised, the paradigm of UMA will not be attained. Currently, MPEG-21 is the most comprehensive multimedia description standard for the deployment of multimedia applications/systems. It relies on XML Schema to represent the elements. Although HTML is a simple multimedia description format, it has been widely accepted. The same happens with the nowadays-dominant multimedia systems relying on HTTP web servers. Description languages with a higher level of expressiveness such as the one proposed by the Semantic Web already exists.

The evolution of the future multimedia description systems is unpredictable. This work considers that a higher level of expressiveness (and corresponding complexity) in the multimedia description will be accepted only if it provides more added values. CAIN-21 aims to provide this added value by incorporating extensibility and automatic adaptation. Extensible systems tend to demand the inclusion of new descriptions for the concepts or relationships as pluggable adaptations are incorporated. CAIN-21 embraces MPEG-21 and demonstrates that most of these concepts can be represented with this standard. Even though MPEG-21 is a comprehensive description model, this chapter has identified and discussed several handicaps in its description capabilities. The solutions to these handicaps have also been discussed in this chapter. Probably future limitations may be encountered in the current description model of CAIN-21. In this case, existing MPEG-21 description tools or additional description tools can be progressively incorporated.

This chapter concludes the construction of the adaptation engine. The chapter has described how to use *Properties DI* as a simple and effective way in order to remove the ambiguities that the use of an implicit ontology may produce. Chapter 5 will demonstrate that the level of expressiveness of MPEG-21 framework is enough to perform automatic multimedia adaptation by means of AI planning techniques. These techniques achieve the same results that AI planning with an explicit ontology and reasoning but without the complexity of using an explicit ontology (see Appendix B).

Chapter 5:

Selection of feasible adaptations

Synopsis:

This chapter describes a method that, provided with a description of the multimedia content, the adaptation capabilities of the adaptation engine and the usage environment, performs systematic and automatic selection of feasible adaptations. The method allows creating this selection in situations where the adaptation capabilities are not completely known. In this approach, the adaptation planner selects a set of feasible adaptations and transfers the decision on which of these adaptations to perform to a subsequent step. The chapter lays outs the condition under which a partial knowledge of the adaptation capabilities is enough to compute a proper adaptation plan. In addition, this work demonstrates that this partial knowledge of the adaptation capabilities simplifies their description and speeds up the decision process.

1 Introduction

In current multimedia adaptation systems (such as *ffmpeg* or *imagemagick*), the user is responsible for deciding the parameters of the adaptation. These parameters determine the format of the adapted content. Thus, in these systems the user has to know what adaptations the available modules can make, what the adaptation parameters mean and which multimedia formats can be consumed in his/her terminal. Clearly, assigning the user the responsibility of understanding these capabilities hampers the transparent and ubiquitous use of multimedia. Furthermore, in these *manual multimedia adaptation* systems the user often has to take into account additional constraints such as the network capabilities or the delivery mechanisms. As Chapter 1 has justified, *systematic and automatic multimedia adaptation* system aims to identify and execute the adaptation modules and the parameters for multimedia adaptation. The main advantage of making this decision systematically is that the decision method can be used to adapt dissimilar content. The main advantage of making this decision automatically is that the adaptation system releases the user from this responsibility and in this way facilitates the use of multimedia.

The main purpose of this chapter is to describe a systematic and automatic selection system that identifies all the conversions and parameters that produce such adapted content. Often, terminals and networks can consume different adapted content. In this case, the selection system encounters different parameters for producing different adapted content. Chapter 6 addresses the problem of deciding which conversions and parameters are the best among the feasible ones produced by the method described here. The user's preferences are frequently considered during this decision. In the rest of this thesis, the term *selection* refers to the process of identifying the feasible adaptations that produce adapted content and the term *decision* refers to the process of choosing one of these feasible adaptations.

Section 5 of Chapter 2 has surveyed several multi-step multimedia adaptation engines, i.e., those that enable the adaptation of multimedia content in several steps. To comply with this idea, Subsection 2.2 of Chapter 3 introduced the term *conversions* to refer to these steps. In addition, this subsection described how these conversions are implemented in the *Adapters*. The mainstream research [16][54][56] in multi-step multimedia adaptation has relied on MPEG-21 to describe the multimedia adaptation *problem* and on different methods to find the *solutions*. These solutions correspond to the feasible computational processes that can convert the multimedia content into adapted content. This reseach has encountered that a frequent limitation in current adaptation systems is that they assume the existence of specific multimedia content, format or usage context (e.g., [59] assume the existence of standard scalable extension for H.264/AVC or JPEG 2000 images). As a result, in these systems, the adaptation occurs in an ad-hoc manner, and so these solutions cannot be easily extended to new kinds of adaptation problems. To fully achieve the ideal of universal interoperability this research focuses on developing a generic, systematic and automatic decision-making system (i.e., a system that does not assume the existence of specific multimedia content, format or usage environment).

In *Early CAIN*, this research was initiated by developing a method [84] that performed multimedia adaptation decision-making in a general manner by solving a *Constraint Satisfaction Problem* (CSP). An advantage of this method is that it is generic, in the sense that it is applicable for every kind of multimedia content, format and usage environment. The method is also systematic, because the same procedure can be used to adapt different types of content. However, the main limitation encountered, which was the motivation for the work in this chapter, was the difficulty in applying this mechanism to problems requiring more than one conversion step. Then, this research was refocused on the use of AI planners [45][62] to address multi-step adaptation. During this research, we developed an AI planning-based automatic decision method [85] for CAIN-21 that is also generic, systematic and automatic. This decision method was implemented in the *Planner* module. This chapter focuses on describing this *Planner*, the innovations that this decision method proposes and its advantages with respect to the existing multi-step decision methods (such as [16][54][56]).

The development of a decision system that could be used with any kind of multimedia is a big effort. A general design principle is to divide big and complex problems into smaller problems and to address them separately. Keeping in mind this principle, we have divided decisions into three *decision points*: the selections, the static decisions and the dynamic decisions. The *selections* are implemented in the *Planner* module, content independent and metadata reliant, and they may obtain several feasible solutions. The *static decisions* are also implemented in the *Planner*, which rely on the metadata describing the user's preferences and do not have access to the resource. The *dynamic decisions* are implemented in the *Adapter*s (and launched from the *Executer* module), specific for a kind of content, and have access to the resource. These decision points are executed in sequence. First, we execute the *selections*, then the *static decisions* and finally the *dynamic decisions*. In general, the selections and static decisions are executed more quickly than the dynamic decisions, as the former two only use metadata.

The main objective of this chapter is the development and evaluation of a *Planner* module which, provided with a *Content DI*, a *Configuration DI* and *Context Repository* (see Fig. 13 of Chapter 3), computes all the feasible sequences of conversions and parameters that produce this adapted content. This overall objective can be divided into different sub-objectives, purposely:

- *Multimedia properties*. All information required in the adaptation process is consistently described with multimedia properties. Multimedia properties refer to all the MPEG-21 elements of the multimedia system that have been described in Chapter 4. To access these properties, CAIN-21 implements a highly efficient addressing mechanism described in Section 6 of Chapter 4.
- *Domain-specific planner*. The *Planner* has to fulfil the objectives of CAIN-21, make quick multimedia adaptation decisions, and all with a small memory footprint. Standard planners, such as the one in [16], or the use of standard plan representation schemes, such as *Planning Domain Definition Language* (PDDL) [62], were deliberately excluded from this research.
- *Multi-step adaptation*. The available conversions can be combined in sequence, and so the *Planner* has to determine the parameter values (represented as properties) that can be used for these conversions.
- *Tolerating partial description*. To facilitate a compact description of the *AdapterCapabilities* and *ConversionCapabilities*, the *Planner* includes a mechanism that provides semantics for absent properties.
- *Alternative environments*. The *Planner* supports alternative usage environments by means of multi-valued properties (e.g., *frame_size* = {320x240, 640,480}, *network_capability* = [0..200000]). Even though general AI planning research has considered alternative goals (see, for instance [86]), previous research in multi-step multimedia adaptation has not taken into account this option [16][54][56][87], which characterizes many practical scenarios.
- *Static and dynamic decisions*. In CAIN-21, the static decisions are made in the *Planner* and the dynamic decisions are transferred from the *Planner* to the third party pluggable *Adapters*. Different *Adapters* allow dividing the responsibility for different kinds of dynamic adaptation decisions. These dynamic decisions allow the *Adapters* to produce different outcomes, which

are represented with multi-valued properties. Moreover, the multi-valued properties of the *Adapters* are combined with the multi-valued properties of the terminal. Consider, for instance, a specific *Adapter* capable of receiving *format*={*mpeg*-1, *mpeg*-2} and producing a video with *format*={*mpeg*-2, *mpeg*-4, *flv*}. If the *Planner* has selected these parameters, the *Adapter* can produce any of these outputs and the subsequent *Adapter*/terminal will accept all these video formats.

The structure of this chapter is as follows. Section 2 provides a conceptual view of the *Planner* module and its elements. Section 3 sketches practical problems that may occur with a complete and rigid multimedia description model and proposes semantics that lessen this rigidness. Section 4 describes the main algorithm of the *Planner*. Section 5 analyzes these algorithms, proves the finiteness of the algorithms, and in addition it proves the finiteness and completeness of the plan obtained. Section 6 provides a comparison between the *Planner* and other multi-step multimedia adaptation solutions. Section 7 describes the limitations and difficulties that we have identified in the proposed solution. Finally, Section 8 concludes the chapter. The demonstration and experiments involving this *Planner* are in Section 2 of Chapter 7. The major findings are collected in Section 1 and 2 of Chapter 8.

2 Planning with multimedia conversions

This section describes our domain-specific planner, how it uses the multimedia properties and how it selects the conversions of the multi-step adaptation process. The static decisions that the *Planner* implements are described in Chapter 6.

2.1 Conversion modules that make decisions

Previous multi-step multimedia adaptation engines [16][54][56] have used classical or neoclassical AI planners. The goal of these planners is determined by the constraints of the terminal. These planners do not take into account either the idea of using alternative parameters for the conversions or that of different alternative goals. After the planning phase, the conversions are executed in sequence. One of the contributions in our approach is postponing the decision from the *decision phase* (the selections and static decisions in the *Planner*) to the *execution phase* (the dynamic decisions in the *Adapters* launched from the *Executer*). In this new approach, the *Planner* uses the *AdapterCapabilities* to determine the constraints that the *Adapters* must obey during their dynamic decisions. Therefore, in our approach, the *Planner* only makes a selection (i.e., a partial decision) and offloads some decisions to be made later by the *Adapters*.

Subsection 4.1 of Chapter 7 describes scalable video adaptation tests to exemplify (demonstrate) this kind of dynamic decisions. In these demonstrations, the *Adapter* uses the *AdaptationQoS* description to decide which is the best layer from the ones satisfying the usage environment constraints. Subsection 4.2 of Chapter 7 describes dynamic decisions that demonstrate the use of ROIs (see Subsection 4.3 of Chapter 2) to improve the result of the adaptation. The *OnDemandVideoTranscoderAdapter* (whose adaptation capabilities are in Listing 25 of Appendix A) is yet another example of an *Adapter* that makes dynamic decisions. This *Adapter* embeds the *ffmpeg* transcoding tool. When certain parameters (such as the frame rate or bit rate) are not explicitly provided, the *ffmpeg* tool chooses default values, which usually depend on the transcoding operation that will be carried out.

2.2 Conversions vs. planning actions

Section 6 of Chapter 2 has surveyed the standard AI planners, states and actions. In the *Planner*, a *conversion* is a kind of action that changes the characteristics of a *Component* element. This term has been chosen following the MPEG-21 Part 7 nomenclature. A conversion represents the partial or complete state of the *Component* before and after its adaptation. Conversions have three main features:

- *Dynamic decisions*. Conversions have the option of performing dynamic decisions. One conversion corresponds to a group of related actions. Graphplan-like planners (described in Subsection 6.2 of Chapter 2) also incorporates the idea of grouping actions and produce a planning graph in which each state represents a set of related actions. However, in Graphplan-like planners, these sets of actions are partially instantiated states that will be fully instantiated before the planner terminates. In the case of our conversions, the parameters are partially instantiated and the *Planner* never fully instantiates these parameters because these decisions are postponed until the execution phase.
- *Bounded non-deterministic actions*. Conversions are bounded non-deterministic actions. They are *bounded* because the *Planner* limits the parameter values, but the parameter values could be multi-valued. They are *non-deterministic* because the existence of dynamic decisions makes the actions non-deterministic. A non-deterministic action is a specific type of action as described in Subsection 6.3 of Chapter 2.
- *Tolerating partial description*. Conversions incorporate semantics for absent properties as described in Section 3 of this chapter.

2.3 Conceptual view of the Planner

The inputs to the *Planner* are the *Component* (from the *Content DI*) to be adapted, the *ConversionCapabilities* of the available *Adapters*, and a UED (from the *Context DI*) describing the current usage environment. Conceptually, all these elements of the *Planner* are represented with a group of properties referred to as *conversion states*. The conversion states will be formalized in the next section. Fig. 14 shows this conceptual view of the *Planner* together with the description tools that represent each type of description element. In this conceptual model, the description elements that correspond to conversion states can be homogenized as follows. The instances of the *ConversionCapabilities* elements have both preconditions and postconditions. The instances of the *Component* description element have only postconditions. The instance of the UED tool has only preconditions. Hence, the *Component* and the UED description elements can be seen as *ConversionCapabilties* elements that only have postconditions and preconditions, respectively.

The term *sequence of conversions* (soc_i) will be used to refer to any sequence of conversion states cs_n, cs_{n-1}, ..., cs_1 that leads from the initial content cs_n (i.e., an instance of the *Component* to be adapted) to the adapted content cs_1 (i.e., the instance of the UED). Since the *Planner* is a backward planner, indices appear in reverse order from n to 1. As there may be several ways of adapting the input media resource to the usage environment, the *Planner* builds a *virtual tree of conversions*. In the virtual tree of conversions, the nodes correspond to the conversion states and the arrows correspond to changes in the conversion states. The term *set of sequences of conversions* (**SSOC**) will refer to all of these sequences of conversions, i.e., $\mathbf{SSOC} = \{soc_1, ..., soc_k\}$. For reason of convenience, the *Planner* creates several instances of the conversion state that represent the initial content (i.e., of the *Component* that is going to be adapted). For example, in Fig. 14 $soc_1 = \{cs_{a4}, cs_{a3}, cs_{a2}, cs_{a1}\}$, $soc_2 = \{cs_{b3}, cs_{b2}, cs_{b1}\}$, $soc_3 = \{cs_{c4}, cs_{c3}, cs_{c2}, cs_{c1}\}$ and $\mathbf{SSOC} = \{soc_1, soc_2, soc_3\}$. The output of the *Planner* is not the virtual tree of conversions, but the **SSOC**.

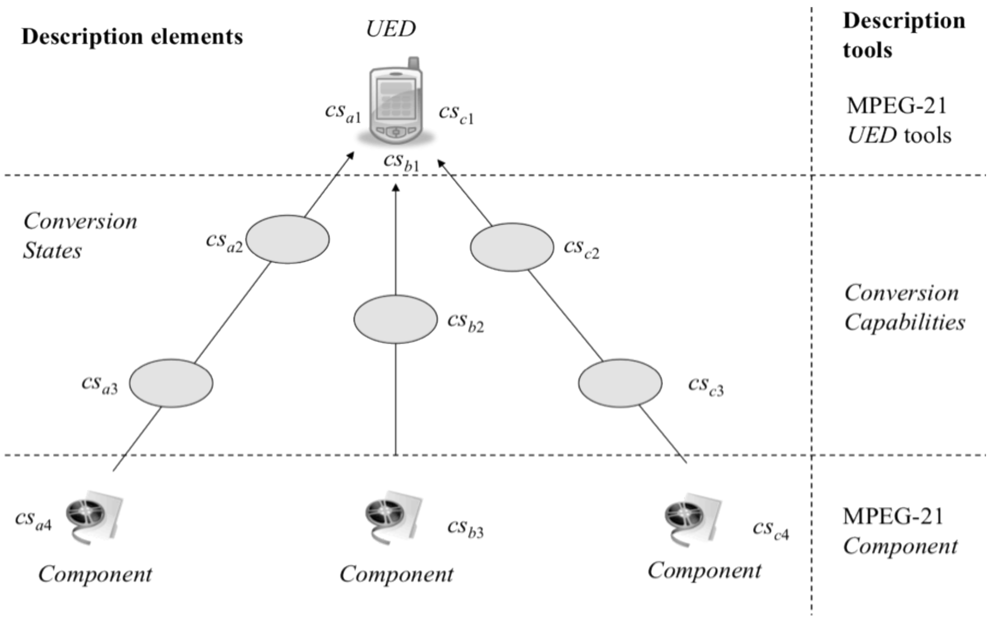


Fig. 14: Conceptual view of the *Planner*

2.4 Bounded non-deterministic conversions

In contrast to previous multi-step multimedia adaptation proposals [16][54][56], in this work the *Planner* is going to bind the output of the dynamic decisions to a subset of its potential outputs. The advantage of this design is that it permits postponing decisions from the decision phase to the execution phase. Since the *Adapters* make dynamic decisions during the execution phase, their output depends on these decisions and therefore the *Adapters* are non-deterministic from the point of view of the *Planner*, i.e., the *Planner* cannot completely anticipate the outcome of these *Adapters*, and therefore of the adaptation.

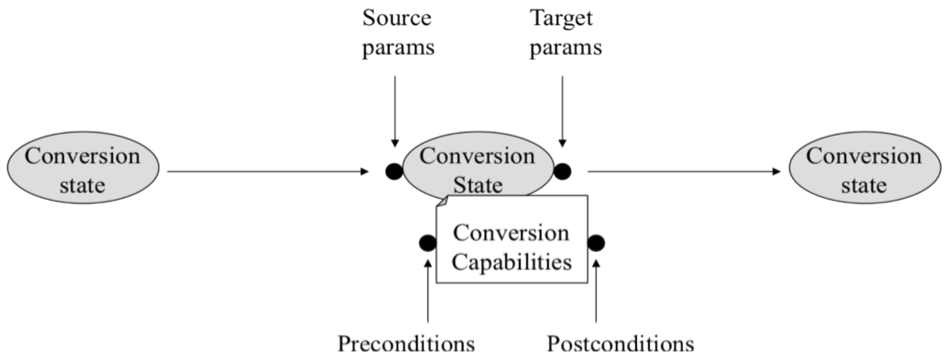


Fig. 15: Elements of a bounded non-deterministic conversion

According to their outcomes, conversions can be divided into three groups:

1. *Deterministic conversions*, in which the outcome is always bound to single value.
2. *Unbounded non-deterministic conversions*, in which the outcome is not always the same, and the planner cannot select the outcome.

3. *Bounded non-deterministic conversions*, in which the outcome can vary but the planner can select which outcomes are permitted (from the ones available in the postconditions).

In order to allow postponing decisions, this work uses bounded non-deterministic conversions that represent the steps in the *Planner*. Fig. 15 shows the elements that take part of these bounded non-deterministic conversions. The following subsections describe these elements and their relationship.

2.4.1 Property-based representation of the predicates

Preconditions, postconditions and invariants (described in Subsection 6.1 of Chapter 2) have been traditionally represented with first order logic predicates that have shown to be sufficiently capable of expressing many planning problems [88][89]. This research evaluates an alternative representation of predicates based on properties (i.e., 0-ary propositional predicates). The advantage of this alternative representation is that it facilitates the description of alternative options and ranges of values. The term *single-valued property* will be used to refer to a label (variable assignment) with only one value (e.g., *width*=320). Similarly, the term *multi-valued property* will refer to a label with multiple homogeneous values (e.g., *format*={*mpeg*-1,*mpeg*-2,*mpeg*-4} or *bitrate*=[16000..780000]).

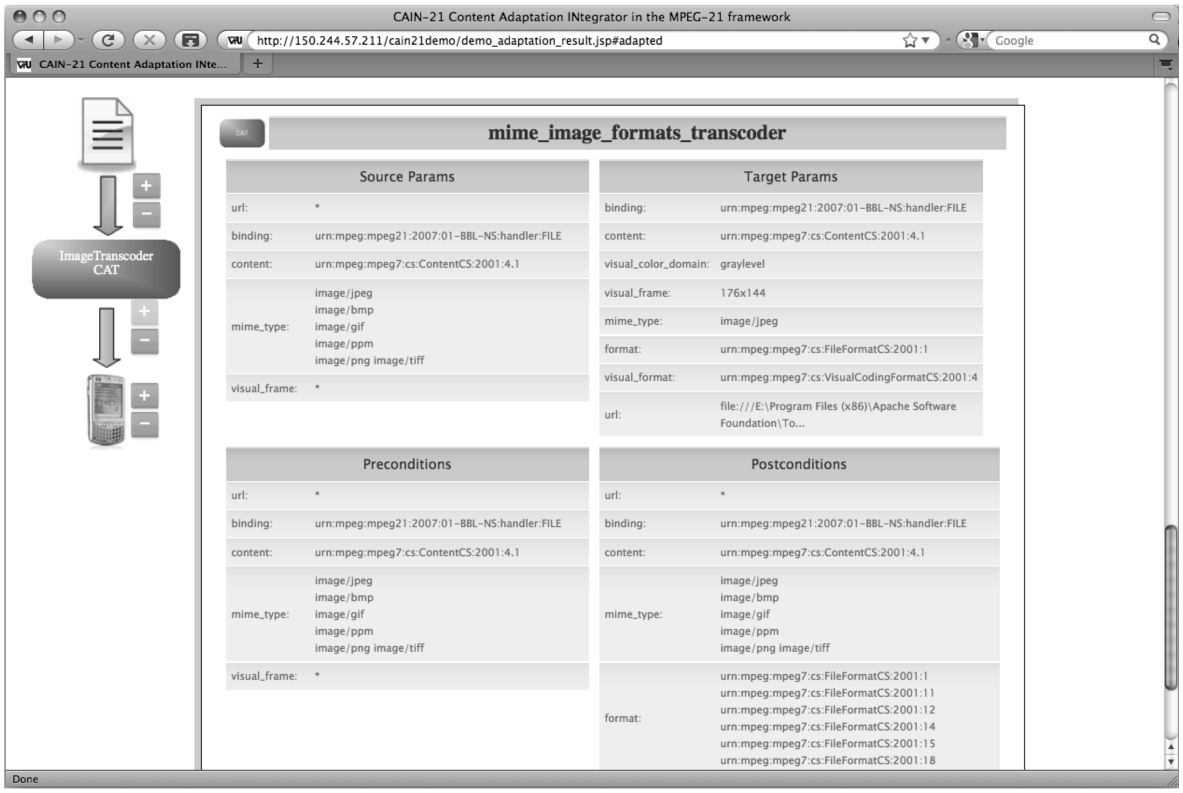
2.4.2 Conversion capabilities and conversion states

As described in Subsection 6.3 of Chapter 2, non-deterministic planning has addressed uncertainty in the output of the actions with belief states. This work proposes an alternative approach to address non-deterministic actions with conversion states.[16] Specifically, the term *conversion capabilities* cc_i[17] will refer to the range of properties that the *Adapter* accepts and produces. By extension, we can envision the first element of the sequence (the *Component*) as a conversion capabilities object that only has postconditions. Likewise, we can envision the last element of the sequence (the UED) as a conversion capabilities object that only has preconditions. The term *selected conversion state* (cs_i), or *conversion state* for short, will refer to the subset of properties that the *Planner* is considering for execution in a sequence of conversions. The term *realized conversion state*, shortened to *realized*(cs_i), will refer to the result of executing a non-deterministic conversion (i.e., set of single-valued property valuations). Thus, given any conversion module the following relation holds: $realized(cs_i) \subseteq cs_i \subseteq cc_i$.

In the *Planner,* only the properties in *realized*(cs_i) have to be single-valued. For example, when given a conversion capabilities element cc_i that accepts *format*={*mpeg*-1, *mpeg*-2, *divx*} and produces *format*={*mpeg*-1, *mpeg*-2, *mpeg*-4, *divx*}, the *Planner* may generate a conversion state cs_i accepting *format*={*mpeg*-1, *mpeg*-2} producing *format*={*mpeg*-1, *mpeg*-2, *mpeg*-4}. In this example, the values accepted by cs_i are a subset of the values accepted by cc_i. Similarly, the values produced by cs_i are a subset of the values produced by cc_i. After its execution, the conversion module may end up receiving *format*={*mpeg*-2} and producing *format*={*mpeg*-4}. The result of executing the conversion corresponds to *realized*(cs_i). Fig. 16 shows an example of such a conversion state in the CAIN-21 demo.

[16] As described in Subsection 3.3.5 of Chapter 2, MPEG-21 Part 7 proposes the term *conversion act* to refer to a conversion and its parameters. This work uses the term conversion state to stress that the conversion states are states in the AI planner.

[17] The term *conversion capabilities* (shortened to *cc*) makes reference to the range of properties accepted and produced by the conversion module whereas the term *ConversionCapabilities* makes reference to its description using MPEG-21 XML description elements.


CAIN-21 Content Adaptation INtegrator in the MPEG-21 framework
http://150.244.57.211/cain21demo/demo_adaptation_result.jsp#adapted
Google
CAIN-21 Content Adaptation INte...
ImageTranscoder CAT
CAT
Done

mime_image_formats_transcoder

Source Params	
url:	*
binding:	urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE
content:	urn:mpeg:mpeg7:cs:ContentCS:2001:4.1
mime_type:	image/jpeg image/bmp image/gif image/ppm image/png image/tiff
visual_frame:	*

Target Params	
binding:	urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE
content:	urn:mpeg:mpeg7:cs:ContentCS:2001:4.1
visual_color_domain:	graylevel
visual_frame:	176x144
mime_type:	image/jpeg
format:	urn:mpeg:mpeg7:cs:FileFormatCS:2001:1
visual_format:	urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:4
url:	file:///E:\Program Files (x86)\Apache Software Foundation\To...

Preconditions	
url:	*
binding:	urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE
content:	urn:mpeg:mpeg7:cs:ContentCS:2001:4.1
mime_type:	image/jpeg image/bmp image/gif image/ppm image/png image/tiff
visual_frame:	*

Postconditions	
url:	*
binding:	urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE
content:	urn:mpeg:mpeg7:cs:ContentCS:2001:4.1
mime_type:	image/jpeg image/bmp image/gif image/ppm image/png image/tiff
format:	urn:mpeg:mpeg7:cs:FileFormatCS:2001:1 urn:mpeg:mpeg7:cs:FileFormatCS:2001:11 urn:mpeg:mpeg7:cs:FileFormatCS:2001:12 urn:mpeg:mpeg7:cs:FileFormatCS:2001:14 urn:mpeg:mpeg7:cs:FileFormatCS:2001:15 urn:mpeg:mpeg7:cs:FileFormatCS:2001:18

Fig. 16: Conversion state in the CAIN-21 demo

2.4.3 Preconditions, postconditions, source and target parameters

Subsection 4.1 of Chapter 4 explained that each *ConversionCapabilities* element contains two elements: the *Preconditions* and the *Postconditions*. The *Preconditions* element describes (using multi-valued properties) the conditions that must be fulfilled before the conversion can be executed. The *Planner* must select one or more of its values. The *Postconditions* element describes (using multi-valued properties) the outputs that the conversion can produce. These description elements correspond to the conversion capabilities cc_i, preconditions object $pre(cc_i)$ and the postconditions object $post(cc_i)$. If a property of the conversion module is not described in the preconditions and in the postconditions, the property is invariant, i.e., its value is not altered in the conversion. For this reason, in the model for bounded non-deterministic conversions invariant properties are not directly represented.

The term *selected source parameters* $s\_params(cs_i)$ (or *source parameters* for short) will be used to refer to the finite subset of preconditions $s\_params(cs_i) \subseteq pre(cs_i)$ that the *Planner* selects for executing a conversion. Similarly, the term *selected target parameters* $t\_params(cs_i)$ (or *target parameters* for short) will be used to refer to the finite subset of postconditions $t\_params(cs_i) \subseteq post(cs_i)$ that might be obtained during a specific conversion execution. The selected source and target parameters are determined during the decision phase. If the target parameters were multi-valued, the *Planner* would be transferring decisions to the *Adapters*. In this case, they have the option to decide which output to produce from the available ones. In the same manner, the terms *realized source parameters* $s \in s\_params(cs_i)$ and *realized target parameters* $t \in t\_params(cs_i)$ (whose names, in contrast with the selected parameters, are not shortened in this document) will be used to refer to the single-valued properties that the *Adapter* receives and produces during the execution phase.

Fig. 15 shows the conversion state as an instance of the conversion capabilities: the source parameters are a subset of the preconditions and the target parameters are a subset of the postconditions. The source and target parameters in Fig. 15 will become selected parameters at the end of the decision phase and realized parameters at the end of the execution phase. Note that the conversion states are represented with only one object whose properties (in the case of selected conversion states) may be multi-valued.

The example in Fig. 16 shows a conversion state (source and target parameters) together with its conversion capabilities (preconditions and postconditions). The source parameters must fulfil the preconditions, and hence the source parameters must be a subset of the preconditions, i.e., $s\_params(cs_i) \subseteq pre(cs_i)$. Similarly, the target parameters must be a subset of the postconditions $t\_params(cs_i) \subseteq post(cs_i)$. Target parameters are always associated with the postconditions and not with the invariants, because invariant properties by definition cannot be changed and are not explicitly modelled. In Fig. 16, the conversion labelled as *mime_image_formats_transcoder* is defined in its postconditions as *mime_type*={*image/jpeg, image/bmp, image/gif, image/ppm, image/png, image/tiff*}, which means that the conversion module can produce all of these image formats. Subsequently, in the example, the *Planner* has decided that the output format must be *image/jpeg*, and therefore *mime_type*={*image/jpeg*} is in the selected target parameter. In general, during the decision phase the target parameters can be assigned several values. For instance, if the terminal accepts *Graphic Interchange Format* (GIF) and JPEG images, the target parameter would be *mime_type*={*image/gif, image/jpeg*}. In this case, the *Planner* is transferring the decision of what value to produce to the *Adapter*.

2.5 Bounded non-deterministic planning

In this work, multimedia conversions are modelled as bounded non-deterministic conversions where a conversion can be executed by providing a set of target multi-valued parameters. After that, the execution of each conversion has to choose which of these parameter values to produce (a subset of the set of the postconditions). An important difference between the bounded non-deterministic planner (the *Planner*) and other multimedia planners (such as the one in [54]) is that the bounded non-deterministic planner must determine, not only whether preconditions and postconditions match, but also the source and target parameters settings to execute the conversions. This bounded non-deterministic planner should not be confused with other techniques such as continuous planning [90] or planning under uncertainty [62]. These two latter techniques implement unbounded non-deterministic planners in which the actual outcome of an action is not known before its execution. Such situations can for example arise, when the execution of an action fails due to mechanical or an unforeseen change in the environment occurred since the planning phase. Therefore, this other group of techniques requires the existence of *contingency decisions*, that is, alternative decisions that are chosen depending on the outcome of executing an action (such as to repeat the action or to select an alternative action). Contingency decisions are needed in continuous planning and planning under uncertainty because if contingency decisions were not used, the execution of the plan would not always lead to a goal; the goal would be only reached when there was no "failure" in the sequence of actions. Moreover, these contingency decisions have the effect that no linearly ordered sequence of conversions exists. Bounded non-deterministic planners, however, can build linear sequence of actions that always succeed. The *Planner* uses conversions that – by definition – always produce one of the selected solutions. Therefore, the *Planner* can calculate all the feasible sequences of conversions before the conversions are executed.

3 Tolerating partial description

Multimedia adaptation systems, such as CAIN-21, can be used in complex real-world multimedia platforms, which might involve a relatively large number of multimedia properties. From the viewpoint of both the user that needs to adapt his/her multimedia content and of the *Adapter* implementer (i.e., the third-party that implements a pluggable *Adapter*), supplying accurate values for the entire set of properties may become tedious and error prone. For this reason, we decided to search for a *Planner* that can operate with a partial description. When only a partial description of the conversion capabilities is available, the *Planner* assumes default meanings for the unprovided properties. Specifically, the *Content DI* (i.e., the *Component* to be adapted) and *Context DI* (i.e., the UED) may arrive at the adaptation engine without giving all their property values. In addition, the *Adapter* implementer is not forced to provide a detailed description of all the properties of the *AdapterCapabilities*. The following subsections describe the semantics designed to tolerate partial description.

3.1 Semantics of the properties of the conversion capabilities

This work proposes the following semantics for existing properties in the preconditions and postconditions of the conversion capabilities (i.e., all the description elements considered in Subsection 2.3). These semantics apply in extended form also to the corresponding properties in the conversion states:

- *Required preconditions*. If a property appears in the preconditions of a conversion capability cc_i, this indicates that the conversion requires this property in the corresponding source parameter of the conversion state cs_i. Specifically, such a source parameter (represented as a property) must be a subset of the corresponding precondition property values.
- *Produced postconditions*. If a property appears in the postconditions of a conversion capability cc_i, this indicates that the corresponding conversion state cs_i produces such a property. In this case, the conversion state may either create the property (represented as a target parameter) if it does not exist in the source parameters, or modify it if it exists in the source parameters.

3.2 Incompleteness semantics

In addition to providing semantics to existing properties, *incompleteness semantics* have been introduced to tolerate partial description. These semantics define how to deal with absent or partially defined properties in the conversion capabilities and corresponding conversion states. Specifically:

- *Admitted properties*. If a property does not appear in the preconditions of a conversion capability cc_i, this situation must be interpreted as meaning that, for this property, *every value is acceptable, including the situation where it is not provided at all*.
- *Wildcard properties*. If a property is marked with a wildcard in the preconditions of a conversion capability cc_i, this configuration must be interpreted as meaning that, for this property, *every value is acceptable, but it has to be provided*.
- *Preserved properties*. If a property appears neither in the preconditions nor in the postconditions of a conversion capability cc_i, this situation must be interpreted as meaning that the conversion state cs_i preserves the value of such a property. This means that the *target parameters forward the property values of the corresponding source parameters without changes*.

Note that the incompleteness semantic of the above-mentioned preserved properties forces the *Planner* to assume that whenever a property does not appear in the conversion capabilities, this property is not modified in the conversion. Therefore, the *Adapter* that implements such a conversion must refrain from modifying this property. This design incorporates a risky assumption, as the *Adapter* implementers are assumed to be careful and precise in their design. The other option would have been to force the *Adapter* implementers to label all the invariant properties that the conversion module does not modify, which would become tedious for the *Adapter* implementers. If the incompleteness semantics had not introduced this preserved properties into the *Planner*, the algorithm that searches for the plan would have had to systematically discard all the conversion modules that were not completely labelled.

3.3 Ignored properties and accumulated effects

An *ignored property* happens when it appears in the preconditions of the conversion capabilities, but does not appear in the postconditions (this situation is inconsistent with the rules of the above incompleteness semantics). The term "ignored" must not be interpreted in the sense that the execution of the conversion necessarily "loses" the property, but in the sense that the conversion capabilities do not provide information about what is going to happen with this property. An example of this is a conversion capability that defines the maximum frame rate that a conversion module accepts, but does not define the maximum frame rate that it can produce. This situation may occur simply because the media produced by the conversion does not have a frame rate at all (e.g., it produces an image as a summary of a video clip), or because it does not specify the output frame rate (although the conversion produces a video).

CAIN-21 does not permit the *Adapters* to ignore properties, and the *Planner* uses the notion of *accumulated effects* to detect ignored properties. Given a step in the sequence of conversion represented with a conversion state cs_i, the *accumulated effects* represent the set of properties that will be modified or created from this step to the end of the sequence. This set corresponds to the union of the properties in the source and target parameters along the path from cs_i to cs_1. During each step in the backwards search, this design of the *Planner* forces the accumulated effects to always increase or remain. Otherwise, ignored properties would have been detected. If there are ignored properties, the *Planner* terminates its execution and reports them to the user.

4 Algorithms

This section describes in detail the most important algorithms in the *Planner*. The *Planner* is implemented in a recursive manner, and its algorithms are described starting from the main control algorithm and ending with the low level algorithms.

4.1 Main control structure

Subsection 2.3 of this chapter and Fig. 14 showed the conversion states as instances of the corresponding description elements. The main control structure of the *Planner* carries out a backwards search that finds all the sequences of conversion states capable of adapting the *initial conversion state* (instance of the *Component*) to the *goal conversion state* (instance of the UED).

Subsection 2.3 of this chapter described how to represent those sequences of conversions soc_i as a set of sequences of conversions **SSOC**=$\{soc_1, soc_2, \ldots, soc_k\}$. Alg. 1 is the top-level control structure of the *Planner* that computes these sequences. It starts by extracting the properties of the goal conversion state from the UED. After that, *getSetOfSequenceOfConversions*() (explained in Subsection 4.2 of this chapter) determines the sequences of conversions that lead to the goal conversion state. As the goal conversion state is not part of the sequences of conversions that *getSetOfSequenceOfConversions*() produces, Alg. 1 adds the goal conversion state to the end of each sequence.

```
Inputs:  problem   // The Component to be adapted, the AdapterCapabilities and the UED
Outputs: SSOC      // Set of sequences of conversions that the Planner has found.
Vars:    goal_cs   // A conversion state with the properties of the UED
         empty_soc // Starts backwards search with an empty sequence of conversion states
SSOC planner(problem)
  goal_cs = problem.getUEDConversionState();
  SSOC = getSetOfSequenceOfConversions(problem,goal_cs,[]);
  for (each soc in SSOC)
      soc = soc + {goal_cs} // Add goal_cs to the end of soc
  return SSOC;
```

Alg. 1: Main control structure of the *Planner* algorithm

All the *sequence of conversion states* must have at least two conversion states: the initial conversion state and the goal conversion state. Thus, if the initial conversion state fulfils the goal conversion state without further changes, the corresponding *sequence of conversion states* will only include these conversion states. Conversely, if there is no sequence of conversions that can adapt the *Component* to the UED, Alg. 1 will return an empty **SSOC**. In this case, CAIN-21 reports this circumstance to the user using plain English explanation messages (e.g., "There is no sequence of *Adapters* capable of converting DivX files to MPEG-4"). This message helps the adaptation engine administrator to configure the *Adapters* installed in the adaptation engine.

4.2 Computing the set of sequence of conversions

Alg. 2 describes the *getSetOfSequenceofConversions*() function. This function is a recursive process that traverses the virtual tree of conversions. The algorithm receives in the *subgoal* parameter either the goal conversion state (i.e., instance of the UED) or any *subgoal conversion state* (i.e., instance of either a *ConversionCapabilities* element or the *Component* element). The algorithm produces a set of sequences of conversions **SSOC**=$\{soc_1, soc_2, \ldots, soc_k\}$ that corresponds to the different paths. In order to prune the search, the algorithm also receives a list of *visited conversion states*. Alg. 2 has two parts:

1. The first loop determines which conversion states can precede the current goal conversion state. The term *prospective* conversion states refers to the conversion states (instances of the conversion capabilities available in the system) that match (according to Alg. 6) with the current goal conversion state. The term *feasible conversion states* refers to those prospective conversion states that contribute to the adaptation. They are explained in Subsection 4.3 of this chapter.
2. The second loop recursively composes the set of sequences of conversions that go from each feasible conversion to the current goal. The visited conversion states are used to avoid infinite loops through the same group of states.

Fig. 17 shows an example of this virtual tree of conversions. This example demonstrates the changes in the *source* and target parameters through the different paths in the virtual tree of conversions.

```
Inputs:  problem     // The Component to be adapted, the AdapterCapabilities and the UED
         subgoal     // The goal or subgoal in each recursive step
         visited_conversion_states // List of conversion state objects that have
```

```
                                    // already been evaluated
Outputs: SSOC        // Set of sequences of conversions that the Planner has found.
Vars:    prospective // The set of conversion states (instance of either a
                     // ConversionCapabilities element or the Component element)
                     // that matches the current subgoal
         feasible    // Subset of prospective conversion states that Alg. 4 has selected
         sub_ssoc    // The SSOC that reach the current subgoal
SSOC getSetOfSequencesOfConversions(problem, subgoal, visited_conversion_states)
  prospective = getProspectiveConversionStates(problem, subgoal);
  feasible = {};
  for (each cs in prospective)
      if (isFeasibleConversion(cs, subgoal, visited_conversion_states))
          feasible.add(cs);
  SSOC = {};
  for (each cs in feasible)
      if (cs isa Component)
          soc = {cs};
          SSOC.add(soc);
      else
          visited_conversion_states.add(cs);
          sub_ssoc = getSetOfSequencesOfConversions(problem, cs,
                                                    visited_conversion_states);
          visited_conversion_states.remove(cs);
          for (SequenceOfConversions soc : sub_ssoc)
              soc.addTailStep(cs);
              SSOC.add(soc);
  return SSOC;
```

Alg. 2: Set of sequences of conversions

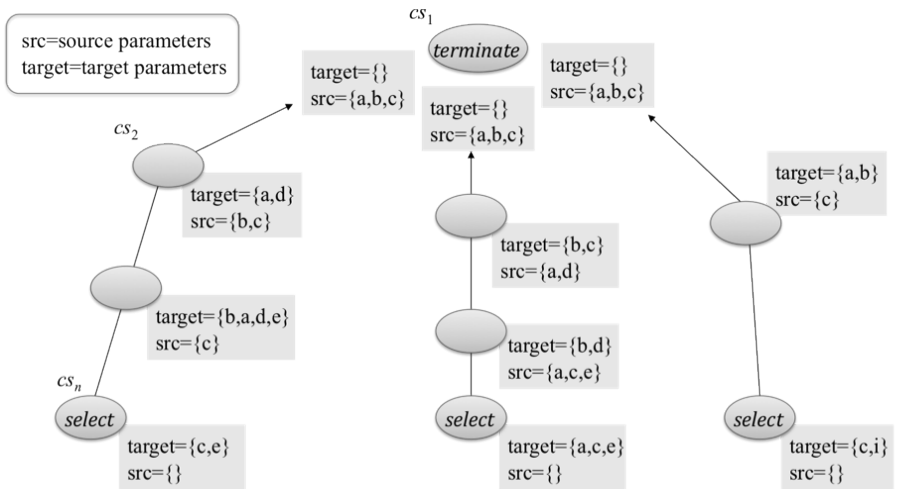


Fig. 17: Example of a virtual tree of conversions

4.3 Prospective and feasible conversion states

The *prospective conversion states* are obtained from both the *Component* to be adapted and the *ConversionCapabilities* of the *Adapters* currently installed in the system. In Alg. 3, the first part determines whether the conversion capabilities that correspond to the *Component* to be adapted matches (according to Alg. 6) the current goal conversion state. The second part of Alg. 3 obtains the set of conversion capabilities that match the current goal.

```
Inputs: problem  // The Component to be adapted, the AdapterCapabilities and the UED
        subgoal  // The goal of subgoal in each recursive step
Outpus: cs_set   // Set of prospective conversion states.
```

```
Vars:   child_cs // Conversion state that matches the current goal
SSOC getProspectiveConversionStates(problem, subgoal)
  cs_set = {};
  child_cs = match(subgoal, problem.getComponentConversionCaps());
  if (child_cs ≠ null)
      cs_set.add(child_cs);
  for (each cc in problem.getAdaptersConversionCaps())
      child_cs = match(subgoal, cc);
      if (child_cs ≠ null)
          cs_set.add(child_cs);
  return cs_set;
```

Alg. 3: Prospective conversion states

The feasible conversion states is a subset of the prospective conversion states that contribute to the adaptation. Alg. 4 shows the three conditions that make a prospective conversion state become a feasible conversion state:

1. It must not be previously visited.
2. It must contribute to the progress of the adaptation according to Alg. 5.
3. In the case of the initial conversion state, all its preconditions have to be satisfied, i.e., there are no preconditions of the UED that have not been produced during any step of the sequence.

```
Inputs: cs      // Prospective conversion state to be determined
                // as feasible conversion state
        subgoal // The goal of subgoal in each recursive step
        visited_conversion_states // List of conversion state objects that
                                  // have already been evaluated
Output: If the prospective conversion state is feasible or not
boolean isFeasibleConversion (cs, subgoal, visited_conversion_states)
  if (visited(cs, visited_conversion_states))
      return false;
  if (!contribute(cs, subgoal))
      return false;
  if (cs isa Component AND cs.SatisfiesAllPreconditions())
      return false;
  return true;
```

Alg. 4: Feasible conversion states

4.4 Conversion states that contribute to the adaptation

In each step within the backward search process through the virtual tree of sequences of conversions of Fig. 17, the *Planner* must determine which conversion states contribute to the adaptation process. The criterion to determine their contribution is as follows:

- If the intersection between the property list of the target parameters of the cs_{i+1} conversion state and the property list of the target parameters of the cs_i goal conversion state is a set with one or more empty property values, then this configuration means that cs_{i+1} contributes (with the empty properties in the intersection) to the progress of the adaptation. In this case, the cs_{i+1} conversion state is maintained.
- Otherwise, it means that the same outcome can be reached with just cs_i, and so the cs_{i+1} conversion state is discarded.

Consider an example in which the target parameters of cs_i are *visual_format*={*mpeg-1*, *mpeg-2*} and *audio_format*={*mp2*, *mp3*} and the target parameters of cs_{i+1} are *visual_format*={*mpeg-4*} and *audio_format*={*mp2*, *mp3*}. In this case, the intersection of the *visual_format* values is an empty set. This means that the *mpeg-4* visual format can be adapted by adding cs_{i+1} to the virtual tree of conversions. Note that this discarding condition never occurs when the cs_i conversion state is the goal conversion state. This happens because *getUEDConversionState*() always returns a conversion state with an empty set of target parameters, and in this case the source parameters correspond to the properties of the UED (see Fig. 17).

Alg. 5 determines whether or not a conversion state contributes to the adaptation. It starts by determining the intersection between the target parameters of cs_i and cs_{i+1}. Subsequently, a property with empty values denotes the existence of states that cannot be reached without cs_{i+1}. This means that cs_{i+1} contributes to the adaptation.

```
Inputs: cs      // Prospective conversion state.
                // The algorithm determines if it contributes to the adaptation
        subgoal // The goal of subgoal in each recursive step
Outputs: Whether the cs conversion state contributes to the adaptation
         (that produces the subgoal conversion state) or not
boolean contribute(cs, subgoal)
 intersected_props_values = cs.getTargetParams() ) subgoal.getTargetParams();
  for (each p intersected_props_values)
      if (p.isEmpty())
          return true;
  return false;
```

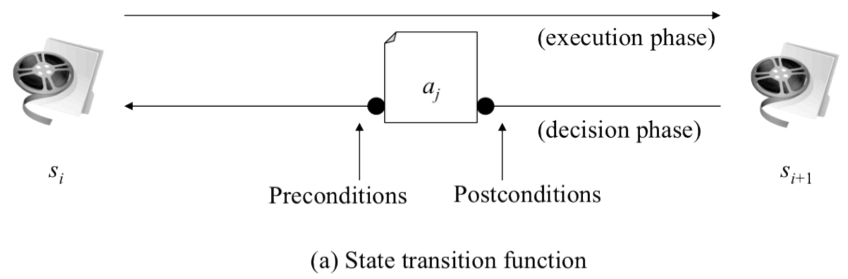
Alg. 5: Conversion states that contribute to the adaptation

4.5 The matching process

Section 6.1 of Chapter 2 surveyed how classical planning algorithms represent changes in the state of the system using a state transition function. Fig. 18 (a) shows the typical elements involved in the state transition function. The state transition function $\gamma(s_i,a_j)$ evaluates the preconditions of an action $a_i \in A$ and determines if the action can be applied to the input state $s_i \in S$ in order to produce the output state $s_{i+1} \in S$ so that $s_{i+1} = \gamma(s_i,a_j)$. Accordingly, three constrained features define classical planning algorithms:

- *States are fully observable*. The system has complete knowledge of the world, and therefore observes the outcomes in a single state, i.e., the current unique state of the system.
- *Actions are deterministic*. Actions have single-valued states, i.e., if applicable during the s_i state, each action a_i leads to a single new state s_{i+1}, so that $|\gamma(s_i, a_j)| \leq 1$.
- *Actions are always unbounded*, i.e., they have no *source* or target parameters. Therefore, the result of executing an action is always the same, and so it makes no sense to use parameters to select single-valued properties.

This research proposes that relaxing these traditional assumptions allows for modelling and solving a wider range of problems. Thus, this proposal for a bounded non-deterministic planner replaces the state transition function $\gamma(s_i,a_j)$ with a *matching process* $\gamma(cs_i,cc_{i+1})$. Fig. 18 (b) shows the elements of this matching process. The matching process is a function that receives as input the *i*-th conversion state cs_i together with the (i+1)-th conversion capability cc_{i+1} and produces as output the (i+1)-th conversion state cs_{i+1}. In other words, the matching process identifies the cs_{i+1} conversion state whose target parameters are acceptable by the cs_i conversion state. In this case, the conversion capabilities cc_{i+1} matches the conversion state cs_i. The rationale behind this change is to introduce multi-valued properties representing postponed decisions for the bounded non-deterministic conversions.



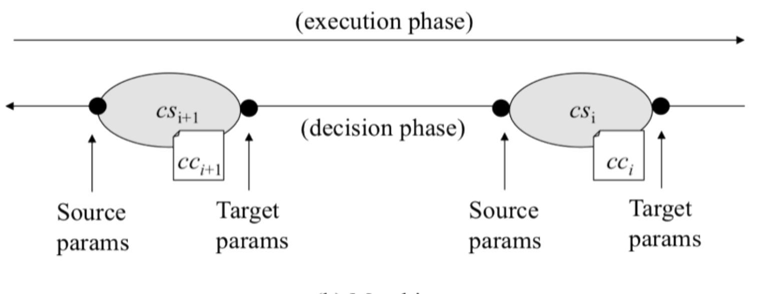


Fig. 18: Elements of the state transition function and of the matching process

Alg. 3 uses the matching process to obtain the prospective conversion states. The arrows in Fig. 18 show that the decision phase progresses from the target to the source, whereas the execution phase progresses from the source to the target. In this work, indices are always assigned to the states according to the order in which the decision phase evolves.

The state transition function used in classical planning algorithms creates an explicit intertwined sequence of states and actions as shown in Fig. 18 (a). On the other hand, as Fig. 18 (b) shows, in the proposed model the implicit state of the *Component* that is being adapted can be removed from the *sequence of conversion states* without losing information. The conversion state implicitly represents the state of such a *Component* through its source and target parameters. Specifically, the source parameters represent the *Component* before executing the conversion and the target parameters represent the *Component* after executing the conversion. Therefore, the new conversion state object cs_{i+1} includes its corresponding conversion capabilities cc_{i+1} (represented as preconditions and postconditions) together with the *source* and target parameters that the matching process has selected for the conversion state. The details of the matching process algorithm are given in Alg. 6.

```
Inputs:  cs0 // Conversion to be reached (subgoal)
         cc1 // The ConversionCapabilities or Component to be considered
Outputs: cs1 // Conversion state (an instance of cc1) that matches cs0
             // or null if there is no matching
Vars:    key // The key of a property. Properties contains one key and a set of values
         p   // Used to refer to each postcondition of cs1
         q   // Used to refer to each source parameter of cs0
ConversionState match(cs0, cc1)
  cs1 = new ConversionState(cc1);
  // Step 1: Gather source parameters that come from cs0
  for (each q in cs0.getSourceParams())
      key = q.getKey();
      p = cc1.getPostcondition(key);
      if (p ≠ null)
          r = p ∩ q;
          if (r.isEmpty())
              // Fails since cc1 is not capable of producing a property requested by cs0
```

```
            return null;
        cs1.addTargetParam(r);
    else
        cs1.addSourceParam(q);
// Add the preconditions of cc1 to the source parameters of cs1
// if they have not been taken from cs0 in step 1
for (each p in cc1.getPreconditions())
    key = p.getKey();
    if (cs1.getSourceParam(key) = null)
        cs1.addSourceParam(p);
// Add the postconditions of cs1 to the target params of cs1
// if they have not been selected during step 1
for (each p in cc1.getPostconditions())
    key = p.getKey();
    if (cs1.getTargetParam(key) = null)
        cs1.addTargetParam(p);
return cs1;
```

Alg. 6: Matching process

5 Theoretical analysis of the Planner

The following subsections conduct a theoretical analysis of the *Planner's* algorithms and of the plan that these algorithms produce.

5.1 Finiteness of the algorithm

This subsection proves that Alg. 1 is *finite*, i.e., it always terminates. Firstly, note that the algorithm progresses backwards from the goal conversion state to the initial conversion so that progressively the source parameters are removed and the target parameters added. Once the algorithm terminates, the goal conversion state contains only source parameters and the initial conversion state contains only target parameters (see Fig. 17). That being said, note that:

1. The number of properties in the goal conversion state is always the same, regardless of the sequence of conversions that reaches the goal conversion state. For example, in Fig. 17 all the sequences lead to the source parameters $\{a,b,c\}$. However, since the properties of the goal conversion state could be multi-valued, the values of these properties that each sequence of conversions produces may vary. This result is consistent with the capability of the UED to accept alternative properties. For instance, suppose that in Fig. 17 the UED preconditions accept a=\{*mpeg*-2, *mpeg*-4\}, which means that the a property with one of these values must be produced by every sequence of conversions. However, some of these conversions might produce a=\{*mpeg*-2\}, other sequences might produce a=\{*mpeg*-4\}, and even another group of conversions might produce both a=\{*mpeg*-2, *mpeg*-4\}.
2. It is not guaranteed either that the source parameters will be monotonically removed, or that the target parameters will be monotonically added. However, Subsection 3.3 of this chapter explains that, as the *Planner* does not accept conversion capabilities with ignored properties, the accumulated effects (the union of the source and target parameters) of the cs_{i+1} are a superset of the source parameters of cs_i. For instance, in Fig. 17, the goal conversion state has the source parameters $src=\{a,b,c\}$ and the cs_2 conversion state has the accumulated effects $src=\{a,d\} \cup target=\{b,c\}=\{a,b,c,d\}$, which is a superset of the source parameters of the goal conversion state. *Theorem 1* makes use of the accumulated effects to prove that the *Planner* algorithm always terminates.

Theorem 1: Alg. 1 is finite.

Proof: Given any goal conversion state cs_i, representing either the goal conversion state or any subgoal conversion state, it can be inferred that:

1. As conversion capabilities with ignored properties are not allowed, the set of accumulated effects remains stable or increases in size in each step from cs_i to cs_{i+1}. Therefore, in each conversion step going backwards from cs_1 (the goal conversion state) to cs_n (the initial conversion state), the number of accumulated effects never decreases (see Fig. 17). This idea is similar to the one proposed in [89] where "delete lists" are removed from a *Stanford Research Institute Problem Solver* (STRIPS)-like planner in order to guarantee decidability.
2. As the number of *ConversionCapabilities* elements installed in CAIN-21 is finite, Alg. 3 always expands a finite number of prospective conversion states (as defined in Subsection 4.3 of this chapter). In practice, cs_1 always has a finite number of source parameters (number of properties of the UED), which must be reached by its subsequent conversion states during each step of the sequence of conversions from cs_i to cs_{i+1}. Hence, only two situations may occur: a) the number of accumulated effects increases from cs_i to cs_{i+1}, or b) the number of accumulated effects from cs_i to cs_{i+1} remains the same. However, in the second case Alg. 2 cannot use conversion states stored in *visited_conversion_states* again. Therefore, Alg. 2 always terminates (and thus Alg. 1). In the worst-case scenario, Alg. 2 will terminate when all the conversion states are stored in *visited_conversion_states*.

5.2 Finiteness and completeness of the plan

This subsection proves that the plan that Alg. 1 produces is *finite* (always terminates) and *complete* (the virtual tree of conversions covers all the feasible conversion states).

Theorem 2: The plan is finite.
Proof: As, according to *Theorem 1*, Alg. 3 always expands a finite number of conversion states, and because the *visited_conversion_states* guarantees no cycles in the virtual tree of conversions, the plan reaches all the feasible conversion states in a finite number of steps.

Theorem 3: The plan is complete.
Proof: *Theorem 1* guarantees that Alg. 1 expands all the conversion states that reach the goal conversion state and *Theorem 2* guarantees that the plan is finite. Hence, the plan will reach all the feasible conversion states, and therefore is complete.

6 Multi-step adaptation engines comparison

This section provides a comparative review of four multi-step multimedia adaptation decision systems: VRT [54], koMMa [16], MAGG [56] and CAIN-21. These engines have been introduced in Section 5 of Chapter 2. The comparison is based on the following six aspects:

1. The *representation* of the multimedia content, adaptation capabilities and terminal capabilities.
2. The *matching process* that the system uses to compare the multimedia content with the adaptation and terminal capabilities.
3. Whether *partial description* is supported.
4. Whether *alternatives* in the terminal consumption capabilities are considered.
5. The *decision points* in which the system makes decisions.
6. Whether the *finiteness and completeness* of the plan are considered.

Table 5 summarizes this comparison and shows the publication year of these systems.

	VRT	koMMa	MAGG	CAIN-21
Year	2005	Apr. 2006	Dec. 2006	2010
Representation	MPEG-21, OWL-S, UAProf	MPEG-7/21, OWL-S	Not indicated	MPEG-7/21, *AdapterCapabilities*
Matching process	OWL-S matchmaching	OWL-S matchmaching	Tuples = matchmaching + Q	Alg. 6
Partial description	No	No	No	Yes
Alternatives in the terminal	No	No	No	Yes
Decision points	Static	Static	Static	Selection + Static + Dynamic
Finiteness and completeness	No	No	Completeness	Finiteness + completeness

Table 5: Summary of comparison for the multi-step adaptation decision systems

6.1 Representation

MAGG provides a description of its own multi-step decision algorithm; however, the authors do not indicate the multimedia representation schemes used.

VRT, koMMa and CAIN-21 use MPEG-7/21 to describe the multimedia content. To represent the terminal capabilities VRT uses UAProf, whereas koMMa and CAIN-21 use MPEG-21. VRT and koMMa only use the terminal capabilities to define the constraints of the adapted content. CAIN-21, however, also takes into account the network capabilities (e.g., maximum bit rate) to define these constraints.

While VRT and koMMa perform directly the matching between the multimedia content document and terminal capabilities document, CAIN-21 transforms these documents into multimedia properties before performing the matching. The use of multimedia properties presents several advantages with respect to using standard XML documents:

- All the decisions-related information can efficiently be held in memory.
- Changes in the underlying XML documents do not imply changes in the source code of the whole adaptation engine. Section 6 of Chapter 4 explains that it only implies changes in the *Properties DI* document.
- The multimedia properties directly represent the information that the *Planner* requires.
- The information is represented homogeneously.
- Alternatives are easily represented by means of multi-valued properties.

In reference to the description of the adaptation capabilities, VRT and koMMa study the use of OWL-S to describe the IOPE of the service. The preconditions and effects denote external conditions required by the service and the effects resulting from its execution. In contrast, CAIN-21 uses *Preconditions* and *Postconditions* elements to represent these conditions, i.e., CAIN-21 does not take into account the invariant effects. The justification of this change is motivated in Subsection 2.4.3 of this chapter.

VRT and koMMa use explicit ontologies to represent the adaptation actions whereas MAGG and CAIN-21 use implicit ontologies. The use of implicit ontologies may give rise to ambiguities and do not permit to infer knowledge from existing knowledge as AI reasoning can make (see Subsection 4.2 of Chapter 2). However, in the multi-step multimedia adaptation domain, if the implicit ontology is used carefully, ambiguities can be avoided. In addition, the simplicity of AI planning is enough to obtain finite, complete and efficient adaptation plans. To avoid ambiguities, CAIN-21 creates a defined semantic for each property using the *Properties DI* (see Section 6 of Chapter 4). Section 5 of this chapter has proven the finiteness and completeness of the adaptation plans. Section 2 of Chapter 7 demonstrates that the multimedia properties allow for the development of efficient AI planning.

6.2 Matching process

All the decision systems analyzed perform some form of *semantic matching* [91] between three elements: the structure of the content, the available adaptation actions and the target platform's capabilities. This subsection discusses the differences among these matching processes.

The composition of Semantic Web services is examined in detail in [92]. In order to compose conversions, VRT and koMMa use AI planning with an explicit ontology and reasoning to check possible relations. Specifically, VRT and koMMa check either equivalence or subsumption relations between classes describing the source document and the target platform. MAGG provides its own definition of adaptation services by means of tuples and matches the properties of these tuples. CAIN-21 uses multi-valued property matching to check these relations. Multi-valued properties allow the representation of non-deterministic actions in which parts of the decision are transferred from the *Planner* to the *Adapters*.

OWL-S is an ontology that builds on top of OWL to describe Web Services. The classical *matchmaking* process [93] was proposed to find appropriate providers and requesters. This process specifies the transformation produced by the service in terms of IOPEs. VRT and koMMa bring these *matchmaking* ideas to the multimedia domain. MAGG also uses this matchmaking idea by adding the Q attribute to represent quality attributes (in terms of cost and time). CAIN-21 matches multimedia properties, instead of documents, to identify these relations.

The matchmaking process is quite effective because it allows the extraction of new knowledge by means of reasoning and has a wider range of application domains than simple property matching. The main advantage of the multi-valued property matching is that it allows postponing decisions. Another advantage of property matching is that it is faster than making inference from an explicit representation of the ontology. Section 2 of Chapter 7 demonstrates the efficiency of property matching.

In reference to the topology of the search, MAGG produces a graph of conversions, whereas CAIN-21 produces a virtual tree of conversions. The authors of VRT and koMMa do not specify the topology of the search.

6.3 Partial description

Traditional decision systems assume that the capabilities of the actions are completely known. VRT, koMMa and MAGG do not mention partial description and therefore we assume that their adaptation actions are completely described. Furthermore, the authors of koMMa state in [94]: "These descriptions have to be constructed by hand and it is the responsibility of the engineer that the preconditions and effects of the program execution are defined in a valid and complete manner. *Valid* means that only function symbols are used that are allowed in the MPEG standards, which can be checked using standard XML-validation; *complete* means that all of the required preconditions and effects are listed, whereby no automatic check is possible in general. Note however, that in the error case where the action description uses undefined symbols, the planner will produce no plans that include such an action but try to use other transformation plug-ins to reach the goal state."

CAIN-21 deals with absent properties, which are useful in the practical application domain of multimedia adaptation. Furthermore, absent properties, in conjunction with multi-valued properties, allow the *Planner* to navigate through a *set of conversion states*. The work in [63] demonstrated that these sets reduce the number of states that must be evaluated and therefore speed up the decision process.

Planning under uncertainty uses belief states (explained in Subsection 6.3 of Chapter 2) that associate a probability distribution over the state space in order to represent multiple states. In contrast, this work uses selected and realized states (instead of belief states) to represent the manifold states that the non-deterministic actions can produce. Absent properties must not be confused with un-instantiated actions and goal attributes within a classical planner; the former correspond to information that is never given, while the latter are unbound attributes that must be bound after producing a plan. Absent properties must also not be confused with flexible planning [64]. Absent properties correspond to a lack of information; flexible planning introduces soft constraints in the classical planning domain definition.

6.4 Alternatives

Although general AI research has considered alternative goals (see, for instance [86]), VRT, koMMa and MAGG have not taken into account alternatives in the terminal capabilities (e.g., the terminal accepts several media formats).

Conversely, CAIN-21 has developed a model for non-deterministic conversion states that allows representing alternative constraints in the terminal. Additionally, this model allows for representing alternatives in the dynamic decisions of the *Adapters*. To this end, this work have replaced the notion of action with the notion of conversion in such a manner that different parameters of the conversion lead to different actions. Multi-valued parameters make it possible to gather related actions in a single conversion state. The source parameters (that may be multi-valued representing the selected inputs) represent the input to the conversion. The target parameters (that also may be multi-valued allowing the *Adapter* to make dynamic decisions) control the output of the conversion. Hence, one conversion can be comparable to a set of actions in a Graphplan-like planner [63].

6.5 Decision points

VRT, koMMa and MAGG make all the decisions during the decision phase (static decisions according to the taxonomy in Section 1 of this chapter). Conversely, the *Planner* of CAIN-21 transfers parts of the decision to the *Adapters* (dynamic decisions) and in this way, the decision and the execution phases are intertwined. Another advantage of this division is that the metadata-based general decisions are performed in the *Planner* and particular decisions that depend on the media resource are offloaded to the *Adapters*.

In contrast to continuous planning (introduced in Subsection 2.5 of this chapter), the conversions of the *Planner* are bounded non-deterministic actions. As a result, the *Planner* does not perform further decisions that depend on the result of the dynamic decisions transferred to the *Adapter*. That is, the *Planner* computes all the sequences of conversions before the *Executer* can start executing the *Adapters*.

6.6 Finiteness and completeness

VRT and koMMa are forward search planners and hence only search for one feasible sequence of actions. Conversely, normally backward search planners such as MAGG or CAIN-21 identify all the sequences of conversions. MAGG introduces the idea of completeness by identifying all the feasible sequence of conversions. Subsequently, MAGG searches for what the Berhe et al. refer to as the optimal adaptation paths. During the study of these optimal adaptation paths, they take into account the execution and transmission time and cost of the service.

In general, an AI plan can be infinite [62] (i.e., have cycles). Proving that a plan always terminates is not a trivial task, which have only been accomplished by CAIN-21. Section 5 of this chapter has conducted a theoretical analysis of the *Planner* and has demonstrated that the plans that it produces are finite (always terminate). In general, finiteness does not hold when a planning algorithm permits the removal of effects. For this reason, CAIN-21 incorporates a planner that never removes effects and has formally proven that in this way the plan is always finite.

CAIN-21 is the only decision system that formally proves the completeness of the plans that it produces (see Section 5 of this chapter). The virtual CAIN-21's tree of conversions corresponds to all the feasible sequences of conversions capable of producing adapted content. This feature allows further decisions in order to pick the sequence that optimizes some criterion (such as execution time or resulting spatial resolution). In addition, in contrast to a neoclassical planner, the bounded non-deterministic planner must find the source and target parameters that must be supplied to the non-deterministic conversions.

7 Limitations and difficulties

This section discusses the limitations and difficulties encountered with the proposed *Planner*.

This research has identified three main limitations:

1. *Dynamic matching*. The matching process does not allow a definition of a dynamic matching between conversion states. In *static matching*, the values of the properties are declared in the *Preconditions* and *Postconditions* elements. For instance, the property *format*={*mp2,mp3*} in the *Postconditions* of one conversion and the property *format*={*mp3,aac*} in the *Preconditions* of another conversion can be statically matched. In *dynamic matching*, the relationship between preconditions and postconditions cannot be expressed with a declarative approach. For instance, the output bit rate of a video transcoder depends, in general, on the input frame size. Although it is theoretically possible to express a property as a function of other properties, it is not always easy or possible to find a function that provides the exact value of a property in terms of other properties (e.g., the exact output bit rate might not be represented as a function of the input frame width and height properties). To address this limitation, the declarative approach presented in this work uses a range of values to relate the output properties (the postconditions) with the input properties. If the *Adapter* implementers want to create this declarative relationship, they must create several conversion capabilities with different

"profiles" for the input and output property ranges. For instance, one *ConversionCapabilities* element might describe frame sizes between 44x36 and 177x144 that produce a bit rate between 2000 bits/s and 8000 bits/s. Another *ConversionCapabilities* element might describe the frame sizes between 177x144 and 704x576 that produce a bit rate between 8000 bits/s and 48000 bits/s. In Subsection 4.2.1 of Chapter 7, the *Image2VideoAdapter* uses this approach to define several profiles for the image sizes (namely, the *big_image_2_video*, *medium_image_2_video* and *small_image_2_video*).

2. *Prohibit properties in the terminal*. The semantics of the MPEG-21 UED tools do not provide a mechanism to prohibit properties in the description of the media that can be consumed. According to these semantics, if the MPEG-21 terminal does not declare the audio capabilities, it does not mean that the media cannot include audio; rather it means that the *Component* will be accepted regardless of the existence of such an audio stream. Thus, the terminal may end up receiving properties that it does not understand. To avoid such problems, the terminals must not use properties that have not been declared in its UED.
3. *Forward constraints in the original content*. The input *Content DI* (where the *Component* to be adapted is located) cannot impose constraints on involving properties if the *Adapters* or the usage environment does not consider these properties. Specifically, in a sequence of conversion the properties of the *Component* have to fulfil the constraints in the preconditions of the first conversion. Otherwise, the *Planner* will not evaluate this conversion. However, the properties of the *Component* that do not appear in the preconditions of the conversion are always allowed and ignored. In fact, these properties implicitly become postconditions of the conversion state. The cause of this effect is the preserved properties' incompleteness semantic (see Subsection 3.2 of this chapter). Again, to avoid this difficulty, the terminal must not use properties that have not been declared in its UED.

In addition, during the implementation of the *Planner*, the following difficulties were identified:

1. *Semantic gaps*. There are semantic gaps among the MPEG-7 description of the *Component* and the MPEG-21 UED. The solution to this gap has been discussed in Subsection 7.1 of Chapter 4. These difficulties have been hidden behind the *getUEDConversionState*() function in Alg. 1.
2. *The Planner consider that the terminal properties are mandatory constraints*. The sequence of conversions must produce all the properties of the terminal. More precisely, the terminal properties are a conjunction of preconditions where all the preconditions must be "produced" at a certain step of the sequence of conversions. This is a limitation because sometimes it is desirable to describe optional properties. For instance, if a terminal accepts visual and audio streams, the *Planner* would consider a video composed of just a visual stream (i.e., without an audio stream) non-consumable by this terminal. To address this limitation, Section 7 of Chapter 4 describes how extending the standard *mpeg21:TerminalType* description tool with *optional properties*. In contrast with ordinary properties (also referred to as mandatory properties), the matching process can ignore optional properties. Optional properties are only allowed in the *Context DI*. The *Content DI* and *AdapterCapabilities* cannot declare optional properties. However, the *AdapterCapabilities* can declare properties according to the admitted and wildcard properties' incompleteness semantics (described in Subsection 3.2 of this chapter).
3. *Verbose description of the adaptation capabilities*. It is frequently necessary to divide one *AdapterCapabilities* element into several *ConversionCapabilities* elements. This decomposition produces a verbose description. For example, in the *ConversionCapabilities* description scheme, it is cumbersome to describe a property that preserves its value, but the property

must exist and take one value from a set of selected values. In this case, the *ConversionCapabilities* element must be divided into several *ConversionCapabilities* elements, so that the preconditions of each *ConversionCapabilities* element accept only one value and produce the same value. Another verbose description occurs in order to represent a disjunction of preconditions or postconditions. In the *Preconditions* and *Postconditions* description scheme, the properties describe a conjunctive condition. To express a disjunction of properties, these properties must be listed in different *ConversionCapabilities* elements. The reverse situation occurs with an individual multi-valued property, with values that describe a disjunction in which only one value must be selected. Still another condition that is difficult to express succinctly is combinations of property values that are not accepted. An example would be a conversion that accepts JPEG and PNG images so that JPEG images are accepted in both colour and greyscale, but PNG images are only accepted in greyscale. In this case, the capabilities must be split into two separate *ConversionCapabilities* elements: one stating that JPEG images are accepted in both colour and greyscale and another in which PNG images are accepted only in greyscale. If the *AdapterCapabilities* have several of these restrictions, its list of *ConversionCapabilities* elements will quickly become long and unwieldy. This difficulty could be handled through a *Graphic User Interface* (GUI) managing these descriptions. In addition, the semantics that tolerate partial description also help.

4. *Precise compliance with adpatation semantics*. The correct operation of the system depends on its precise compliance with the semantics of the *AdapterCapabilities* representing the preconditions and postconditions. This means that the whole system's usefulness depends on the *Adapter* authors correct description of the *ConversionCapabilities*, i.e., according to the precise semantics of the parameters. The matching process does not make inference, and therefore it does not work effectively if there were different policies with respect to the meaning of the properties or different labels to represent the same value. For instance, consider MPEG-4 videos that have different levels and profiles. One conversion might be capable of processing all the MPEG-4 video file levels without ever specifying the levels in its capabilities description. Another conversion might only be capable of processing one level and describing that it can accept only this level. In this case, the two conversions might not work together effectively on MPEG-4 media, i.e., the output of one conversion cannot be used as input of the other conversion. To answer this problem, the *Adapter* implementer must pay special attention to describe the preconditions and postconditions of the *ConversionCapabilities* elements according to their semantics. Making use of a set of standardized classification schemes is very useful in this case. MPEG-7 Part 5 classification schemes such as the *ContentCS*, *FileFormatCS*, *VisualCodingFormatCS* and *AudioCodingFormatCS* can be used in this case (see Appendix B of the MPEG-7 Part 5 standard [95]).

8 Conclusions

Life can only be understood backwards; but it must be lived forwards.
Soren Kierkegaard
Danish philosopher (1813 - 1855)

This chapter has dealt with the applicability of AI planning methods for the computation of multi-step multimedia adaptations. To adapt multimedia, some extensions to standard AI planning methods have been proposed. Traditionally, multi-step adaptation has been implemented with an AI planner that makes all the decisions before beginning the adaptation. Taking into account that there are decisions that can only be made, or are easy to made, during the execution phase (i.e., when the media resource is available), this chapter has proposed the inclusion of these decisions into the adaptation process. To accomplish this goal, this research work has modelled multimedia conversions as bounded non-deterministic conversions and has developed a bounded non-deterministic planning algorithm. The *Planner* allows for dealing with decision-making problems in which the conversions to perform can be controlled (i.e., are bound), even though under some circumstances they may produce different outcomes (i.e., are non-deterministic). These outcomes may also only be partially observable by the *Planner*.

The proposed planning algorithm is capable of computing all sequences of conversions that adapt an MPEG-21 *Component* to the constraints of the UED. In addition, mechanisms that deal with partial observability tolerating partial description have been proposed. The theoretical analysis has proven the finiteness of the *Planner* and the finiteness and completeness of the plan produced. The *Planner* has been compared with other multi-step multimedia adaptation decision systems. Finally, the most important findings, limitations and difficulties pertaining to multi-step multimedia adaptation have been discussed. The next chapter address the problem of deciding the best sequence of conversions when the *Planner* concludes that several sequences can be used to produce adapted content.

Chapter 6:

Best adaptation decision-making

Synopsis:

After the set of feasible adaptations has been selected, this chapter aims to decide which adaptation is the best. With this end in mind, the chapter brings into play the preferences of the different types of users. Different preference-based decision methods are analyzed. Then, the chapter describes how to integrate and coordinate all these methods as well as its execution in the decision points that the previous chapter has identified.

1 Introduction

Section 2 of Chapter 2 introduced the idea of user-centric adaptation. Modern GUIs facilitate the gathering of the preferences of different types of users, progressively and interactively. In addition to end-users, the content creators and content providers can also provide their preferences (e.g., the authors of [51] label the web pages with transcoding hints that steer the adaptation process). For this reason, this document uses the term *user preferences* to refer to the preferences of any agent in the multimedia system (including the end-user, the content creator, the content provider and the adaptation engine itself). In contrast, the term *user's preferences* refers to the preferences of a particular user.

Currently there is no consensus on the best method to systematically translate all these preferences into automatic adaptation decisions. In some cases, the content creator or content providers want to give to their end-users the capability of providing preferences in order to indicate how to perform these decisions manually. This *manual decision* process facilitates the user's the supervision of the technical aspects of the adaptation process. In order to let the user steer the decision, the multimedia system has to provide a mechanism for the interactive collection of the user's preferences. For instance, the system in [96] collects the user's preferences to summarize scalable multimedia documents. The system does not perform decisions, but reacts to the user's inputs by showing or hiding different parts of the scalable documents.

In other cases, the content creator or content provider chooses to remove their users from this responsibility by means of endowing their systems with an *automatic decision* process. This automatic decision process is responsible for choosing the outcomes that maximize the user's experience according to the defined decision criteria. Frequently, these systems use some kind of objective or subjective quality metrics to assign utility to the automatic decisions. Practical adaptation systems usually combine manual and automatic decisions in the more effective manner. Sometimes transferring this decision to the user depends on whether the user is expected to be capable of making this decision. In other cases, the system provides default automatic decisions and the user can customize these decisions.

The previous chapter developed a *Planner* capable of automatically identifying the sequences of conversions that produce multimedia content fulfilling the technical constraints of the current terminal and network. As the terminal may be capable of consuming different multimedia formats with different attributes, we found that the result of this automatic *decision process* is a *selection*, that is, it usually comprised of a set of feasible sequences of conversions (see Subsection 2.4 of Chapter 5). Subsection 2.3 in Chapter 5 introduced the symbol **SSOC** to refer to this set. This current chapter develops a *decision process* that enables both manual and automatic decisions of the best adaptation in this set.

The result of Chapter 5 is that different sequences of conversions lead to different adaptations, i.e., different *outcomes*, all of them fulfilling the technical constraints that the selected terminal and network impose. After enforcing the *hard constraints*, we started to realize that if more than one adaptation remain, it would be useful to decide which adaptations to perform. This decision can be taken according to the user's preferences on media format and adaptation schedule. In AI and knowledge-based systems [97][98], *soft constraints* allow for modelling a wide variety of constraints that should be fulfilled as much as possible. This research uses soft constraints to formally model user preferences.

Therefore, the objectives of this chapter are:

1. To build a preferences model that assists in the decision of the outcomes that best suit the preferences.
2. To model preferences in such a way that if the user moves to a different terminal or network (i.e., the hard constraints change) the gathered user's preferences should nevertheless continue to be valid and applicable.
3. To incorporate this model into the MPEG-21 representation schema.
4. To design systematic and automatic decision-making methods that employ preferences to decide both the best sequence of conversions and the best parameter values. The preference-based decision methods must benefit from the preference model.
5. To incorporate preference-based manual and automatic decision into CAIN-21 and to combine them effectively.

The structure of this chapter is as follows. Section 2 defines a preference model for multimedia adaptation. Section 3 studies the static decisions that can be made during the decision phase. Section 4 studies the dynamic decisions that can be made during the execution phase. Section 5 makes a comparison between the preference-based decision methods developed in this chapter and the preference-based decision methods in the literature. Finally, Section 6 concludes this chapter.

2 Preference model for multimedia adaptation

The purpose of this section is to define a model that integrates different multimedia adaptation preferences. Fig. 19 shows the differences between our multimedia adaptation preference model and the preference model of the MPEG-21 framework. The next sections use this preference model to identify the best methods for automatically choosing the best adaptation.

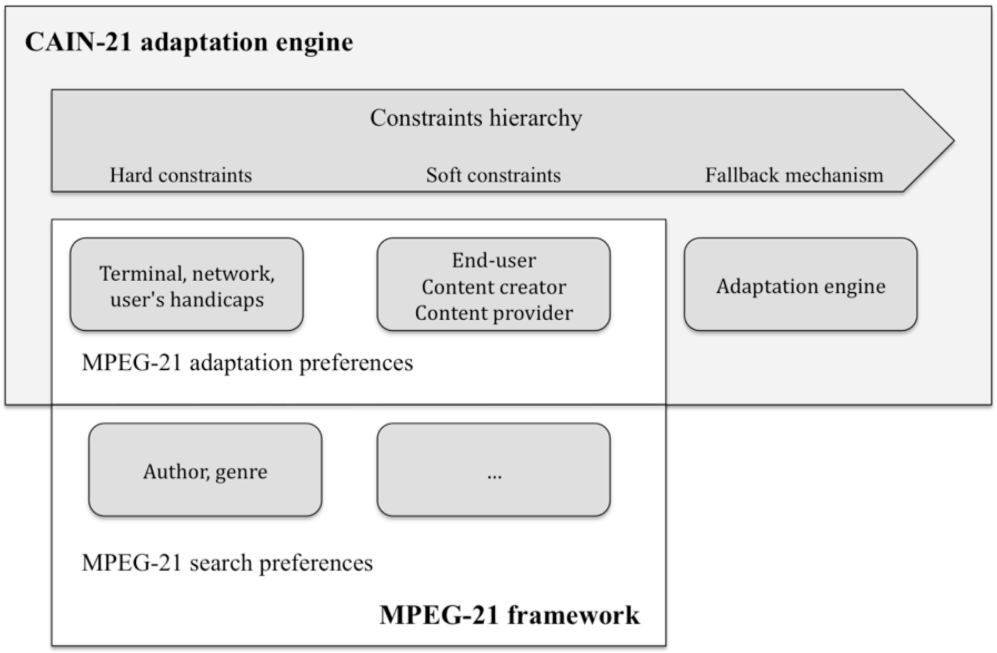

CAIN-21 adaptation engine
Constraints hierarchy
Hard constraints
Soft constraints
Fallback mechanism
Terminal, network, user's handicaps
End-user Content creator Content provider
Adaptation engine
MPEG-21 adaptation preferences
Author, genre
...
MPEG-21 search preferences
MPEG-21 framework

Fig. 19: MPEG-21 vs. CAIN-21 preference model

2.1 Search, adaptation and delivery engines

Subsection 7.1 of Chapter 2 introduced the notion of preference variables and domain. In order to define the variables that comprise the preference model for the multimedia adaptation domain, it is important to define the responsibilities for each subsystem in the entire multimedia system. For the purposes of this discussion, we use the term *engine* to refer to these subsystems. More specifically, this subsection creates a taxonomy that clarifies the differences between the search, adaptation and delivery engines.

A multimedia system is intended to manage a range of media and multimedia elements. Such multimedia elements are video, pictures, audio files, etc, as well as structure level compositions like HTML pages or MPEG-21 DIs. One or more *media repositories* store all of these elements. The *search engine* is the subsystem that assists the user in navigating through the media repositories. A search engine assists the users in finding the multimedia elements that they are interested in by providing querying, filtering and browsing (which in turn may include adaptation) functionalities (typically using a GUI). In the MPEG-21 framework, the search engine finds the DIs that the user is interested in. Currently CAIN-21 does not include these search capabilities.

Once a DI has been found, it might be necessary to modify the content of this DI to meet the usage environment constraints. The subsystem responsible for performing such tasks is an *adaptation engine* (such as CAIN-21). Note that the term DI is not used only to refer to DIs stored in the media repository, but also dynamically generated DIs. For instance, a browsing menu generated by the search engine could also need to be adapted. In the MPEG-21 framework, this browsing element is also a DI.

Finally, the *delivery engine* is the subsystem in charge of transporting the media and multimedia elements (DIs in the MPEG-21 framework) through the network. Although the main purpose of the CAIN-21 is adaptation, CAIN-21 also includes delivery capabilities through the *HTTPVideoServerAdapter*. It is important to notice that the DI description can be transported before or after the adaptation of its resources. Indeed, it is also possible to perform the adaptation of the DI and its resources through several nodes (for more details on this distributed approach see, for instance, [58]).

2.2 Search and adaptation preferences

Subsection 3.3.1 of Chapter 2 described how the *mpeg7:UserPreferences DS* lets the users specifing their preferences (likes and dislikes) for certain types of content. MPEG-21 Part 7 connects the *mpeg21:User* to the *mpeg7:UserPreferences DS* through the *mpeg21:UsagePreferencesType* description tool. Some of these preferences seem to be more suitable during searches (such as the movie title or the actors names provided in the *mpeg7:CreationPreferences DS*) and other preferences seem to fit better in the adaptation (such as the media modality or media format provided in the *mpeg7:SourcePreferences DS*). Taking into account the fact that one multimedia element (*Content DI* in the proposed system) may comprise several variations of the same media content, and considering also the purpose of the preferences, we propose to make a distinction between these two kinds of preferences:

1. *Search preferences*, which are used during the user's search for *Content DIs* in the media repository. Their main objective is to help the users in deciding the *Content DIs* that best suit their query.

2. *Adaptation preferences*, which are used during the adaptation of a *Content DI* (that previously might have been selected by the search engine) to the preferences in the usage environment (mainly terminal related).

Of course, some of these preferences may be used in both engines. For instance, the spoken language may be used during the search assuming that the user has provided this preference in the query. They also can be used during the adaptation, if the DI holds several language variations of the same content.

2.3 Preference-based static and dynamic decisions

Section 1 of Chapter 5 introduced the notion of decision points. Then, Chapter 5 described how to perform the selection of the set of feasible adaptations (formally represented as a **SSOC**). This selection process takes into account only the hard constraints. In contrast, the decision process (both static and dynamic decisions) is heavily *preference-based* and thus this work uses soft constraints to model these preferences. This subsection lays out how the static and dynamic decisions use the user preferences to decide which is the best adaptation.

In particular, the *static decisions* are implemented in the decision phase and they rely only on metadata. Section 3 of Chapter 7 will demonstrate how CAIN-21 uses metadata to make these static decisions. Specifically, CAIN-21 uses two groups of metadata; the media format preferences and the adaptation schedule preferences.

Of course, it can be argued that, in the practical sense, media users have more interest in media content itself than in its format. Nevertheless, media format preferences are very useful in some cases. For instance, the next section introduces the adaptation engine preferences. In this case, the media format preferences may become useful for the multimedia system in which the adaptation engine has been deployed. Another case in which the media format preferences are useful is the fallback mechanism described in Subsection 2.5 of this chapter.

In reference to *dynamic decisions*, these decisions can be made using both the metadata and the media resource. In CAIN-21, the dynamic decisions are transferred to the *Adapters*. Section 4 of Chapter 7 will describe adaptations involving dynamic decisions. Subsection 5.4 of this chapter provides examples of different kinds of semantic adaptation decisions.

Another difference between precefence-based static and dynamic decisions is that static decisions are rather *semantic-agnostic*, i.e., they do not consider the meaning of the preferences variables and values, but their location in the **SSOC** or preference graph. On the other hand, dynamic decisions are *semantic-aware*, i.e., they need to understand the semantic of the preferences in order to perform the best adaptation.

2.4 Constraint hierarchy

Aside from expressing the user's preferences, it would also be useful for the adaptation engine itself (and the multimedia system in which the adaptation engine has been deployed) to be capable of expressing its preferences. For instance, the adaptation engine could express the preference for executing the conversions with lower execution cost, the sequence with the lowest number of conversions or executing lossless conversions rather than the lossy ones.

This research has found that, in contrast to the user's preferences, the MPEG-21 UED tools do not consider the preferences of the adaptation engine (e.g., preference for minimizing the number of conversions, preference for reducing the content degradation, preference for online or fast conversions). From this perspective, the lack of description tools to represent the preferences of the adaptation engine (in addition to the end-user's preferences) is a shortcoming of the UED preferences model.

This subsection creates a constraint hierarchy and investigates the incorporation of the adaptation engine preferences into the preference model. In particular, the preference model of CAIN-21 prioritizes the end-user's preferences over the adaptation engine preferences. Specifically, a *constraint hierarchy* with three levels has been defined:

1. *Hard constraints*. This group include the terminal (e.g., decoding capabilities), network's hard constraints (i.e., maximum bandwidth) and user's preferences hard constraints (e.g., user's handicaps).
2. *Network, end-user, content creator and content provider soft constraints*. Actually, in CAIN-21 the network's minimum bandwidth is a soft constraint. If it is not possible to fulfil the bandwidth constraints, the constraints are only fulfilled as much as possible (e.g., this happens with the network profile labelled as id="modem"). An example of end-user's preferences soft constraints is the fast adaptation vs. high quality adaptation results.
3. *Adaptation engine's preferences soft constraints* (e.g., preferred audio or video format).

With this hierarchy, the adaptation engine can provide preference values for the end-user's preferences. For example, the adaptation engine can recommend H.264/AVC instead of MPEG-2, since this format provides a better compression rate. In this case, the adaptation engine is providing default values for the end-user's preferences that are automatically applied when the end-user does not provide them.

2.5 Fallback preferences

Analyses of human factors indicate that users cannot be expected to have the patience (or sometimes the ability) to provide detailed preference relations or utility functions [68]. In addition, user preferences tend to be incomplete and change in different contexts. In many practical domains, the set of preferences is very large and forcing the end-users to provide all their preferences becomes unreasonable. If the preferences elicitation cost is taken into account, it becomes necessary to consider decision-making with partial preferences information. Danan [101] has proposed that, in general, it is not a good idea to make the user choose all the preferences. In fact, it is recommended to let the user choose only those preferences whose meaning he/she *knows*. Making the user fill in all the preferences gives rise to a set of preferences that do not describe the real wishes of the user. For this purpose, Danan et al. have proposed two types of preferences: (1) *choice preferences* that describe the user's selections and (2) *knowledge preferences* that describe the user's knowledge about his/her real wishes.

The preference model of CAIN-21 enables the management of incomplete preferences. To this end, CAIN-21 provides *fallback preferences*. The fallback preferences are a complete set of default values for the user preferences. The default preference values are automatically applied when the user does not provide them. The fallback preferences are implemented in the third level (i.e., adaptation engine preferences) of the constraint hierarchy described in Subsection 2.4 of this chapter.

2.6 Preferences elicitation with and without feedbacks

Subsection 7.4 of Chapter 2 surveyed the user preferences elicitation techniques. According to the method of interaction (handshaking) between the user's interface and the user, these elicitation modes can be divided into:

1. *Cumulative elicitation mode*, where all the user preferences are collected before the decision phase starts.
2. *Interactive elicitation mode*, where the end-user is prompted with queries depending on the progression of the decision and execution phase.

Faltings et al. [102] have described why, in general, the normal user is not aware of all the preferences until he/she realizes that they are to his/her disliking. Faltings et al. [102] also coined the term *agile preferences* to refer to the incremental construction and revision of a preferences model by the end-user. The preference model developed in this work brings the notion of agile preferences to multimedia adaptation.

This subsection studies the construction of a multimedia preference model using these two elicitation modes. The pros and cons of these elicitation modes vary depending on where we have to perform static or dynamic decisions.

In reference to the *preference-based static decisions* model, they are performed during the decision phase and just after the selection process. Hence:

- If we use the cumulative elicitation mode, all the preferences are gathered before initiating the adaptation process. Therefore, the decision and execution phase are launched in sequence without waiting for the user's feedback.
- If we use the interactive elicitation mode, the end-user is asked for his/her preferences progressively and on demand (i.e., for those preferences that become necessary to make a decision). An interactive user interface can be used in this case.

The interactive elicitation mode in turn can be implemented in two sub-modes:

1. *Decide and ask*. In this interactive mode, right after making a decision phase, the tentative outcome of the adaptation (and perhaps also the tentative sequence of conversions) is presented to the user. If the user agrees with the proposed outcome, the execution phase is launched. Alternatively, instead of offering only one outcome (adaptation) to the user, the system can present to the user a list of feasible outcomes (adaptations) in order of their utility, and according to the existing preferences. The main advantage of this decide and ask mode is that usually the decision phase is much faster than the execution phase (especially when large videos need to be adapted). The main drawback is that the user has to decide according to a description of the outcome, and not according to a visualization of the outcome. This drawback is particularly important when considering untrained end-users. However, note that within the MPEG-21 framework, the term *User* is not only related to human users, but other subsystems of the multimedia systems are also considered users of the adaptation engine. In cases where the *User* of the adaptation engine is a multimedia system (and not a human), such a decide and ask model is especially valuable. This is based on the fact that this mode bypasses the extra cost related to executing the resource adaptation several times.

2. *Execute and ask*. In this interactive mode, the decision and execution are completed, and after that, the adapted multimedia document is presented to the end-user for evaluation. If the end-user disagrees with the result, the end-user can ask again for the execution of the whole adaptation process with a different set of user's preferences. This type of interactive mode is suitable for human users. For example, the adaptation engine could perform a video adaptation to the current display size preferences of the user, but if the user is not happy with the dimensions of the video, he/she can change his/her preferences accordingly. In this way, it is possible to progressively implement a preferences feedback model that is consistent with the notion of agile preferences.

In reference to the *preference-based dynamic decisions* model, these decisions are made inside the *Adapters* and during the execution phase. Thus, it is impossible to know the outcome of the adaptation before executing it. As the decision is taken during the execution phase, the decide and ask mode cannot be used with dynamic decisions. Hence, only two elicitation modes can be used with dynamic decisions:

1. *Cumulative elicitation mode*. In this case, all the preferences are collected before initiating the adaptation and the user is not interactively prompted. For instance, the transcoding hints or the objective quality measures of scalable visual layers can be used in this mode.
2. *Execute and ask*. In this case, after the execution phase the end-user is prompted for his/her approval. For instance, the end-user can adjust the ROIs in the image.

2.7 Independence of the hard constraints

Section 1 of this chapter laid out the objective of allowing the user to move to another terminal or network, and even in this case the user's preferences must continue to be valid and applicable. This means that the user's preferences do not have to depend on the current usage terminal and network. To permit the same soft constraints to be used along with different hard constraints, the preference model has been intentionally developed to decouple the soft from the hard constraints.

For instance, let us suppose that the user has provided a preference ranking for the visual and audio formats of a set-top box terminal capable of rendering both visual and audio. The demonstration in Subsection 3.2 of Chapter 7 can be applied here. In this demonstration, the user has provided a preference ranking for the format of two different preference variables. Now, let us suppose that we change the target terminal (but not the user's profile) to a mobile device that only supports audio. Clearly, in this case the set of theoretical and feasible outcomes for the mobile device would be different (see Subsection 7.5.2 of Chapter 2). However, the preference ranking that the user supplied for the audio format continues to be the same (although the visual ranking has become irrelevant). Obviously, if the user has different preferences for different devices, it would be necessary to create a different user's preferences profile for each device.

In MPEG-21, the terminal, network and user's preferences are stored in different description elements. This structure facilitates the decoupling of hard and soft constraints. Subsection 3.2.2 of Chapter 7 will describe the representation schema for the preferences of CAIN-21 that are consistent with this additional advantage.

3 Static decisions

During the planning phase, the *Planner* first selects the set of feasible sequences of conversions (**SSOC**), and then it makes a series of static decisions. These static decisions aim to find which is both the best sequence of conversions and the best set of parameters (i.e., outcome of the sequence). The following subsections describe how to make both decisions.

3.1 Best-sequence-based decisions

This subsection addresses the problem of deciding which sequence of conversions is the best by taking into account the number of steps, the execution cost, whether all the conversions of the sequence can be executed online and the content degradation.

The *execution cost* is a number representing the time that the adaptation takes. It is general knowledge that video transcoding often has high execution costs. Standard benchmark tools have been developed to measure the processing performance of a computing device. The Dhrystone benchmark open source tool [103] determines the performance of a processor in terms of dhrystones per second. Goularte et al. [104] developed an exhaustive model for measuring the execution cost of video transcoding. This work aims to develop a model for the execution cost that is independent of both the hardware and the size of the resource. To this end, this model calculates the execution cost by multiplying the time needed to transcode one second of video by the dhrystones of the processor, on which the experiments were carried out.

If all the conversions of the sequence *can be executed online*, this means that the *OnA* mode can be used to speed up the user perception of the execution cost. This feature is appropriate for long resources such as videos. If any of the conversions does not support online adaptation, the *OdA* mode have to be used and the adapted media would not be delivered to the user until the whole resource is adapted. In our system, the decision on whether to use the sequence with the lower execution cost or the sequence that suport online adaptation depends on the user preferences.

The *content degradation* is a number ranging from 0 to 1, which represents the loss of information that the multimedia content suffers during its adaptation. Specifically, 1 stands for eliminating all of the content items and 0 stands for a lossless adaptation. The way in which this number is calculated depends on the multimedia content and format. It is up to the *Adaptor*'s implementer to provide this number. The following sections provide and justify some examples that assign this number to the conversion capabilities of different *Adaptors*.

To determine the execution cost and content degradation of the conversions, we have incorporated to CAIN-21 descriptions for both the conversions and the user preferences. In particular, different *ConversionCapabilities* elements represent multimedia conversions with different execution costs and content degradation. For instance, one *ConversionCapabilities* element might describe a conversion module that efficiently adapts a media stream to video without the audio component. In this example, another *ConversionCapabilities* element might describe a slower conversion module, which is able to adapt the media stream without dropping the audio component. In the case of this example, the first conversion module would be labelled with a lower value in the *ExecutionCost* description element than in the second conversion module. On the other hand, in the first conversion module the *ContentDegradation* might be, for instance, 0.5 as one of the two media components is eliminated during the adaptation. However, the second conversion module would be labelled with *ContentDegradation* equals to 0 (assuming that there is no content degradation during this adaptation).

In order to determine which adaptation is the best, it is necessary to know the user's preferences, i.e., whether the user prefers a fast adaptation or a lower content degradation. For this reason CAIN-21 provides the schedule preferences sketched in Table 6. There are default values for all these preferences. The content provider could redefine the preferences. In most cases, however, the end-user is not interested and does not provide these schedule preferences. Note that the content degradation is stored in the *ConversionCapabilities* description elements using a number from 0 to 1 whereas the user's preference for content degradation is stored in the *pref_content_degradation* property using a preference relationship. The *pref_online* preference indicates that the adaptation engine prefers to execute conversions, which could be executed online. The *pref_min_conversions* preference indicates that the adaptation engine prefers to minimize the number of conversion steps. The content provider can customize the importance of each one of these preferences by assigning different weights to them.

Subsection 3.1 of Chapter 7 will provide adaptation tests that make explicit the trade-offs between the execution cost and the content degradation. These tests allow the adaptation platform to choose the best sequence using the user's adaptation schedule preferences in Table 6.

pref_content_degradation
The user's assessment of the utility of the content degradation
Type: range Value: lossless ≻ lossy
pref_online
The user's assessment of the utility of the online generation of content
Type: range Value: online ≻ offline
pref_min_conversions
The user's assessment of the utility of the minimisation of the number of conversions
Type: range Value: 0 ≻ 1 ≻ 2 ≻ 3 ...

Table 6: Default adaptation schedule preferences in CAIN-21

For the static decisions, the *ConversionCapabilities* elements store pre-calculated values for the execution cost and content degradation. The main advantage of pre-calculating these values is that the decision can be made quickly. The downside is that these numbers are estimated values based on the analysis of previous adaptation sequences. Subsection 3.1.2 of Chapter 7 will demonstrate that, in general, these estimated values help in choosing the adaptation that best fits the user's preferences.

Subsection 4.1 of this chapter describes a dynamic decision method that further analyses the content degradation. To make a decision in this method, instead of just labelling the estimated degradation of the conversion, the method calculates the real content degradation in the different layers of a scalable visual stream.

3.2 Best-outcome-based decisions

The previous subsection analyses how different sequences in the **SSOC** suit the user's preferences throughout the conversion steps of each sequence. Different sequences of conversions change different groups of properties and produce different values. This subsection analyses how the user's preferences suit the outcomes of these sequences. In particular, given a sequence with multi-valued parameters, this analysis has to determine which outcomes can be achieved and which outcome is the best.

3.2.1 Obtaining the optimal outcomes

Subsection 7.5.2 of Chapter 2 introduced the concept of *theoretical outcomes*. The theoretical outcomes ($\mathbf{X}$) correspond to the Cartesian product of the preferences variable domains, i.e., $\mathbf{X} = X_1 \times X_2 \times ... \times X_M$. That subsection also introduced the concept of *feasible outcomes* ($\mathbf{O}$), which correspond to the subset of theoretical outcomes, i.e., $\mathbf{O} \subseteq \mathbf{X}$, where the variable assignments fulfil the hard constraints of the environment (see Fig. 11 of Chapter 2). Finally, the last part of that subsection defined the notion of *optimal outcomes* ($\mathbf{O}^*$) as the subset of feasible outcomes, i.e., $\mathbf{O}^* \subseteq \mathbf{O} \subseteq \mathbf{X}$, that are Pareto optimal.

Subsection 7.5.3 of Chapter 2, however, did not provide the criteria to decide whether a specific value ($o_1 \in \mathbf{o}_1$) is better or worse than another value ($o_2 \in \mathbf{o}_2$). The preference model developed in Section 2 of this chapter is in charge of gathering these criteria. Then, this current subsection combines both ideas and describes how to accomplish the decision of which of these outcomes is the best. In particular, to accomplish the decision of the best outcome, we propose combining the adaptation engine preference model (which was developed in Section 2 of this chapter) and a preference graph (described in Subsection 7.5 of Chapter 2).

This study uses six preference variables. Table 6 shows three preference variables for the adaptation schedule and Table 7 shows three additional preference variables for the media format. The preference variables in Table 6 have an implicit and hardcoded ranking in their list of values (see Subsection 7.1 of Chapter 2). For instance, *pref_content_degradation* has the ranking values (*lossless* $\succ$ *lossy*) so that *lossless* is always considered better than the *lossy* value. In the case of *pref_min_conversions*, the best choice is the one that minimises the number of conversions. Therefore, the user can only provide the utility for the variables in Table 6. Conversely, the preference variables in Table 7 have an explicit ranking (i.e., the user can provide this ranking). In addition, the user can provide a utility for each variable.

pref_file_format
The user's assessment of the utility of the file format along with a ranking over the list of values
Type: range Value: mpeg-4 $\succ$ mpeg-2 $\succ$ mpeg-1
pref_visual_format
The user's assessment of the utility of the visual format along with a ranking over the list of values
Type: range Value: MPEG-2 MainProfile@MainLevel $\succ$ MPEG-4 Visual Simple profile $\succ$ MPEG-4 Visual Advance Simple Profile)
pref_audio_format
The user's assessment of the utility of the audio format along with a ranking over the list of values
Type: range Value: AAC $\succ$ MPEG-2 Audio $\succ$ AMR

Table 7: Default media format preferences in CAIN-21

The first step in collecting the user preferences is by means of a GUI. The software that collects these preferences is not relevant as long as it generates a description compatible with the *cde:UsagePreferencesType* description tool proposed in Subsection 3.2.2 of Chapter 7. Once these preferences have been gathered, a preference graph can be generated to decide which is the best outcome. Subsection 7.5 of Chapter 2 described how to generate this graph.

For example, let us assume that the adaptation engine has provided the default preference rank of values in Table 6. Let us also assume that the *Planner* has produced a **SSOC** with two sequences: the first sequence can produce MPEG-2 video online and in three steps. The second sequence can produce both MPEG-2 and MPEG-4 video on demand and in only one step. In this case, the boxes in Fig. 20 (a) correspond to these three *feasible outcomes*. Arrows point to preference relations, and the bold boxes correspond to the *optimal outcomes*. The set of optimal outcomes comprises the Pareto frontier.

Now, let us assume that the user indicates his/her preference for MPEG-4 video format to MPEG-2 video format. Fig. 20 (b) shows the new preference graph that results from this information. At this time, the new preference graph contains a new arrow that goes from the MPEG-2 outcome to the MPEG-4 outcome. Consequently, we can reduce the Pareto frontier. In this example, we have obtained a new Pareto frontier with only one outcome.

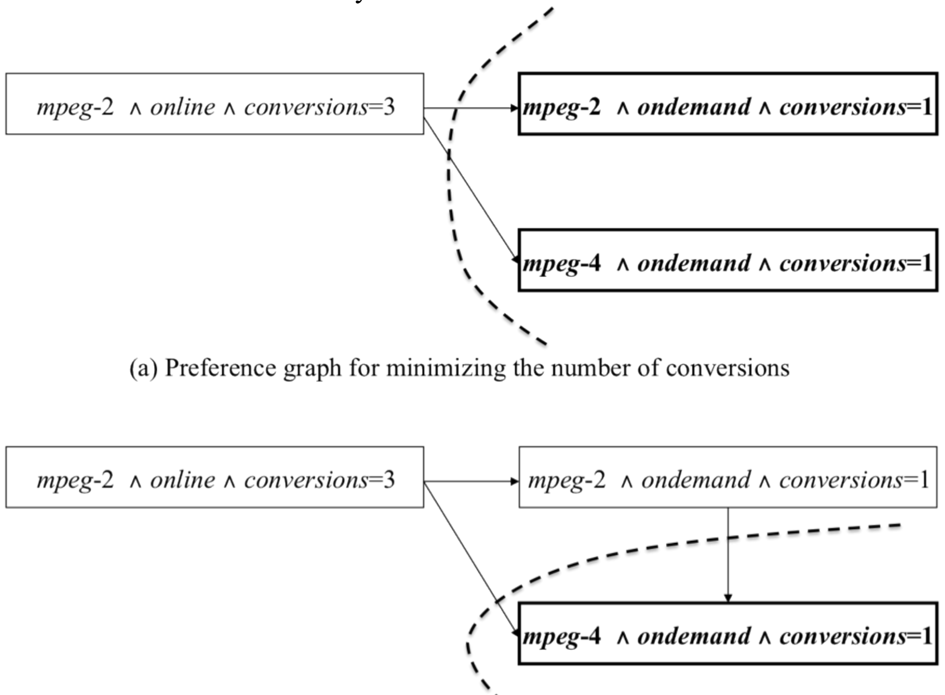

mpeg-2 ∧ *online* ∧ *conversions*=3
***mpeg*-2 ∧ *ondemand* ∧ *conversions*=1**
***mpeg*-4 ∧ *ondemand* ∧ *conversions*=1**
(a) Preference graph for minimizing the number of conversions
mpeg-2 ∧ *online* ∧ *conversions*=3
mpeg-2 ∧ *ondemand* ∧ *conversions*=1
***mpeg*-4 ∧ *ondemand* ∧ *conversions*=1**
(b) Preference graph for the media format and number of conversions

Fig. 20: Example of theoretical and feasible outcomes in a preference graph

3.2.2 Optimality

Once the set of optimal outcomes has been obtained, according to the Pareto principle, this set will contain the equally "best" outcomes (i.e., at this stage, suboptimal solutions would have been discarded). The Pareto frontier is defined in terms of the level of achievement of the available (and possibly incomplete) preferences. If there were several outcomes within the Pareto frontier and no more preferences available, whatever outcome we pick would be optimal.

This means that multiple optimal outcomes may occur if there are not enough preferences, and this result can be observed in many practical cases. Subsection 2.6 of this chapter has explained that forcing the user to choose all the preferences is not, in general, a good idea. In the case in which more information is available, this information would have been used to further reduce the set of optimal outcomes. Nonetheless, if we are uncomfortable with this "unpredictable" behaviour we can define a total order in the default adaptation engine preferences. In this case, the fallback mechanism (see Subsection 2.5 of this chapter) guarantees unique and deterministic results.

3.2.3 Evaluation of the MPEG-21 preferences

After studing the MPEG-21 Part 7 description elements and its suitability to represent outcomes, three shortcomings were identified in the user preferences.

1. The *mpeg21:UsagePreferencesType* description tool makes reference to the *mpeg7:FilteringAndSearchPreferences DS* (introduced in Subsection 3.3.1 of Chapter 2). This description scheme defines a weighing mechanism for different preference values of a particular variable (e.g., different spoken languages with different preference weights), but it does not provide a weighing mechanism among different preference variables (e.g., spoken language vs. media modality).
2. As discussed in Subsection 2.4 of Chapter 6, the *mpeg21:UsagePreferencesType* description tool comprises only the human user's preferences and not those of the adaptation engine. In this work, we are interested in examining a constraint hierarchy in which the adaptation engine preferences are interpreted as the default preferences so that we can use the fallback mechanism. Thus, it is possible to remove this incompleteness (see Subsection 3.2.2 of Chapter 6) in its corresponding preference graph.
3. The standard *mpeg21:ConversionPreferenceType* description tool is ambiguous because it includes two weighing mechanisms defined by the *order* and *weight* attributes. The *order* attribute represents the qualitative preference of the user for that conversion. The *weight* attribute provides the same information in a quantitative manner.

Section 3.2.2 of Chapter 7 extends these standard description tools to address the first two shortcomings. In reference to the third shortcoming, MPEG-21 Part 7 states, "*User preference for a conversion is divided into two levels, qualitative and quantitative. First, a User can specify the relative orders for possible conversions of each original modality or format. The orders help an adaptation engine find the destination modality or format when the original one needs to be converted under a given constraint. Second, a User can further specify the numeric weights for conversions, which can be considered as a User's QoS preferences on the conversion of one modality or format to another.*"

In order to address this third shortcoming, we propose reducing this standard two-level preference relation to a one-level qualitative preference relation. Subsection 7.2 of Chapter 2 has described how to convert a quantitative preference relation into a qualitative preference relation. It could be argued that removing the quantitative preference relation reduces the description capabilities; however, this is not the case here. The conversion preferences are discrete relationships and therefore they could be described using only a ranking of the preference values. In addition, as explained in Subsection 7.4 of Chapter 2, ranking queries require less cognitive effort from the user than providing numerical values (utility queries). To make this change in the description tools of

CAIN-21, Listing 21 of Appendix A modifies the content of the standard *mpeg:21ConversionPreferenceType* in order to remove the optional *weight* attribute. Listing 21 maintains the standard and mandatory *order* attribute. In Listing 22 of Appendix A, the *User* element with *id="cain21"* shows how CAIN-21 uses this description tool to describe its default conversion preferences.

4 Dynamic decisions

Clearly, physical and technical aspects of the terminal and network (such as the screen size or the bandwidth) limit the user experience. In CAIN-21, the *Planner* is in charge of these technical aspects by selecting the feasible sequences. In addition, the *Planner* is in charge of the static decisions (according the user preferences).

However, once all of these technical aspects and static preferences have been considered, the user's experience (i.e., the utility) can be further improved by analyzing the quality and the semantics of the adapted content. This section describes how these dynamic decisions (i.e., quality-based and semantic-based) can be accomplished[18].

In CAIN-21, the *Adapters* are the modules responsible for making the dynamic decisions. These modules are specialized modules that transform particular type of content (e.g., PNG images, MPEG-2 video or standard scalable extension for the H.264/AVC video).

4.1 Quality-based decisions

Quality-based adaptation decisions measure the objective or subjective quality of the adapted content in order to improve the user's experience. This subsection demonstrates the use of objective video quality metrics to make decisions during the adaptation of scalable video.

To analyse scalable video adaptation, this research uses an *Adapter* named *SVCAdapter*. With scalable video, frequently the *Planner* identifies multi-valued parameters for the *SVCAdapter*. In this case, all of these parameters fulfil the hard constraints. In addition, the *Planner* makes the static decisions that guarantee that all the values of these parameters are Pareto optimal. Once all of these parameters have been obtained, this subsection describes how the *SVCAdapter* selects which values of these parameters produce the maximum visual quality.

4.1.1 The adaptation process

Fig. 21 shows the order of the scalable video adaptation process involving dynamic decisions. The *transform*() operation (see Subsection 2.1 of Chapter 3) receives two DIs. The *Content DI* represents both the scalable video and its metadata. In this case, the metadata is an *Adaptation-QoS* description tool (see Subsection 3.3.4 of Chapter 2). Using this tool, the available layers of the scalable video, the offline pre-computed bitstream components and the quality of each adaptation are described. The second DI is the *Context DI*, which stores the description of the current usage environment.

In the *decide*() operation, the *Planner* selects the **SSOC** and subsequently the *Planner* makes a static decision (i.e., using the user preferences) that guarantees that the remaining outcomes will

[18] Authors such as Prangl et al. [105] use the terms *perceptual quality* and *semantic quality* to refer to the decision parameters aiming to improve the user experience.

be Pareto optimal. At this moment, one sequence of conversions is chosen (*soc*∈**SSOC**), but it may remain multi-valued parameters. Therefore, the sequence is transferred to the *execute*() operation. This operation is implemented in the *Executer* module and is responsible for executing the conversions in this sequence. During its execution, the *SVCAdapter* chooses the optimal parameters.

In the simple case (as happens in Fig. 21), the sequence to execute contains only one step in which the *SVCAdapter* is executed. In other cases, this sequence contains several conversions involving different *Adapters*. For instance, Subsection 3.1.2 of Chapter 7 will provide the results of adapting scalable video to legacy and non-scalable video terminals (e.g., an standard MPEG-1 terminal). In this case, the scalable video is first adapted and during a second step, it is transcoded to the terminal's video format.

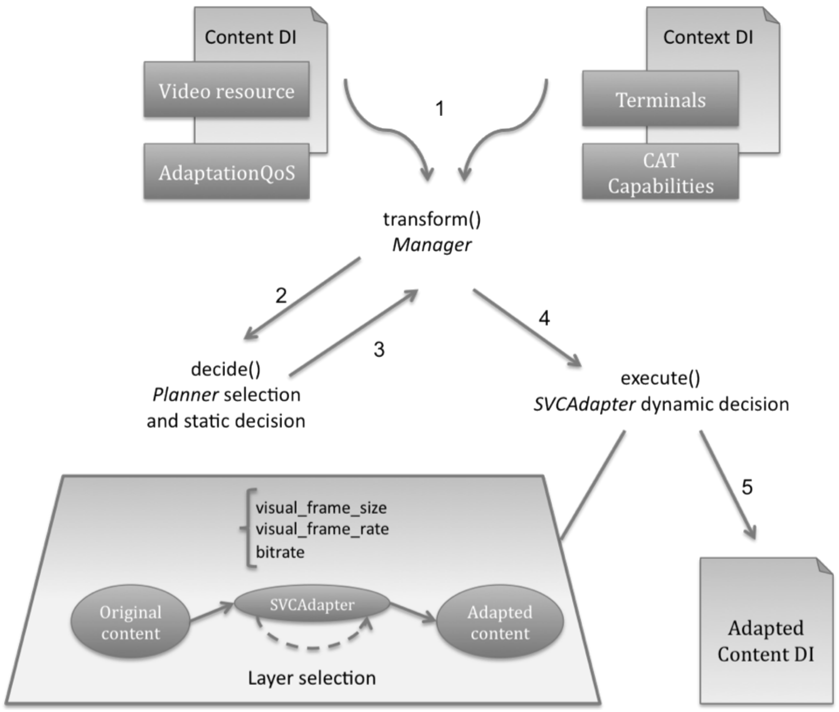


Fig. 21: Order of the scalable video adaptation process involving dynamic decisions

4.1.2 Using objective quality metrics to choose the best layer

After executing these tests for scalable video adaptation (further described in Subsection 4.1.2 of Chapter 7), three parameters remain multi-valued at the end of the planning phase. In Fig. 21, these parameters are labelled as *visual_frame_size*, *visual_frame_rate* and *bitrate*. These multi-valued parameters correspond to the multiple layers that usually the scalable video has. In this case, these multi-valued parameters are transferred to the *SVCAdapter*, which makes a dynamic decision on the target frame rate, frame size and bit rate. The *SVCAdapter* consults the *AdaptationQoS* description to decide which layer fulfilling the constraints of the parameters represents the best quality.

Three important difficulties that must be considered during the development of this kind of quality-based dynamic decision have been identified:

1. There are several well-known objective quality metrics to make the scalable video layer selection. This work has considered three of them: PSNR, VQM and SSIM (see Section 2.3 of Chapter 2). Fortunately, the *AdaptationQoS* supports the description of several quality metrics for each layer. If the *AdaptationQoS* contains several metrics, the user preferences can be used to decide which metrics to use. If the user has not provided his/her preferences, the fallback mechanism of CAIN-21 has to decide which metric to use. In this latter case, the authors of [17] have proposed to conduct a study of the subjective quality to determine which objective metric is the best for each kind of video content and genre.
2. If the *Content DI* is not labelled with an *AdaptationQoS* description, the *Adapter* could compute in runtime this information, make the quality-based decision and append the *AdaptationQoS* to the adapted *Content DI*. Clearly, in this case, the adaptation process would be slower.
3. With scalable video, sometimes the *Planner* does not properly identify all the feasible solutions. This happens because, usually, the terminal accepts decimated versions of the content. For instance, if a scalable video terminal accepts 352x288 pixel frame size, then it also accepts 177x144 frame sizes. The same happens with the frame rate; if the terminal accepts 30 fps, then the terminal also accepts 15 fps. This implict capability of the terminal to accept decimated versions must be properly provided in the description of the terminal capabilities.

4.1.3 Relationship between quality-based and best-sequence decisions

Subsection 3.1 of this chapter describes how to pre-compute and label the execution cost and content degradation of each conversion in order to make future decisions. The conducted experiments have revealed that, with scalable video, the execution cost of the layer extraction operation is very low and practically does not depend on the selected layer. Therefore, with scalable video the execution cost can be dismissed (i.e., we can assign 0 to the execution cost of the *SVCAdapter*).

Regarding the content degradation, scalable video allows for accurately determining the real content degradation. Therefore, if the *AdaptationQoS* is available, with the *SVCAdapter* it is not necessary to pre-compute and label the estimated content degradation. Furthermore, the real content degradation can be easily mapped to the content degradation values. In particular, using the raw (unencoded) original media is a lossless conversion because it preserves all the available information, and thus its content degradation is exactly 0. Choosing the highest layer would imply a very low content degradation and so its content degradation would be nearly 0. On the other hand, removing all the media corresponds to the maximum content degradation (i.e., the content degradation would be exactly 1). Similarly, removing all the enhancement layers (i.e., retaining only the base layer) would produce a relatively high content degradation value.

4.2 Semantic-based decisions

Semantic adaptation (introduced in Subsection 4.3 of Chapter 2) takes into account the meaning of the multimedia content to improve the user experience. An *Adapter* named *Image2VideoAdapter*[19] has been developed to investigate these kinds of semantic-based automatic decisions. The *Image2VideoAdapter* receives an image and produces a video. The main purpose of the *Adapter* is to enable the presentation of large images on small video terminals.

[19] F. Barreiro, J. M. Martínez and V. Valdés have conducted the implementation of this software. Their work was published in [49]. For the purposes of this research, their software has been wrapped inside the *Image2VideoAdapter*. In addition, the description documents of CAIN-21 have been created and the performance of the adaptation tests has been measured.

This subsection describes how automatic decisions are implemented in this *Image2VideoAdapter*. Subsection 4.2 of Chapter 7 describes several adaptation tests.

The *Image2VideoAdapter* allows further control of the adaptation through ROIs. The MPEG-7 Part 5 *StillRegionType* description tool is used to represent the ROIs. For instance, in Fig. 22 the ROIs are the faces in the photo. The description of the ROIs is not mandatory in the *Content DI*. If present, the *Image2VideoAdapter* uses these regions to semantically improve the adaptation. Otherwise, the *Image2VideoAdapter* shows a zoomed view of the image that scrolls through the whole image. Further functionalities, such as face detection, could be easily added in the future.

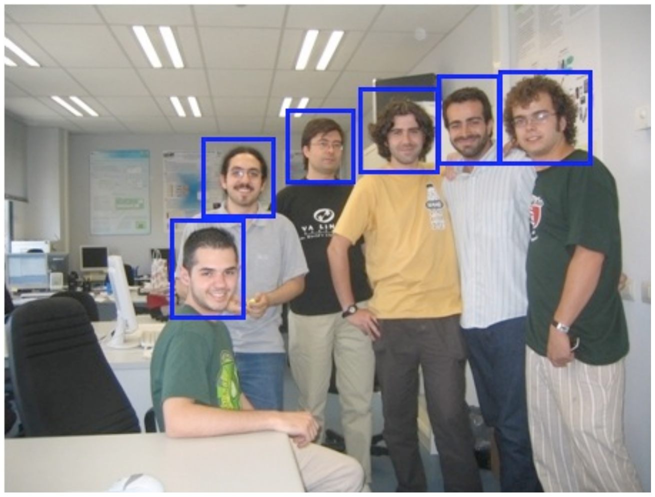


Fig. 22: Example of a resource to be semantically adapted

Listing 10 shows a *Content DI* named *people_roi_di.xml* labelled with ROIs using the *mpeg7:StillRegionType* description tool. Each region is labelled with a *mpeg7:Box* description element.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:cdi="urn:vpu:cain21-di"
      xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004">
 <Item xsi:type="cdi:ItemType">
  <Component xsi:type="cdi:ImageComponentType" id="c1">
   ..................
   <Descriptor xsi:type="cdi:Mpeg7DescriptorType">
    <Statement mimeType="text/xml">
     <Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2004">
      <DescriptionUnit xsi:type="StillRegionType" id="faces">
       <SpatialMask>
        <SubRegion>
         <Box mpeg7:dim="2 2"> 136 178 165 212</Box>
        </SubRegion>
        <SubRegion>
         <Box mpeg7:dim="2 2"> 157 108 181 135</Box>
        </SubRegion>
        ...........
        <SubRegion>
         <Box mpeg7:dim="2 2"> 357 73 378 98</Box>
        </SubRegion>
       </SpatialMask>
      </DescriptionUnit>
     </Mpeg7>
    </Statement>
   </Descriptor>
   <!-- The resource itself -->
   <Resource mimeType="image/jpeg"  ref="people.jpg"/>
  </Component>
 </Item>
</DIDL>
```

Listing 10: Example of a *Content DI* with ROIs

The MPEG-21 Part 7 UED does not take into account the MPEG-7 semantic description tools. This handicap has been addressed in [29] by extending the MPEG-21 Part 7 UED description capabilities. This extension has not been necessary for this work. In CAIN-21, the user's preferences are used to indicate which group of ROIs the user is interested in. As shown in the example in Listing 11, CAIN-21 uses the *mpeg21:FocusOfAttentionType* description tool to indicate the ID of the ROIs. In Listing 11 the user is interested in the ROIs labelled as *"faces"*. These ROIs have been provided in Listing 10.

```
<User id="roi_faces_preference" xsi:type="UserType">
 <UserCharacteristic xsi:type="FocusOfAttentionType">
  <ROI uri="#faces"/>
 </UserCharacteristic>
</User>
```

Listing 11: User preference for the ROIs

During the decision process the value of the *mpeg21:FocusOfAttentionType* description tool is stored in the *roi* property as an optional precondition. This optional precondition is satisfied if the *Content DI* provides the value of this property. In this case, sequences of conversions that provide this property are considered preferable to other sequences that do not provide this property. During the execution phase, the *Executer* transfers this property to the *Image2VideoAdapter*, which performs the semantic conversion taking into account this property.

5 Preference-based decisions comparison

In the literature, independent authors have employed the user preferences to decide which adaptation and parameters best increase the multimedia experience. However, these studies are closely focused on using preferences to make decisions in specific application domains. This chapter studies the use of these preference-based decision methods in a systematic and general manner. The chapter has identified the decision points in which these decision methods can take place. CAIN-21 has been used to demonstrate how these decisions can be integrated.

The following subsection separately compares these preference-based decision methods with instances of these decision methods in the literature. First, it compares the static decision methods that eliminate suboptimal sequences and parameters. Second, it compares the dynamic decision methods that use the preferences to steer the adaptation of particular media or multimedia resources.

5.1 Best-sequence-based decisions comparison

The problem of deciding which sequence of conversions is the best has been partially addressed in koMMa [16] and MAGG [56]:

- koMMa uses a forward search planner. The first version of this planner [94] only aims to encounter one feasible sequence and does not study the problem of analyzing different sequences. In a subsequent work [16] its creators highlight that, "*The general challenge with respect to plan quality is that there may be several possible plans to reach the same goal. While we consider the selection of alternative tools for the same transformation step as a minor problem, the effectiveness of the produced plans can heavily depend on the order of the transformation steps.*" In this work, they also propose for future work optimizing the selection of the tools in the sequence. Specifically, they state, "*we are currently investigating how we can introduce general concepts for expressing such search and optimization strategies as well as heuristics into our framework. In particular, we see opportunities in re-using and extending the already existing mechanisms for providing adaptation hints in the context of Adaptation-QoS.*"

- MAGG [56] has a backward search planner that searches for the optimal path for multimedia adaptation. In particular, MAGG constructs a *Directed Acyclic Graph* (DAG) in order to study different sequences of conversions. They take into account the execution cost of the adaptation (but not the content degradation) and search for the optimal path using the Dijkstra's algorithm.

Chapter 1 has pointed out that the use of multi-step adaptation modules increases the level of software reusability and the range of adaptations that can be achieved. However, in general, multi-step adaptation reduces execution performance. This happens because adaptation modules have to be executed in sequence and each individual module has to wait for the termination of the previous module. In addition, multi-step adaptation may reduce the quality of the adapted content because the media may have been quantized or recompressed several times.

This observation provided a motivation for the development of the first best-sequence-based innovation of the *Planner*. In addition to searching for the optimal path, the *Planner* also takes into account that sometimes there is a trade-off between the execution cost of the sequence and the content degradation; sequences with a higher execution cost may produce a lower content degradation. For this reason, in comparison with the previous research work, this research takes into account two additional criteria:

1. The analysis of the trade-off between the execution cost and the content degradation.
2. The use of soft constraints when different criteria produce different optimal results. Specifically, this work uses the constraint hierarchy of the preference model developed in Subsection 2.4 of this chapter to consider both the user's and the adaptation engine's preferences.

The static decision criteria (i.e., execution costs and content degradation) are just estimated values. The dynamic decision criteria (i.e., quality metrics in the scalable video) are more accurate measures. The experiments in Section 3 of Chapter 7, nonetheless, will reveal that these estimated values for static decisions are, in most cases, sufficient to improve the result of the adaptation decisions in comparison to a decision system that does not take into account the execution costs or the content degradation (such as the multi-step multimedia adaptation system developed in [16][56]).

Note that it is always possible to assign the same value to the execution costs and content degradation for all of the conversion modules (e.g., 1 to the execution costs and 0 to the content degradation). In this manner, the problem of deciding the best sequence is reduced to the problem of deciding the minimum length. Thus, the method proposed here is a generalization of the model proposed in [56].

The second best-sequence-based innovation of this research is the use of *non-deterministic conversion states* in order to postpone decisions to the execution phase. In this model, the static decisions are made during the decision phase, and during this phase, additional dynamic decisions are integrated.

The third best-sequence-based innovation is founded on the realization that the planners may over-diminish the content. This happens, for instance, if a specific conversion heavily scales down the frame size and the next conversion scales up the content. Although the quality of the content is over-diminished, from the point of view of the planner this parameter configuration has produced adapted content. In this research, the accumulated effects property of the *Planner* (see Subsection 3.3 of Chapter 5) avoids the over-diminished downside effect because the accumulated effects guarantee that the properties are modified a minimum number of times. Therefore, over-diminishing the content only occurs if (due to technical reasons described in the conversion capabilities) a specific step cannot produce higher quality content. For instance, if the terminal spatial resolution was *frame_size*={352x288}, the *frame_size* property would be set up during the computation of the target parameters of the last conversion in the sequence. Next, this value would be transferred to the source parameters of this conversion and then transferred again backward along the chain of previous conversions up to the first one. This parameter setup would only change if one specific conversion did not support this size. In this case, the *frame_size* property would have to be modified again in the backward path. The effect of the same minimum number of changes would also occur with multi-valued properties if the terminal, for instance, supported *frame_size*={176x144, 352x288}.

5.2 Best-outcome-based decisions comparison

Best-outcome decisions rely on the user preferences to decide which outcome **o** from a set of feasible outcomes **O**, i.e., **o**∈**O** best suits the user preferences. This concept has been described in Subsection 7.5.3 of Chapter 2. In the field of multimedia adaptation, this concept has been referred to as Pareto optimality [7][13][107] or skylines queries [53][70]. These works have identified the existence of multiple solutions involved in a partial order preference relation.

Köhncke et al. [53] have implemented preference-based static decisions to eliminate outcomes outside the skyline. In their work, the *mpeg21:UsagePreferencesType* description tool provides a qualitative partial order of the user's preferences. The authors then build a preference graph (see Subsection 7.5 of Chapter 2) in which they show how to use this graph to obtain the skyline. They also explain that due to the qualitative nature of the preferences, some combinations are incompatible. In this case, the adaptation decision can explore several other equally preferable options.

A comparable approach, followed in [37][99], is the use of utility values (instead of preference relations). In this case, as described in Subsection 7.3 of Chapter 2, standard optimisation techniques can find the maximum of a multiattribute optimisation problem.

The literature has identified a lack of expressiveness in representing certain kinds of preferences. For instance, Köhncke et al. [53] state, "*the MPEG-7/21 standard still lacks expressiveness. In the course of this paper we demonstrate this shortcoming...*" Then, they propose extensions to the *mpeg21:UsagePreferencesType* description tools to combine several preferences. Independently, Zufferey et al. [50] state, "*no facilities are provided for the description of user preferences expressed in terms of semantic entities and their relations.*" Similarly, Tsinaraki et al. [29] state, "*The MPEG-7 Semantic DS have powerful semantic description capabilities and [support] using semantic entities specified in domain ontologies in multimedia content descriptions. However, the*

MPEG-7/21 usage environment allows neither the specification of semantic user's preferences nor the exploitation of domain knowledge and MPEG-7 semantic metadata description." Even though this current chapter was initiated with the intention of developing a preference model that meets the MPEG-21 description tools, expressiveness limitations (described in Subsection 3.2.3 of Chapter 6) have been also encountered. This subsection provides a solution that is consistent with those of the above-mentioned authors.

The *mpeg7:FilteringAndSearchPreferences DS* merges search preferences (such as the movie title) and adaptation preferences (such as the video format). The preference model proposed by Köhncke et al. in [53] also merges these preferences. In contrast with previous works, Subsection 2.2 of this chapter recommends making a distinction between these two kinds of preferences. As CAIN-21 is a multimedia adaptation engine, it only makes use of the adaptation preferences. In addition, CAIN-21 introduces new adaptation preferences that are not considered by the standards.

Previous multimedia adaptation methods did not consider the suitability of forcing user choices. The preference model in this work, however (see Subsection 2.5 of this chapter), takes into account that making the user fill in all his/her own information gives rise to a set of preferences that may not properly describe the wishes of the user. To address this issue, this work proposes defining a constraint hierarchy. With this hierarchy, the adaptation engine's preferences are interpreted as default preferences, making use of the fallback mechanism. In this way, it is possible to remove the incompleteness (see Subsection 3.2.2 of this chapter) in its corresponding preference graph. The constraints hierarchy also allows the preference model to keep separate the hard constraints and the soft constraints [97]. As far as we know, the distinction between hard and soft constraints has not been considered in the other multimedia adaptation decision systems.

5.3 Quality-based decisions comparison

Subsection 3.3.3 of Chapter 2 introduces the MPEG-21 BSD tools. This tool enables to steer the adaptation of a binary media resource. The adaptation is made in a format independent manner. Several studies have used this tool to perform utility-based adaptation. Mukherjee et al. [13] clearly describe how to associate the content with the metadata. This relationship associates feasible adaptation choices and pre-computed utilities so that the utility is a function of the choice. They also explain how to cast the dynamic decisions into a generic constrained optimization problem with integer variables. Hutter et al. [58] present a server-side adaptation engine that reacts to context changes and accordingly modifies scalable video bitstream adaptation. Iqbal et al. [108] demonstrate how to perform H.264/AVC video frame dropping decisions in the compressed domain. Klofer et al. [109] have investigated the use of end-to-end-based rate control algorithms for steering the adaptation of scalable video. Wang et al. [37] present a content-based statistical paradigm to facilitate the prediction of the quality functions. Instead of modelling through analytical models (e.g., rate-distortion), they formulate the prediction of the quality as a classification and regression problem. In particular, each video is assigned a unique category and then local regression is used to predict the quality value. Two different solutions to address the performance in the generation of the BSDs have been proposed in [59] and [107]. The research in Subsection 4.1 of this chapter also addresses the problem of making these dynamic decisions, but also contributes to the following areas:

- In previous works, the problem of defining the feasible outcomes before computing the optimal adaptation is not addressed. They simply assume that these feasible outcomes exist. In this research, the *Planner* defines a systematic method for obtaining the feasible outcomes, and then transfers these outcomes (represented as multi-valued properties) to the *Adapter* that performs utility-based decisions.

- This research computes the utility by measuring the objective quality according to different metrics.
- This research permits the use of the user preferences (including the adaptation engine preferences) to select which of these quality metrics to use.

Authors such as [99][47] have also exploited the preferences to guide the adaptation of their media resource, but taking into account other criteria. In particular, [99] takes into account the user's perceived utility, and [47] extracts desired fragments according to the user preferences.

The bit rate of a scalable video can be reduced if this video contains only the layers that have a chance to be used. The elimination of unfeasible layers can be easily accomplished dynamically inside the *SVCAdapter*. This elimination of unfeasible layers has been investigated in [110].

Another difficulty with scalable video is that it complicates the customization of the video for a user subscribed to a multicast stream. This issue can be also addressed in the *SVCAdapter* if the base and enhancement layers are sent through different channels. This solution has been studied in [111]. In this proposal, the authors use different *Real-time Transport Protocol* (RTP) channels to send the base and enhancement layers.

5.4 Semantic-based decisions comparison

During the execution phase, the content can be analyzed and its semantics extracted to improve the user's experience.

To perform the decision phase of the semantic adaptation (i.e., semantic-based automatic decisions), authors have used different methods. Usually, these methods label the meaning of the different parts of the content. Subsection 4.2 of this chapter demonstrates how CAIN-21 performs semantic adaptation decisions involving the image to video adaptation and in which the ROIs are labelled.

Different comparable semantic adaptation decisions can be found in the literature. Metadata with transcoding hints are frequently used in the dynamic decisions. For example, Kodikara Arachchi et al. [100] get access to the video resource by cropping the ROIs. M. Prangl and I. Kofler [99] get access to the media resource to measure its perceptual and semantic quality, thus taking into account the user preferences to decide the adaptation to be carried out. The BSD tools are used in [10][59] to annotate the level of violence in the scenes. In this way, violent scenes can be extracted or eliminated according to the end-user's preferences. S. Kim and Y. Yoon [106] use content mining to rate cognitive content (i.e., informative) and affective content (i.e., emotional). Then, they insert or eliminate each video shot depending on user's preferences.

As already described in Subsection 5.2 of this chapter, authors such as [29][50] have realized that the MPEG-21 UED does not allow the description of user's semantic preferences, and so they have proposed several extensions based on the MPEG-7 *Semantic DS*. In the demonstration for image to video adaptation, the user just has to provide the ID of the ROIs that he/she is interested in, and therefore it is not necessary to use the MPEG-7 *Semantic DS*.

6 Conclusions

Usually, multimedia adaptation systems have used the user preferences to increase the multimedia experience. In the case of MPEG-21, the preference model combines adaptation and search preferences. This chapter starts by separating the adaptation preferences and then it goes on to develop a preferences representation model for multimedia adaptation. Previous work in the area of multimedia adaptation has only taken into account the preferences of human users (typically, the content provider, the content creator and the end-user). The proposed preference model also considers the adaptation engine preferences. In addition, the preference model brings to multimedia adaptation the novel facet of incomplete preferences. In this way, the user is not forced to provide the preferences whose meaning the user does not completely understand. The preferences are classified using a constraint hierarchy with three levels. In this hierarchy, the fallback mechanism enables the transparent inclusion of the adaptation engine's preferences whenever the user does not provide them.

The chapter has identified and analyzed the decision points in which the user preferences can be exploited. This analysis justifies the classification of the preference-based decision methods according to these decision points. To integrate and coordinate all these decisions methods, they have been classified into static and dynamic methods. The static decisions are applied during the decision phase and the dynamic decisions during the execution phase. Static decisions focus on choosing the sequence of conversions, parameters and outcomes that best suit the user's preferences. Dynamic decisions focus on guiding the adaptation of the resource to increase the user's satisfaction. All these adaptation points have been demonstrated in the CAIN-21 adaptation engine.

In the current literature, there is no consensus on which preferences should be used and the decision point in which each group of preferences can be used. Different authors have employed the preferences at different levels and in an ad-hoc manner. In addition, their multimedia preference model and set of preferences vary widely. The last part of the chapter aligns the preference-based decision methods in the literature with the decision points in the proposed preference model.

Overall, this chapter has shown that the user preferences are frequently necessary to decide the adaptation that maximizes the user's experience.

PART III:

RESULTS AND CONCLUSIONS

Chapter 7:

Experiments and demonstrations

Synopsis:

This chapter has been organized according to the systematic and automatic multimedia adaptation decision-making methods described in the previous chapters. Here, different experiments and demonstrations illustrate the decision methods' results, exemplify their applicability, evaluate their performance, and validate the results of this thesis.

1 Introduction

This chapter validates the proposals resulting from the thesis by showing their applicability scope, assuring that they fulfil the intended purpose and measuring their performance. To validate the proposals the thesis uses three tools:

- The *demonstrations* (i.e., demos) show the viability of the chosen approach by describing several adaptation tests that show the advantages of the proposals.
- The *experiments* provide evidence that support the hypotheses that were assumed to be true in Chapter 5.
- The *theorems* in Section 5 of Chapter 5 have already been formally *proven*, so this chapter does not provide further demonstration or experimentation to support them.

The ultimate practical objective of this thesis is systematic and automatic multimedia adaptation decision-making and therefore the demonstrations have been classified according to the decision points developed in Chapters 5 and 6. In particular, these decision points are the selection, the static decisions, which include best-sequence-based and best-outcome-based decisions, and the dynamic decisions, which include quality-based and semantic-based decisions. In addition, Section 5 of this chapter demonstrates how the extensions to the MPEG-21 schema remove the ambiguities that Chapter 4 points out.

2 Selection process

Subsections 6.1 and 6.2 of Chapter 5 hypothesised that, in reference to the selection process, the multimedia properties together with the incompleteness semantics (dicussed in Subsection 6.3 of Chapter 5), significantly reduce the description length of the adaptation capabilities. In addition, Chapter 5 hypothesised that these conditions allow for the development of efficient AI planning. To support these two hypotheses, this section contains some experiments that shed light on their validity.

2.1 Claims and hypotheses

The subsequently reported experiments focus on providing evidence for the following claims:

(Claim 1) A partial description of the adaptation capabilities suffices for computing all the feasible adaptation plans.
(Claim 2) Requiring only partial description of the conversion modules significantly reduces the description length of the adaptation capabilities.
(Claim 3) Requiring only partial description of the conversion modules significantly reduces the decision time.

Claim 1 has been theoretically proven in Subsection 5.2 of Chapter 5. With respect to Claim 2 and 3, the following subsections test three hypotheses.

(H1) The average-case computational cost of the *Planner* is significantly[20] lower than the theoretical worst-case computational cost.

[20] As further described in Subsection 2.4 below, *significantly* means that it is statistically unlikely to have occurred by chance.

(H2) The adaptation capabilities description size decreases significantly when partial description is allowed.
(H3) The decision time decreases significantly when partial description is allowed.

All these hypotheses assume that in both cases (total and partial descriptions) the *Planner* obtains (among others) the same optimal solution. Claim 1 and its founding in Subsection 5.2 of Chapter 5 proved that the *Planner* obtains a complete plan (i.e., all the solution are obtained). Therefore, the optimal solution is the one that, given a best-sequence and a best-outcome criteria, best fits these criteria. However, the solutions that the partial and total descriptions produce may not be the same. For instance, if we replace a multi-valued *audio_format*=* wildcard property (see Subsection 3.2 of Chapter 5) in the partial description by a single-valued property in the total description (e.g., *audio_format*={*mp3*}), then the set of solutions may vary. To obtain the same solutions and make them more comparable, wildcard properties have been replaced with multi-value properties that include all the feasible values (e.g., *audio_format*={*wav*, *mp2, mp3, amr, wma, aac*}).

2.2 Theoretical worst-case computational costs of the Planner

This subsection analyzes the theorethical worst-case computational cost[21] of Alg. 2 and Alg. 6 of Chapter 5, both of which are the core algorithms of the *Planner*. The other algorithms described in Section 4 of Chapter 5 serve as subfunctions.

Let C be the number of conversion capabilities elements existing in the available *Adapters*, and N the number of properties of the conversion capabilities to be considered in the matching process of Alg. 6. In the worst-case (assuming that the same conversion capabilities are not instantiated more than once), Alg. 2 would be invoked $C!$ times. Similarly, in the worst-case, Alg. 6 would be invoked C times from Alg. 2, i.e., in the worst-case Alg. 6 is invoked $C \cdot C!$ times, which is of the order of $(C+1)!$. Assuming that N is the upper bound of properties in a conversion capabilities element, Alg. 6 would perform in the worst-case N^2 property comparisons during each invocation. Thus, the theoretical worst-case computational cost of the *Planner* is of the order of $N^2 \cdot (C+1)!$ with respect to the number of comparisons between properties.

2.3 Empirical methodology, dataset and metrics

This subsection justifies the empirical methodology, dataset and metrics used in the experiments. The following subsections conduct the empirical analysis and present the results.

To increase the objectivity of the experiments, adaptation tests aiming to cover different media adaptations were selected. In this way, different sets of properties would be involved in the experiments. Specifically, this research uses most of the adaptation tests available in the CAIN-21 demo, but distributing them into four groups: I→I (Image to image), I→V (Image to video), V→V (Video to video), SVC (Scalable video coding). Table 8 show these 24 adaptation tests distributed in four groups: 6 image to image adaptations, 6 image to video adaptations, 6 non-scalable video adaptations and 6 scalable video adaptations. In Table 8, the last column shows the type of these tests. The adaptation tests use a different number of *AdapterCapabilities* elements and therefore a different number of *ConversionCapabilities* elements. In Table 8 C represents the number of *ConversionCapabilities* elements used in each adaptation test. In addition, each *Con-*

[21] In this work, the term *computational cost* refers to analyzing the theoretical worst-case or average-case computational complexity of the *Planner* algorithms (i.e., the number of times that particular instructions of the algorithm are executed), whereas the term *execution cost* refers to an estimation of the time that would take the execution of the adaptation.

versionCapabilities element contains multiple properties. Therefore, the number of properties involved in the experiments is higher than the number of adaptation tests. Table 9 shows the specific number of properties, *N***,** for each *ConversionCapabilities* element. The tests were implemented and executed in the JUnit testing framework[22]. The source code of these tests is publicly available at *cain21.sourceforge.net*. All these tests were executed in the same hardware, a Mac Book Pro with a 2.4 GHz Intel Core 2 duo and 4GB of *Random-Access Memory* (RAM). It is worth noting that the execution times obtained in these experiments do not depend on the media resource in the *Content DI*, but only on its metadata. This is because we are not measuring the computational cost of the execution phase, but rather the computational cost decision phase.

Test	**Original *Content DI***	***C***	**Target terminal**	**Type**
1	*photo_di.xml*	4	*gray_images_viewer*	I → I
2	*painting_di.xml*	4	*jpeg_images_viewer*	I → I
3	*castle_di.xml*	4	*images_viewer_without_resolution*	I → I
4	*mesh_di.xml*	4	*audiovisual_mobile_1*	V → V
5	*circuit_di.xml*	4	*bmp_image_viewer*	I → I
6	*truck_di.xml*	4	*png_image_viewer*	I → I
7	*park_di.xml*	4	*image_viewer_several_formats*	I → I
8	*comic_di.xml*	7	*mpeg4_mp2_online_web*	V → V
9	*terminator-salvation_di.xml*	9	*mpeg1_720x576_adapted_online_web*	V → V
10	*newsitesm_di.xml*	11	*mpeg1_desktop*	V → V
11	*newscast_intro_di.xml*	11	*flash_player*	V → V
12	*comic_di.xml*	11	*iphone*	V → V
13	*photo_di.xml*	11	*h264_desktop*	I → V
14	*bus_di.xml*	13	*svc_no_audio_176x144_15fps*	SVC
15	*he_asked_di.xml*	13	*mp4_mobile_audio*	SVC
16	*footbal_di.xml*	13	*mpeg2_without_audio*	SVC
17	*bus_di.xml*	13	*mpeg1_without_audio*	SVC
18	*he_asked_di.xml*	13	*mpeg1_with_audio*	SVC
19	*footbal_di.xml*	16	*svc_with_audio_352x288_15fps*	SVC
20	*group_di.xml*	19	*mpeg2_medium_desktop*	I → V
21	*people_di.xml*	19	*mpeg2_big_desktop*	I → V
22	*group_roi_di.xml*	19	*h263_offline*	I → V
23	*group_roi_di.xml*	19	*h263_online*	I → V
24	*tennis_di.xml*	20	*mpeg2_small_desktop*	I → V

Table 8: Adaptation tests

To study H1, the average-case computational cost of the *Planner* has been measured for each test and has been compared with the corresponding theoretical worst-case computational cost. Specifically (see Table 10), the number of invocations to the functions that implement Alg. 2 and Alg. 6 as well as the number of property comparisons within Alg. 6 were counted. To obtain the theoretical worst-case costs, in Table 10 the formule explained in Subsection 2.2 of this chapter were used. In order to provide an upper bound for the theoretical worst-case (assuming partial description), this research assumes N=30 properties (therefore, N^2=900) as the largest number of properties in each conversion capabilities element. In addition, Table 10 has a column with the minimum number of steps in the sequence of conversions needed to perform the adaptation.

[22] Available online at *http://junit.org/*

***ConversionCapabilities* element**	***N***	***ConversionCapabilities* element**	***N***
ondemand_mpeg1_http_video_server	18	*visual_format_image_formats_transcoder*	17
ondemand_mpeg4_http_video_server	18	*ondemand_mpeg_transcoder*	25
online_mpeg1_http_video_server	18	*ondemand_mp4_transcoder*	25
online_mpeg2_http_video_server	18	*mpeg2_online_transcoder*	23
online_mpeg4_http_video_server	18	*raw_video_combiner*	21
online_h264_http_video_server	18	*online_resource_loader*	14
big_image_2_video	25	*svc_without_audio_transcoder*	22
medium_image_2_video	25	*svc_with_audio_transcoder*	22
small_image_2_video	25	*svc_to_mp4*	20
mime_image_formats_transcoder	17	*visual_to_svc*	21
image_formats_transcoder	17	*audiovisual_to_svc*	23

Table 9: Number of properties in each *ConversionCapabilities* element with partial description

Test	**Invocations**					**Theoretical worst-case**		
	C	Min steps	Alg. 2	Alg. 6	Comparisons	Alg. 2	Alg. 6 $(C+1)!$	Comparisons $N^2 \cdot !(C+1)$
1	4	1	4	20	96	2.40e+01	1.20e+02	1.08e+05
2	4	1	4	20	94	2.40e+01	1.20e+02	1.08e+05
3	4	1	4	20	91	2.40e+01	1.20e+02	1.08e+05
4	4	1	2	10	47	2.40e+01	1.20e+02	1.08e+05
5	4	1	4	24	108	2.40e+01	1.20e+02	1.08e+05
6	4	1	4	24	106	2.40e+01	1.20e+02	1.08e+05
7	4	1	4	24	108	2.40e+01	1.20e+02	1.08e+05
8	7	1	2	16	199	5.04e+03	4.03e+04	3.62e+07
9	9	1	108	1080	3648	3.62e+05	3.62e+06	3.26e+09
10	11	1	54	590	1850	3.99e+07	4.79e+08	4.31e+11
11	11	1	30	360	1013	3.99e+07	4.79e+08	4.31e+11
12	11	1	30	360	1017	3.99e+07	4.79e+08	4.31e+11
13	11	3	30	360	1400	3.99e+07	4.79e+08	4.31e+11
14	13	1	231	3756	14745	6.22e+09	8.71e+08	7.84e+11
15	13	2	297	4158	16826	6.22e+09	8.71e+08	7.84e+11
16	13	5	173	2595	10174	6.22e+09	8.71e+08	7.84e+11
17	13	5	212	3870	13131	6.22e+09	8.71e+08	7.84e+11
18	13	5	281	4131	14937	6.22e+09	8.71e+08	7.84e+11
19	16	1	321	5457	21740	2.09e+11	3.55e+12	3.20e+17
20	19	3	131	2620	8162	1.21e+17	2.43e+18	2.18e+21
21	19	3	232	3575	10456	1.21e+17	2.43e+18	2.18e+21
22	19	3	308	5231	18341	1.21e+17	2.43e+18	2.18e+21
23	19	3	312	6040	19837	1.21e+17	2.43e+18	2.18e+21
24	20	3	312	6552	20712	2.43e+18	5.10e+19	4.59e+22

Table 10: Number of invocations of the algorithms

To study H2 and H3, an ablation study with two different sets of *Content DIs* and *Adapter Capabilities DI* was performed. The first set did not allow absent properties, i.e., all the properties were provided. The second set only provides sme of these properties for the *Content DI* and *Adapter-*

Capabilities. Whenever incompleteness semantics (explained in Subsection 3.2 of Chapter 5) apply, they are used and the property is removed from the description documents.

Test		**With partial description**		**With total description**	
	C	**Time**	**Comparisons**	**Time**	**Comparisons**
1	4	173 ms	96	320 ms	245
2	4	106 ms	94	315 ms	230
3	4	108 ms	91	257 ms	201
4	4	85 ms	47	112 ms	97
5	4	106 ms	108	220 ms	340
6	4	177 ms	106	208 ms	340
7	4	184 ms	108	371 ms	340
8	7	260 ms	199	487 ms	483
9	9	1027 ms	3648	2012 ms	3834
10	11	1443 ms	1850	3456 ms	4304
11	11	964 ms	1013	2178 ms	3201
12	11	918 ms	1017	1678 ms	2278
13	11	1134 ms	1400	2976 ms	3023
14	13	7539 ms	14745	28678 ms	48675
15	13	8791 ms	16826	33457 ms	52567
16	13	4593 ms	10174	22331 ms	32870
17	13	6131 ms	13131	25312 ms	35322
18	13	6012 ms	14937	25240 ms	38731
19	16	7124 ms	21740	36593 ms	66457
20	19	8163 ms	8162	27964 ms	17586
21	19	9134 ms	9852	34574 ms	19356
22	19	10131 ms	18341	33420 ms	34132
23	19	11005 ms	19837	38432 ms	39962
24	20	10087 ms	20712	59570 ms	86793

Table 11: Time and number of comparisons with partial and with total description

In reference to H2, the demonstration that the size of the adaptation capabilities description decreases with partial description is straightforward. This is demonstrated by observing that the number of properties of the *Content DI* and *AdapterCapabilities* with total description must be longer. The number of properties with partial description *N* varies for each *ConversionCapabilities* element. Table 9 shows the number of properties for each of the *ConversionCapabilities* elements with partial description. The number of properties with total description N_{max}=36 is fixed and determined by the number of properties in the *Properties DI*. In order to check that this number decreases significantly, a significance test has been carried out (as described in Subsection 2.4 below).

In reference to H3, Table 11 shows the time and number of comparisons needed to execute the experiments both with partial and total description. As CAIN-21 allows for specifying the *AdapterCapabilities* to be considered during the decision phase (and therefore the corresponding *ConversionCapabilities*), the number of *ConversionCapabilities* elements *C* is not fixed through the tests. Subsection 2.4 proves that the time and number of comparisons also decrease significantly with partial description.

2.4 Statistical significance of the experiments

In statistics, a result is called *statistically significant* if it is unlikely to have occurred by chance. This subsection studies the statistical significance for the reduction in the average-case computational cost (H1), size of the partial description (H2), and time needed to compute the decision with partial description (H3).

The average-case number of properties compared in each experiment is a statistical variable that depends on the number of *ConversionCapabilities* elements *C* involved in the experiment. If we increase *C* for a given experiment, the number of properties to be compared would also increase. Therefore, this study assumes a normal distribution in this statistical variable only when the adaptation tests have the same *C* number. Formally, the number of comparisons in the theoretical-worst case is not a statistical variable but a fixed upper bound for the given *N* and *C*. Specifically, given *N* and *C,* its mean is the worst-case upper bound and its variance is zero.

The other statistical variables in this study (i.e., the average-case number of invocations of Alg. 2, the average-case number of invocations of Alg. 6, the time needed to execute Alg. 2 with partial description and with total description) can be understood as different measures of how the average-case number of comparisons varies with respect to the number of available properties. To demonstrate that these statistical variables are aligned with the average-case number of property comparisons, their correlations were calculated. The correlation coefficient between the number of invocations of Alg. 2 and the number of property comparisons is 0.987. The correlation coefficient between the number of invocations of Alg. 6 and the number of property comparisons is 0.990. The correlation coefficient between the time needed to compute the plan with partial description and the number of property comparisons is 0.931. The correlation coefficient between the time needed to compute the plan with total description and the number of property comparisons is 0.945. For these reasons, when the adaptation tests have the same *C* number, these experiments also assume a normal distribution for these statistical variables.

On the other hand, as the samples come from different adaptation tests, these experiments assume independence between the samples of the independent variables to be compared in the significance tests. Specifically, the experiments assume independence 1) among the samples of the average-case computational cost and the samples of the theoretical worst-case computational cost (H1), 2) among the samples of the size of the adaptation capabilities with partial and with total description (H2), and 3) among the samples of the decision time with partial and with total description (H3).

In addition, as these experiments have a relatively small number of samples, the experiments are going to validate these hypotheses by testing the difference between the two independent variable means using the Student's *t*-test. In this significance test, the *t*-score (*t*) is calculated as:

$$t = \frac{m_1 - m_2}{SE} \qquad (1)$$

Where m_1 is the mean of the first independent variable, m_2 is the mean of the second independent variable, and *SE* is the *standard error*, which is calculated as:

$$SE = \sqrt{\frac{v_1}{n_1} + \frac{v_2}{n_2}} \qquad (2)$$

In formula (2), n_1, v_1 are the number of tests and variance in the first independent variable, and n_2, m_2 are the number of tests and variation of the second independent variable.

For the three tests, the experiments define two alternative hypotheses: the null hypothesis is that $m_1 \geq m_2$ and the alternative hypothesis is that $m_1 < m_2$. The critical value of t (t_c) depends on the significance level (which is always 0.05 in this study) and on the *degree of freedom* (DF) of the adaptation tests.

2.4.1 Computational cost

To study the reduction in the computational cost for H1, the tests have been divided into four groups with the same *C* number in Table 8. In this way, as has been discussed above, we can assume that the independent variables (i.e., the average-case and worst-case number of invocations) in each *group of tests* follow normal distributions. Specifically, we have selected four groups of tests corresponding to *C*=4, *C*=11, *C*=13, *C*=19. The other tests (i.e., tests with *C*=7, *C*=9, *C*=16, *C*=20) were discarded because there is only one test per group and therefore these tests have zero *degrees of freedom* (*DF*). For the selected four groups, the means to be compared are the average-case computational cost m_1 and the worst-case computational cost m_2. To analyze the number of invocations of Alg. 2, Table 12 collects the means, variances, standard errors, degrees of freedom used to calculate the t critical value, t-score and t critical value for the independent variables in each group. Likewise, Table 13 and Table 14 show the same information for the number of invocations of Alg. 6 and for the number of property comparisons, respectively. The worst-case computational cost is the second independent variable. As the second independent variable is a theoretical upper value, its mean m_2 is constant for each group and therefore v_2=0.

Group	n_1=n_2	m_1	m_2	v_1	v_2	*SE*	*DF*	*t*	t_c
C=4	7	3.71	2.4e+01	0.57	0.00	0.28	6	-7.10e+01	-1.94
C=11	4	36.00	3.99e+07	144.00	0.00	6.00	3	-6.65e+06	-2.35
C=13	5	238.80	6.23e+09	2569.20	0.00	22.66	4	-2.74e+08	-2.13
C=19	4	245.75	1.21e+17	7206.91	0.00	42.44	3	-2.85e+15	-2.35

Table 12: Significance test for the number of invocations of Alg. 2

Group	n_1=n_2	m_1	m_2	v_1	v_2	*SE*	*DF*	*t*	t_c
C=4	7	20.28	1.20e+02	24.57	0.00	1.87	6	-5.32e+01	-1.94
C=11	4	417.50	4.79e+08	13225.00	0.00	57.50	3	-8.33e+06	-2.35
C=13	5	3702.00	8.71e+08	412141.50	0.00	287.10	4	-3.03e+06	-2.13
C=19	4	4366.50	2.43e+18	2408232.33	0.00	775.92	3	-3.13e+15	-2.35

Table 13: Significance test for the number of invocations of Alg. 6

Group	n_1=n_2	m_1	m_2	v_1	v_2	*SE*	*DF*	*t*	t_c
C=4	7	92.85	1.08e+05	4.58e+02	0.00	8.09	6	-1.33e+04	-1.94
C=11	4	1320.00	4.31e+11	1.57e+05	0.00	198.61	3	-2.17e+09	-2.35
C=13	5	13962.60	7.84e+11	6.20e+06	0.00	1113.68	4	-7.03e+08	-2.13
C=19	4	14199.00	2.18e+21	3.31e+07	0.00	2878.05	3	-7.57e+17	-2.35

Table 14: Significance test for the number of comparisons between properties

In all cases, the null hypothesis is rejected because the t-score is higher than the t critical value (i.e., the t-score is in the region of rejection). Hence, this research concludes that the average-case computational cost of the *Planner* is significantly lower than the theoretical worst-case computational cost.

2.4.2 Size of the descriptions

To study H2, the first independent variable is the number of properties in the *ConversionCapabilities* elements with partial description *N* (gathered in Table 9) and the second independent variable is the number of properties in the *ConversionCapabilities* elements with total description (assuming N_{max}=36 as explained in Subsection 2.3). We compare the means of the first independent variable m_1, with the means of the second independent variable m_2. In this significance test, it is assumed that the number of properties in the *ConversionCapabilities* elements follows a normal distribution and thus Table 15 has only one group of tests. The number of properties with total description m_2 is constant (m_2= N_{max}=36) and therefore v_2=0.

Group	n_1=n_2	m_1	m_2	v_1	v_2	*SE*	*DF*	*t*	t_c
All	22	20.45	36.00	11.21	0.00	0.71	21	-21.77	-1.72

Table 15: Significance test for H2

The null hypothesis is rejected because the *t*-score is higher than the *t* critical value. Thus, it can be concluded that the size of the *ConversionCapabilities* elements decreases significantly when partial description is allowed.

2.4.3 Decision time and number of comparisons

This subsection studies the significant reduction in the decision time for partial description that H3 hypothesizes. Table 11 gathers the decision time for the tests with partial and with total description. Again, to assume a normal distribution for the independent variables of each group of tests we have created four groups of tests corresponding to *C*=4, *C*=11, *C*=13, *C*=19 in Table 8. Table 16 shows the result of computing the *t*-score for the decision time with partial and with total description. In contrast to the previous experiments, in this experiment the second independent variable is not theoretical, and thus v_2>0. It can be observed that in all cases, the *t* critical value is higher than the *t*-score, and therefore we conclude that the decision time decreases significantly when partial description is allowed.

Group	n_1=n_2	m_1	m_2	v_1	v_2	*SE*	*DF*	*t*	t_c
C=4	7	134.14	2.57e+02	1.75e+03	75.20e+02	36.39	6	-3.39	-1.94
C=11	4	939.75	2.57e+03	1.81e+05	6.33e+05	451.29	3	-3.61	-2.35
C=13	5	6613.20	2.70e+04	2.56e+06	1.80e+07	2031.37	4	-10.04	-2.13
C=19	4	9608.25	3.36e+04	1.51e+06	1.86e+07	2247.80	3	-10.67	-2.35

Table 16: Significance test for the decision time with partial and with total description

Group	n_1=n_2	m_1	m_2	v_1	v_2	*SE*	*DF*	*t*	t_c
C=4	7	92.85	2.56e+02	4.58e+02	8.37e+03	35.52	6	-4.59	-1.94
C=11	4	1320.00	3.20e+03	1.57e+05	7.00e+05	463.10	3	-4.06	-2.35
C=13	5	13962.60	4.16e+04	6.20e+06	7.35e+07	3993.67	4	-6.92	-2.13
C=19	4	14048.00	2.78e+04	3.47e+07	1.21e+08	6243.81	3	-2.19	-2.35

Table 17: Significance test for the number of comparisons with partial and with total description

Even though H3 only states that the decision time decreases significantly, in Table 17 collects the number of comparisons to prove that the number of comparisons also decreases significantly. In this way, this research aims to demonstrate that the matching process (which executes the property comparisons) is the more costly process in our *Planner*. To provide more evidence for the alignment of the decision time and the number of comparisons, the correlation coefficients between the time and number of comparisons in Table 11 have also computed, which are 0.944 with partial description and 0.901 with total description. This result can be intuitively interpreted by considering that the property comparison is the inner operation and therefore, the operation that is repeated more often.

2.5 Analysis of the results

H1 states that in practical scenarios the average-case computational cost of the *Planner* algorithm is significantly lower than the theoretical worst-case computational cost. Subsection 2.4.1 has confirmed the significant difference that H1 proposes. The intuition behind these results is that Alg. 2 only succeeds in expanding a few conversion states (the feasible conversion states according to Alg. 4). Besides, in practice, most of the properties are optional and rarely used. To empirically see it, we can observe that when *C* increases, the ratio between the theoretical worst-case number of invocation of the algorithms and the real number of invocations also increases. The correlation coefficient between *C* and this ratio is 0.704 for Alg. 6 and 0.666 for Alg. 2. This correlation effect can also be observed in Table 12, Table 13 and Table 14 where when *C* increases the *t*-score increases as well. This suggests that H1 becomes more important with large values of *C*. The reason behind this observation is that with a large number of conversion capabilities elements, Alg. 4 eliminates a larger number of expansions in the virtual tree of conversions.

H2 states that a partial description of the conversion modules has the benefits of reducing the size of the adaptation capabilities and H3 states that partial description also significantly reduces the decision time. To study H2 and H3 an ablation study has been performed. The first group uses a partial description of the *ConversionCapabilities* and the second group uses a total description of the *ConversionCapabilities*. To prove H2, Table 9 collects from the CAIN-21 software the number of properties for each *ConversionCapabilities* description element with partial description. It is obvious that a partial description is smaller than a total description. Subsection 2.4.2 validates H2 because the size of the conversions decreases significantly. To prove H3, Table 11 collects two groups of similar tests that only differ in the *Content DI* to be adapted and the *AdapterCapabilities* description. Table 11 shows both the time and number of comparisons performed. Subsection 2.4.3 demonstrates that in all cases the time and number of comparisons decrease significantly when partial description was allowed. These results validate H3 and show the usefulness of partial description.

2.6 Comparison

In [56], Berhe et al. report experiments in which the planner selects and performs multi-step image adaptation. They do not provide a dataset, but instead report the times needed to construct the DAG, which determines the multi-step sequence. The behaviour of the *Planner* algorithm can be compared with the behaviour of the planning algorithm in [56]. The *Planner* builds a virtual tree of conversions whereas [56] has demonstrated the construction of a DAG that is similar. The main difference is that in the virtual tree of conversions (as explained in Subsection 2.3 of Chapter 5) the *Planner* creates several instances of the conversion state representing the initial *Component*.

Fig. 23 compares the number of conversion capabilities evaluated (i.e., *C* in Table 8) against the time needed to compute the plan for both partial and total description in Table 11. In the *Planner*, both curves increase rapidly, although the decision time with total description is significantly higher, most notably when $C \geq 11$.

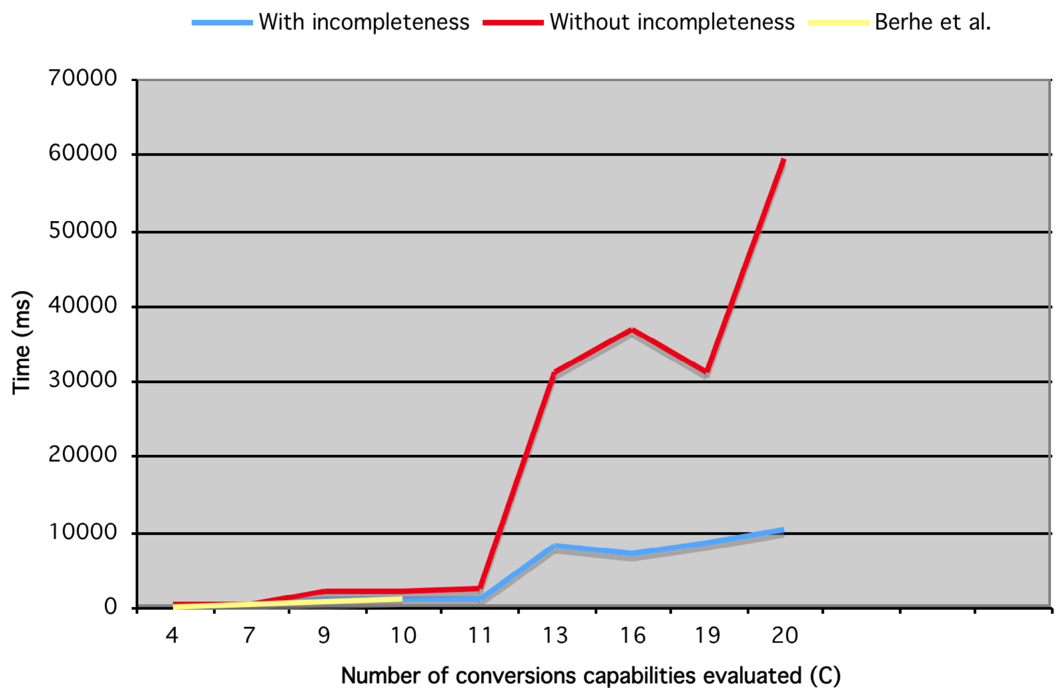


Fig. 23: Plan construction time with partial and with total description

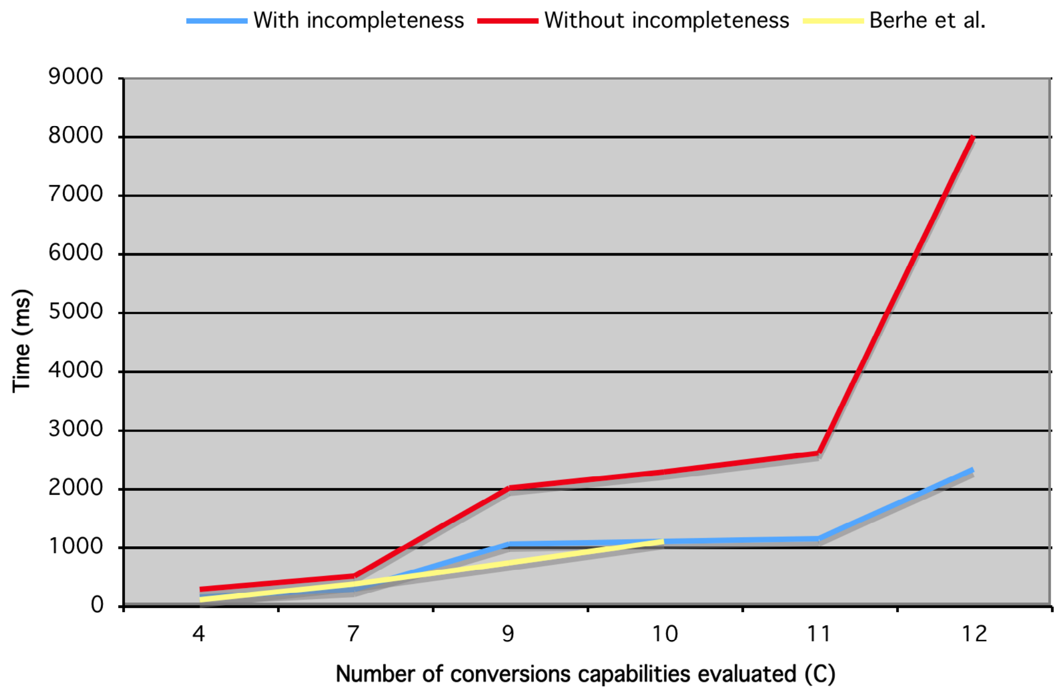


Fig. 24: Plan construction time with $C≤11$

Given that the *Planner's* curves increase so rapidly, in Fig. 23 it is not easy to compare the results of the *Planner* with the results of Berhe et al. [56]. They have reported results with only $C≤10$, therefore, to accomplish this comparison, Fig. 24 shows the *Planner* results with $C \leq 11$. The main difference between our *Planner* and their planner is that in our implementation, the computation time substantially depends on the number of conversion capabilities elements C involved in the decision. Conversely, their computation time almost depends exclusively on the length of the sequence obtained (i.e., their curve increases linearly) and not on the number of conversion capabilities (services in their terminology). Furthermore, they state, "*It was also observed that the progress both for the depth and the width (number of services per transformation) was almost constant with average increase of 40 ms for each depth and 10 ms for each 10 additional services. This implies that having several services per transformation does not [greatly] affect [~~much on~~] the total construction time.*"

In our understanding, this conclusion is incorrect, because, in order to obtain a complete plan (i.e., analyze all the feasible solutions), the algorithm has to compare each sequence with all the states that result from instantating the conversion capabilities. This is why in our results the computational time is increasing rapidly when C increases.

3 Static decisions

Section 3 of Chapter 6 described how the *Planner* makes static decisions after the set of sequences of conversions has been obtained and before the control is transferred to the *Executer*. This section describes several adaptation tests to demonstrate how these decisions are accomplished. In addition, this section describes how to represent in the MPEG-21 framework the adaptation engine preference model described in Section 3.2 of Chapter 6.

3.1 Best-sequence-based decisions

This subsection describes two didactic adaptation tests to show the trade-offs that sometimes occur among the number of steps, execution cost and content degradation for different given feasible sequences of conversions.

3.1.1 Methodology, dataset and metrics

The following demonstrations show how to adapt a *Content DI* to a terminal labelled as *mpeg2_medium_desktop*, which full description is provided in Appendix A. The *Content DI* to be adapted is named *he_asked_di.xml* and its full description is provided in Listing 20 of Appendix A. This *Content DI* uses in the visual stream the scalable extension for the H.264/AVC media format and in the audio stream the AAC media format. Specifically, the visual stream is encoded using scalable format at three spatial resolutions (namely, QCIF: 176x144 pixels, CIF: 352x288 pixels, 4CIF: 704x576 pixels) and frame rates of 30, 15, 7.5 and 3.75 fps. The visual stream was encoded with one base layer and up to two quality enhancement layers using the *Joint Scalable Video Model* (JSVM) 9.13.1 codec[23].

During these demonstrations, the *Planner* selects several conversion modules. Table 18 presents the estimated execution costs and content degradations for each conversion module (this values are stored in the corresponding *ConversionCapabilities* element). Subsection 3.1 of Chapter 6 indicates that these values are estimated values that the *Adapter*'s implementer provides. In the

[23] JSVM is publicly available at *http://ip.hhi.de*

following demonstrations, we have selected a set of values that lets us exemplify how the decision mechanism works[24].

Conversion module	Estimated execution cost	Estimated content degradation
svc_without_audio_transcoder	0.53	0.50
svc_with_audio_transcoder	0.62	0
svc_to_mp4	7.01	0.02
mp4_to_mpeg2	1.97	0.02
svc_to_yuv	3.42	0
yuv_to_mpeg2	1.59	0.01

Table 18: Estimated execution cost and content degradation of the conversion modules

In Table 18, the *svc_without_audio_transcoder* conversion tool is labelled with 0.5 for the content degradation indicating that 50% of the information is lost, as this conversion module drops the audio stream, but retains the visual stream. The *svc_with_audio_transcoder* and *svc_to_yuv* conversion tools are labelled with 0 for the content degradation, as they retain the existing streams. Note that these conversion tools select a layer in the scalable video, and therefore the content is somewhat degraded with respect to the original video. However, our *Planner* does not consider this kind of content degradation but it transfers this decision to the *Adapter* (see Subsection 4.1 of this chapter). The *svc_to_yuv* conversion module is also labelled with 0 for the content degradation because this conversion module decodes the compressed video, and thus it does not lose information. Let us consider that, for our example, the *Adapter's* implementer has used an objective-quality metric and annotated the *yuv_to_mpeg2* conversion module with 0.01 for the content degradation. This number is indicating that 1% of information is lost during the compression process. In the same way, the *svc_to_mp4* and *mp4_to_mpeg2* conversion tools may also imply a small content degradation, as both tools transcode the video (involving decoding and re-encoding). For this demonstration, we assume that the *Adapter's* implementer has measured the degradation and obtained 0.02 for the content degradation of these conversion modules.

3.1.2 Best-sequence-based decisions and analysis of the results

This subsection focuses on the best-sequence-based decisions for the two adaptation tests introduced above.

Demonstration 1

This demonstration involves the adaptation of the *Content DI* named *he_asked_di.xml* to the terminal labelled as *mpeg2_medium_desktop*. For this adaptation, the *Planner* finds two feasible sequences of conversions:

initial → svc_without_audio_transcoder → svc_to_mp4 → mp4_to_mpeg2 → goal
initial → svc_with_audio_transcoder → svc_to_mp4 → mp4_to_mpeg2 → goal

There is no shorter sequence because CAIN-21 does not incorporate any direct *svc_to_mpeg2* conversion tool. Therefore, the visual stream has to be transcoded several times in order to perform the adaptation to the *mpeg2_medium_desktop* terminal. According to Table 18, the estimated execution costs and content degradations are 0.53+7.01+1.97=9.51 and

[24] Subsection 3 of Chapter 8 proposes that, in future work, the mechanism to obtain these values can be further investigated.

0.5+0.02+0.02=0.54, respectively, for the first sequence and 0.62+7.01+1.97=9.6 and 0+0.02+0.02=0.04, respectively, for the second sequence. As the *fast_adaptation* preference is not provided, the *Planner* selects the second sequence, which has a lower content degradation. If the *fast_adaptation* preference were provided, the first sequence would be selected.

This demonstration is consistent with what the user expects and shows how the above estimated values help in choosing the adaptation that best fits the user's preferences. In this demonstration, it should also be noted that if the user chooses the *fast_adaptation* preference the content will be obtained early but the degradation will be higher.

Demonstration 2

For this demonstration, we have used an additional *Adapter* named *YUVAdapter*. The *YUVAdapter* includes two additional conversion modules, named *svc_to_yuv* and *yuv_to_mp2*. Table 18 shows the estimated execution costs and content degradations of these conversion tools. After executing the previous adaptation test with this new *Adapter*, the *Planner* identifies three sequences of conversions:

initial → svc_without_audio_transcoder → svc_to_mp4 → mp4_to_mpeg2 → goal
initial → svc_with_audio_transcoder → svc_to_mp4 → mp4_to_mpeg2 → goal
initial → svc_with_audio_transcoder → svc_to_yuv → yuv_to_mpeg2 → goal

The estimated execution costs are 9.51, 9.6 and 5.63, respectively. The estimated content degradations are 0.54, 0.04 and 0.01, respectively. Note that in this case, the third sequence has both lower execution cost and lower content degradation. This means that regardless of the user's preferences, the third sequence will be selected. This adaptation test demonstrates how the addition of new *Adapters* to CAIN-21 improves both the operational performance and the quality of the adapted content.

3.2 Best-outcome-based decisions

This section describes how to represent in the MPEG-21 framework the preferences of different types of users. The demonstration uses the six preferences introduced in Table 6 and Table 7 of Chapter 6. Once these preferences are collected, different best-outcome-based decision methods can be executed. Subsection 3.2 of Chapter 6 described how to use a classical preference-graph in order to make this best-outcome decision.

3.2.1 Selected preferences

Subsection 7.4 of Chapter 2 indicates that often users may not be able to provide much more than a qualitative ranking (e.g., "I know that I prefer video than audio but I'm not sure about such utility numbers that you are asking for.") of circumscribed outcomes (e.g., "I prefer to listen that podcast about technology."). For these reasons, the preference model developed in this work uses ranking (instead of a more complex description model, such as the utility functions) to elicit and represent the user's preferences.

Subsection 3.2.3 of Chapter 6 analyzed the standard set of user preferences that MPEG-21 Part 7 have proposed. This section accomplishes the integration in the MPEG-21 framework of the six preferences introduced in Table 6 and Table 7 of Chapter 6.

With respect to the media format preferences, we have extended the existing *mpeg7:MediaFormat*, *mpeg7:VisualCoding* and *mpeg7:AudioCoding* descriptors with the inclusion of a new attribute named *preferenceValue*. The XML Schema with this extension can be found in Appendix A. The attribute allows the user to assign a utility value to the existing MPEG-7 description variables and a ranking of the list of values of these variables (see Subsections 7.1 and 7.2 of Chapter 2). Table 7 in Chapter 6 shows the variables of this kind in the preference model of CAIN-21.

With respect to the adaptation preferences, CAIN-21 defines new adaptation schedule preferences (see Table 6 in Chapter 6) that do not have counterparts in the MPEG-7 Part 5 descriptors. These preferences are located in a new element named *cde:AdaptationPreferences* within the *cde:FilteringAndSearchPreferencesType* description tool (see Listing 12).

As explained in Subsection 3.2.1 of Chapter 6, the schedule preferences have an implicit and hard-coded ranking in their list of values. Conversely, the media format preferences variables in Table 7 in Chapter 6 have an explicit ranking. However, in both cases the user can provide a utility for each variable.

3.2.2 Representation of the preferences

Listing 12 shows an example in which two *mpeg21:UserType* description tools are used to describe preferences according to our proposed adaptation engine preference model. The first one with *id*="*reporter*" contains preferences for this group of users. In this case, the users have provided preferences regarding the file format, visual stream format and audio stream format, but they have said nothing about their preferences for the content degradation, online adaptation and number of conversions. The second user has the *id*="*cain21*" and stands for the adaptation engine itself. This special user acts as a default preferences container and provides preference utility for all the feasible preference variables under consideration along with a preference ranking for their values. To this end, we have created the cde: *cde:UsagePreferencesType* and *cde:FilteringAndSearchPreferencesType* description tools, which are extensions of the *mpeg21:UsagePreferencesType* and *mpeg7:FilteringAndSearchPreferences* description tools. The XML Schema with this extension can be found in Appendix A. This new description tool enables the description of the utilities and preference rankings that appear in Table 6 and Table 7 of Chapter 6.

```
<!-- Users description -->
  <UsageEnvironmentProperty xsi:type="cde:UsersType">

<User id="reporter" xsi:type="UserType">
    <UserCharacteristic xsi:type="UsagePreferencesType">
     <UsagePreferences>
      <mpeg7:FilteringAndSearchPreferences
         xsi:type="cde:FilteringAndSearchPreferencesType">
       <cde:SourcePreferences>
        <cde:MediaFormat preferenceValue="50">
         <mpeg7:Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:2">
          <mpeg7:Name>Audiovisual</mpeg7:Name>
        </mpeg7:Content>
         <cde:FileFormat href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:5"
                         preferenceValue="50">
          <cde:Name>MPEG-4 file format</cde:Name>
         </cde:FileFormat>
         <cde:FileFormat href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:3"
                         preferenceValue="0">
          <cde:Name>MPEG file format</cde:Name>
         </cde:FileFormat>
         <cde:FileFormat href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:7"
                         preferenceValue="-50">
          <cde:Name>Audio video interleave format</cde:Name>
         </cde:FileFormat>
         <cde:VisualCoding preferenceValue="30">
          <cde:Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.1"
                      preferenceValue="50">
           <cde:Name>MPEG-4 Visual Simple Profile</cde:Name>
          </cde:Format>
          <cde:Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:2.2.2"
                      preferenceValue="0">
           <cde:Name>MPEG-2 Video Main Profile @ Main Level</cde:Name>
          </cde:Format>
         </cde:VisualCoding>
         <cde:AudioCoding preferenceValue="30">
          <cde:Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:6"
                      preferenceValue="50">
           <cde:Name>AMR</cde:Name>
          </cde:Format>
          <cde:Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.3"
                      preferenceValue="0">
           <cde:Name>MPEG-2 Audio AAC</cde:Name>
          </cde:Format>
         </cde:AudioCoding>
        </cde:MediaFormat>
       </cde:SourcePreferences>
      </mpeg7:FilteringAndSearchPreferences>
     </UsagePreferences>
    </UserCharacteristic>
   </User>

   <User id="cain21" xsi:type="UserType">
    <UserCharacteristic xsi:type="UsagePreferencesType">
     <UsagePreferences>
      <mpeg7:FilteringAndSearchPreferences
         xsi:type="cde:FilteringAndSearchPreferencesType">
       <cde:SourcePreferences>
        <cde:MediaFormat preferenceValue="50">
         <mpeg7:Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:2">
          <mpeg7:Name>Audiovisual</mpeg7:Name>
        </mpeg7:Content>
         <cde:FileFormat href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:3"
                         preferenceValue="50">
          <cde:Name>MPEG file format</cde:Name>
         </cde:FileFormat>
         <cde:FileFormat href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:7"
                         preferenceValue="0">
          <cde:Name>Audio video interleave format</cde:Name>
         </cde:FileFormat>
         <cde:FileFormat href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:5"
                         preferenceValue="-50">
          <cde:Name>MPEG-4 file format</cde:Name>
         </cde:FileFormat>
         <cde:VisualCoding preferenceValue="30">
          <cde:Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:2.2.2"
                      preferenceValue="50">
           <cde:Name>MPEG-2 Video Main Profile @ Main Level</cde:Name>
          </cde:Format>
          <cde:Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.1"
                      preferenceValue="0">
           <cde:Name>MPEG-4 Visual Simple Profile</cde:Name>
          </cde:Format>
          <cde:Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.2"
                      preferenceValue="-10">
           <cde:Name>MPEG-4 Visual Advanced Simple Profile</cde:Name>
          </cde:Format>
         </cde:VisualCoding>
         <cde:AudioCoding preferenceValue="30">
          <cde:Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.3"
                      preferenceValue="50">
           <cde:Name>MPEG-2 Audio AAC</cde:Name>
          </cde:Format>
          <cde:Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:6"
                      preferenceValue="0">
           <cde:Name>AMR</cde:Name>
          </cde:Format>
         </cde:AudioCoding>
        </cde:MediaFormat>
       </cde:SourcePreferences>
       <cde:AdaptationPreferences>
        <cde:ContentDegradation preferenceValue="80"/>
        <cde:Online preferenceValue="50"/>
        <cde:ExecutionCost preferenceValue="30"/>
       </cde:AdaptationPreferences>
      </mpeg7:FilteringAndSearchPreferences>
     </UsagePreferences>
    </UserCharacteristic>
   </User>
  </UsageEnvironmentProperty>
```

Listing 12: Description of the preferences into the MPEG-21 UED

Subsection 2.6 of Chapter 6 has explained why forcing the user to choose all the preferences is not, in general, a good idea. To address this issue, the *cde:FilteringAndSearchPreferencesType* description tool permits the existence of incomplete preferences. Listing 12 shows how incomplete preferences can be managed by means of a constraint hierarchy in which the adaptation engine (i.e., *id*="*cain21*") provides values for the preferences that the user does not supply. As adaptation engine preferences provide values for all the preferences, the fallback mechanism guarantees unique and pre-determined results.

4 Dynamic decisions

The user preferences can be taken into account at two modules: in the *Planner* before the adaptation process starts or in the *Adapters* during the execution of the adaptation. If the *Planner* makes a decision the multi-valued parameters are single-valued and the *Adapters* loses its opportunity to make this decision. If there are no user preferences then the *Planner* transfers decisions to the *Adapter*. For example, if the user provides his or her preferences regarding the audio format then the *Planner* will select this audio format. Otherwise, the *Adapter* has the opportunity to select the audio format to produce, among the multi-valued audio format property that it receives from the *Planner*.

It follows from the above discussion that the *Planner* is the module responsible for making static decisions and that the *Adapters* are the modules responsible for making the dynamic decisions. This subsection provides several demonstrations of how the *Adapters* make utility-based (i.e., quality-based and semantic-based) dynamic decisions.

4.1 Quality-based decisions

Currently, CAIN-21 includes four *Adapters* that perform dynamic decisions whenever the target parameters are not provided or are multi-valued: *ImageTrancoderAdapter*, *AudioTranscoder-Adapter*, *OnDemandVideoTranscoderAdapter* and *SVCAdapter*. To decide the output parameters, the *ImageTrancoderAdapter* relies on the *imagemagick* software tool, whereas the *AudioTranscoderAdapter* and *OnDemandVideoTranscoderAdapter* rely on the *ffmpeg* software tools. The rules that these software tools follow in choosing the output parameters are not clearly defined. In fact, these rules are just hard-coded in the source code and created in an ad-hoc manner. Furthermore, often the criteria these tools follow are not intended to maximize the quality of the adaptation, but to reach a good level of compression (which the software tools implementers may consider has a "good enough" utility). This effect can be observed, for example, when *ffmpeg* does not receive the *qmin* and *qmax* parameters.

On the contrary, the *SVCAdapter* relies on the JSVM codec, in which the criteria for choosing the output parameters have been clearly defined. For this reason, this subsection concentrates on demonstrating how scalable video adaption is carried out inside the *SVCAdapter*. The theoretical basis for these demonstrations has been laid out in Subsection 4.1 of Chapter 6.

4.1.1 Methodology, dataset and metrics

In the following demonstration, the methodology, dataset and metrics are similar to the ones described in Subsection 3.1.1. The only difference is that the following demonstration uses the *Content DIs* named *football_di.xml* and the terminal labelled as *svc_without_audio_352x288_15fps*.

4.1.2 Quality-based decisions and analysis of the results

Demonstration 3

This demonstration involves the adaptation of the *Content DI* named *football_di.xml* to the terminal labelled as *svc_without_audio_352x288_15fps*. Table 19[25] shows the frame size, frame rate, bit rate and quality metrics of the layers labelled in the *AdaptationQoS* description of the *Content D*I.

The multi-valued parameters that the *Planner* transfers to the *SVCAdapter* during this execution are:

Source parameters:	**Target parameters:**
visual_frame_size = {176x144, 352x288}	*visual_frame_size* = {176x144, 352x288}
visual_frame_rate = {3.75, 7.5, 15, 30}	*visual_frame_rate* = {3.75, 7.5, 15}
visual_bitrate = [38.4 .. 910]	*visual_bitrate* = [38.4 .. 500]

In this adaptation test the *Planner* has constrained the *visual_frame_rate* to 15 fps and the *visual_bitrate* to the range [38.4 .. 500] kbps. These parameters have been selected because the *svc_without_audio_352x288_15fps* terminal supports up to 15 fps and up to 500 kbps.

[25] The ILab of the University of Surrey, Guildford, UK has calculated the values in Table 19. I thank S. Dogan, G. Nur, M. Mrak, H. K. Arachchi for calculating this helpful table.

During the execution, the *SVCAdapter* determines which layer, meeting the target parameter restrictions, has higher quality. In our demonstration, the layers marked in bold in Table 19 (i.e., Layers 0, 1, 2, 4, 5, 6, 7, 8, 9, 12, 13, 14, 16, 17, 18) meet these restrictions. In particular, the layers with higher PSNR or SSIM scores have a higher quality from the standpoint of these metrics. Conversely, the layers with lower VQM scores have a higher quality from the standpoint of these metrics. If the adaptation engine preferences provide PSNR as the preferred quality metric then Layer 5 will be selected because in this case this layer has the higher PSNR score. Conversely, if the adaptation engine preferences designate VQM or SSIM as the preferred quality metric then Layer 17 will be selected because in this case this layer has the higher score[26].

Layer	Frame size (pixels)	Frame rate (fps)	Quality layer	Bit rate (kbps)	PSNR (dB)	VQM	SSIM
0	**176x144**	**3.75**	**0**	**38.4**	**34.277**	**0.336**	**0.893**
1	**176x144**	**7.5**	**0**	**53.6**	**33.743**	**0.329**	**0.890**
2	**176x144**	**15**	**0**	**70.4**	**33.297**	**0.315**	**0.886**
3	176x144	30	0	88.1	33.090	0.295	0.884
4	**176x144**	**3.75**	**1**	**64.5**	**35.694**	**0.273**	**0.922**
5	**176x144**	**3.75**	**2**	**92.7**	**37.488**	**0.243**	**0.940**
6	**176x144**	**7.5**	**1**	**89.3**	**35.190**	**0.266**	**0.921**
7	**176x144**	**7.5**	**2**	**124.7**	**36.849**	**0.239**	**0.938**
8	**176x144**	**15**	**1**	**119.4**	**34.777**	**0.251**	**0.918**
9	**176x144**	**15**	**2**	**162.2**	**36.282**	**0.234**	**0.937**
10	176x144	30	1	153.7	34.562	0.243	0.917
11	176x144	30	2	203.9	35.927	0.231	0.935
12	**352x288**	**3.75**	**0**	**219.7**	**33.665**	**0.346**	**0.864**
13	**352x288**	**7.5**	**0**	**296.8**	**33.341**	**0.338**	**0.862**
14	**352x288**	**15**	**0**	**384.0**	**33.029**	**0.323**	**0.859**
15	352x288	30	0	482.3	32.830	0.311	0.857
16	**352x288**	**3.75**	**1**	**326.7**	**35.168**	**0.245**	**0.905**
17	**352x288**	**3.75**	**2**	**453.1**	**37.391**	**0.233**	**0.941**
18	**352x288**	**7.5**	**1**	**435.2**	**34.908**	**0.257**	**0.904**
19	352x288	7.5	2	583.2	36.954	0.229	0.943
20	352x288	15	1	565.3	34.627	0.226	0.903
21	352x288	15	2	735.0	36.517	0.219	0.939
22	352x288	30	1	717.8	34.416	0.218	0.901
23	352x288	30	2	910.0	36.151	0.204	0.937

Table 19: Quality analysis for the layers in *football_di.xml*

4.2 Semantic-based decisions

The semantic-based decisions access and analyse the content meaning to improve the user's experience. This subsection demonstrates two adaptation tests involving semantic-based adaptation. In addition, this subsection demonstrates the elements of CAIN-21 described in Chapter 3 and 4. Subsection 2.1 of Chapter 3 described the DI level adaptation interface. Both operations of this interface – i.e., *transform*() and *addVariation*() – have been used in the tests. To cover a wide

[26] In future work, instead of using the user or adaptation engine preferences, subjective metrics could be performed to decide which quality metrics maximise the user's experience.

range of multimedia adaptations, both images and videos have been selected for the demonstrations in this subsection.

4.2.1 Transforming an image to an small video terminal

Demonstration 4
This adaptation test illustrates how the *Content DI* in Listing 10 of Chapter 6 is adapted to the *id="iphone"* video terminal (shown in Listing 9 of Chapter 4). In this *Content DI* the faces of the people in the image where annotated using ROIs. The full description of these elements is available in Appendix A.

When the adaptation starts, the *transform*() software interface receives a *Configuration DI* to indicate the target terminal. The *Parser* module uses the *Properties DI* to gather the properties of the *Content DI*, *Adapter Capabilities DIs* and *Usage Environment DI*. After that, the *Planner* module produces the following sequence of conversions *initial* → *image_transcoder* → *small_image_2_video* → *ondemand_mp4_transcoder* → *goal*. In this sequence, *initial* represents the properties of the original *Content DI*. The *Preconditions* and *Postconditions* of *image_transcoder*, *image_2_video* and *ondemand_mp4_transcoder* are described in their corresponding *ConversionCapabilities* elements. Lastly, *goal* represents the properties of the adapted content.

The *image_transcoder* conversion module transcodes the image format and size to the preconditions of the *image_2_video* conversion (i.e., JPEG image format and 3:4 aspect ratio). The *image_2_video* conversion accepts only JPEG images and produces only MPEG-2 video. *The ondemand_mp4_transcoder* (whose conversion capabilities appear in Listing 25 Appendix A) transcodes the MPEG-2 video to the constraints of the *"iphone"* terminal (3GPP according to Listing 9 of Chapter 4).

The *Image2VideoAdapter* implements the *small_image_2_video* conversion module that performs the semantic adaptation from image to video. Specifically, it takes into account the ROIs. The result of adapting a *Content DI* with ROIs is more informative since it focuses on showing the ROIs (i.e., faces on the image), instead of scrolling through the whole image (what happens if the ROIs are not provided).

This *Image2VideoAdapter* implements several conversion modules named *big_image_2_video*, *medium_image_2_video*, *small_image_2_video*. Section 7 of Chapter 5 justifies the existence of different profiles in order to address the limitation of the *AdapterCapabilities* declarative approach. These profiles speed up the adaptation process with costly adaptation operations such as the *Image2VideoAdapter*. In this demonstration, the terminal has a small spatial resolution and the image to be adapted is much more bigger. That is the reason why, in this demonstration the *image_transcoder* conversion module (implemented inside the *ImageTranscoderAdapter*) scales down the image before executing the *Image2VideoAdapter*. This execution order significantly reduces the adaptation time.

Demonstration 5
If we change the terminal in *Demonstration 1* from *"iphone"* to *"http_nokia_n95*", we obtain another didactic adaptation test. This test fully demonstrates the usefulness of the binding modes. In this demonstration, the *Planner* produces a sequence with four conversion modules *initial* → *image_transcoder* → *image_2_video* → *ondemand_mp4_transdoder* → *http_delivering* → *goal*.

Note that the *Planner* has added at the end of the sequence the *http_delivering* conversion module to change the binding property from *FILE* to *HTTP*. In *Demonstration 4* the *"iphone"* terminal supported the *FILE* delivery mechanism (see Listing 9 of Chapter 4), which corresponds to the binding property at the output of *ondemand_mp4_transcoder*. Therefore, the *Planner* did not add the *http_delivering* conversion at the end of the sequence. However, in *Demonstration 5*, the *"http_nokia_n95"* terminal only supports the *HTTP* binding mode, and so the *Planner* has added the *http_delivering* conversion at the end of the sequence. This conversion has the *FILE* binding mode in its preconditions and the *HTTP* binding mode in its postconditions. This indicates that the purpose of this module is to transfer forward the input file using the *HTTP* standard protocol.

4.2.2 Summarizing variations of video news items

The following demonstration shows how video adaptation can be effectively combined with video summarization. This demonstration has been extracted from the MESH European project, whose results where also reported in [112]. We only provide details of the adaptation phase, if you are interested in the other phases of this process (i.e., video analysis or video summarization), please refer to [112].

Demonstration 6

This demonstration summarizes a *Content DI* containing a news item to three feasible terminals. Listing 13 shows the more important parts of the original *Content DI* to be adapted. The standard *mpeg7:ClassificationType* description tool indicates the genre of the news item. This *Content DI* is fully provided in Listing 18 of Appendix A.

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

  <!-- Classification -->
  <Descriptor>
   <Statement mimeType="text/xml">
    <Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2004">
     <DescriptionUnit xsi:type="ClassificationType">
      <Genre href="urn:mpeg:mpeg7:cs:ContentCS:2001:1.1.13">
       <Name xml:lang="en">Natural disasters</Name>
      </Genre>
      <Genre href="urn:mpeg:mpeg7:cs:ContentCS:2001:1.5.1">
       <Name xml:lang="en">Political</Name>
      </Genre>
     </DescriptionUnit>
    </Mpeg7>
   </Statement>
  </Descriptor>
  <!-- Original content -->
  <Component xsi:type="cdi:VideoComponentType" id="original">
   <Descriptor xsi:type="cdi:Mpeg7DescriptorType">
    <!-- MPEG-7 MediaDescriptionType describing the resource -->
    ...............
   </Description>
   <Resource mimeType="video/mpeg" ref="../mesh/didl/mpeg2video.mpg"/>
  </Component>
 </Item>
</DIDL>
```

Listing 13: Original *Content DI* to be summarized and adapted

In Listing 13, the MPEG-7 *ClassificationType* description tool indicates that the news item contains natural disaster and political content. The video is stored in a *Component* element with *id="original"*. This *Component* contains an MPEG-7 *MediaDescriptionType* description tool for the *Resource* element. The original video is MPEG-1 video and has a resolution of 720x576 pixels. The visual content is summarized according to the methods explained in [112]. Then, the *Content DI* is adapted to three terminals. The terminals for the adaptation are all MPEG-2 terminals and have, respectively, spatial resolutions of 720x576, 352x288 and 176x144. We have used the *addVariation*() operation to create the adapted videos in additional *Component* elements of the *Content DI*. Listing 14 shows the summarized and adapted *Content DI* with four *Component* elements: the original video and three summarized and adapted variations. The instance of the MPEG-7 *VariationDescriptionType* description tool indicates that the original video (with *id="original"*) has three variations in the *Component* elements with IDs *"big-sum"*, *"medium-sum"* and *"small-sum"*. This *Content DI* is fully provided in Listing 19 of Appendix A.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:cdi="urn:vpu:cain21-di"
      xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004">
 <Item xsi:type="cdi:ItemType">
  <!-- Classification -->
  .......................
  <!-- Original content -->
  <Component xsi:type="cdi:VideoComponentType" id="original">
   ..........
   <Descriptor xsi:type="VariationDescriptionType">
    <VariationSet>
     <Source xsi:type="AudioVisualType">
      <AudioVisual>
       <MediaLocator>
        <MediaUri>#original</MediaUri>
       </MediaLocator>
      </AudioVisual>
     </Source>
     <Variation priority="1">
      <Content xsi:type="AudioVisualType">
       <AudioVisual>
        <MediaLocator>
         <MediaUri>#big-sum</MediaUri>
        </MediaLocator>
       </AudioVisual>
      </Content>
      <VariationRelationship>
       summarization
      </VariationRelationship>
     </Variation>
     <Variation priority="2">
      <Content xsi:type="AudioVisualType">
       <AudioVisual>
        <MediaLocator>
         <MediaUri>#medium-sum</MediaUri>
        </MediaLocator>
       </AudioVisual>
      </Content>
      <VariationRelationship>
       summarization
      </VariationRelationship>
     </Variation>
     <Variation priority="3">
      <Content xsi:type="AudioVisualType">
       <AudioVisual>
        <MediaLocator>
         <MediaUri>#small-sum</MediaUri>
        </MediaLocator>
       </AudioVisual>
      </Content>
      <VariationRelationship>
       summarization
      </VariationRelationship>
     </Variation>
    </VariationSet>
   </Descriptor>
   ...............
   <Resource mimeType="video/mpeg"  ref="../mesh/didl/mpeg2video.mpg"/>
  </Component>
  <!-- Big size summarized content -->
  <Component xsi:type="cdi:VideoComponentType" id="big-sum">
   ................
  </Component>
  <!-- Medium size summarized content -->
  <Component xsi:type="cdi:VideoComponentType"id="medium-sum">
  </Component>
  <!-- Small size summarized content -->
  <Component xsi:type="cdi:VideoComponentType" id="small-sum">
   ...........
</Component>
 </Item>
</DIDL>
```

Listing 14: Summarized and adapted *Content DI*

This demonstration has used three *Configuration DIs*. These *Configuration DIs* use the *ARC* descriptions to request the adaptation according to the three spatial resolutions described above (labelled as *"mpeg1_720x576_adapted_online_web"*, *"mpeg1_352x288_adapted_online_web"* and *"mpeg1_176x144_adapted_online_web"* in the *Usage Environment DI*). For this adaptation test, we needed to create an *Adapter* named *RawVideoCombinerAdapter*, whose *AdapterCapabilities* description appears in Appendix A. This *Adapter* was necessary to retrieve the summarized video from the summarization module (further explained in [112]) through two TCP sockets: one for raw WAV audio and one for RAW video. To this end, we created an additional binding mode labelled as *urn:mpeg:mpeg21:2007:01-BBL-NS:handler:TCP*. Note that the binding mode of the output is *urn:mpeg:mpeg21:2007:01-BBL-NS:MEMORY*. This means that the output of this combined can be efficiently retrieved (i.e., in main memory) by the subsequent *Adapters*. The three terminals in this adaptation test were defined with the standard *HTTP* binding mode in Table 2 of Chapter 4. During the adaptation, the *Planner* produced the following sequence of conversions: *initial* → *raw_video_combiner* → *online_video_transcoder* → *http_delivering* → *goal*.

5 Extensions to the MPEG-21 schema

To recapitulate, justify and validate the extensions to the MPEG-21 schema that this work proposes, *Demonstration 1* (in Subsection 3.1.2 of this Chapter) to *Demonstration 6* (in Subsection 4.2.2 of this Chapter) have used and described these extensions. From these demonstrations, we offer the following conclusions:

1. The description schema proposed in Listing 21 of Appendix A enable the description of multiple terminals. Listing 22 of Appendix A gathers the description of the current terminals in CAIN-21. In *Demonstration 1* to *Demonstration 6* these terminals were those labelled as "*mpeg2_medium_desktop*", "*svc_without_audio_352x288_15fps*", "*iphone*", "*http_nokia_n95*", "*mpeg1_720x576_adapted_online_web*",

"*mpeg1_352x288_adapted_online_web*" and "*mpeg1_176x144_adapted_online_web*", respectively. Appendix A provides a full description of the current terminals in CAIN-21. To indicate the target terminal of the adaptation this information has to be provided. As MPEG-21 does not define a standard description tool for this purpose, Section 3 of Chapter 4 has proposed the ARC description tool.

2. To enable systematic and automatic adaptation decisions, the inputs and outputs of the conversion tools have to be provided. Subsection 4.1 of Chapter 4 proposed the *AdapterCapabilities* description tools. The feasible inputs and output properties are defined using the *Preconditions* and *Postconditions* elements.
3. For systematic and automatic decisions, it is also necessary to know how the media is going to be transferred to either the next *Adapter* or the target terminal. This justifies the introduction of the binding mode in the description schema.
4. The modality of the content appears in the *mpeg7:Content* description tool. However, this information is not provided by the *mpeg21:TerminalType* description tool. During the decision process, the *Planner* has to determine the modality of the content that the terminal accepts. The inference rule described in Section 7 of Chapter 4 can be used in this case. Specifically, from the *mpeg21:TransportCapabilitiesType* description tool of Listing 9 of Chapter 4, the *Planner* can infer that the content has to be visual or audiovisual.
5. In *Demonstration 5*, the decision process needs to know the binding mode to determine whether the *http_delivering* conversion has to be added to the sequence. This information removes ambiguity and validates the first extension in Subsection 7.3 of Chapter 4.
6. Before adding the *optional* attribute to the *mpeg21:AudioCapabilitiesType* description tool (extension 2 in Subsection 7.3 of Chapter 4), the *Planner* did not encounter a sequence for *Demonstration 4* and *Demonstration 5*. This happened because the postconditions of the *image_2_video* conversion did not contain this information. Labelling the audio as *optional* (see Listing 9 of Chapter 4) allows the *Planner* to ignore the audio properties during the computation of the sequence.

6 Conclusions

The objective of this chapter is to validate the results of this thesis using two tools: *experiments* that test the hypothesis and *demonstrations* that describe how different types of automatic decisions are executed. Theorems formally *proven* (see Section 5 of Chapter 5) do not require further validation The first part of the chapter has studied the incompleteness semantics, whose usefulness is grounded in the fact that it significantly reduces both the description length of the adaptation capabilities and the planning decision time.

The second part of this chapter have shown how, in order to identify the feasible sequence of conversions, the *Planner* makes static decisions that choose the best sequence and the best parameters. This part of the chapter has demonstrated how the *Planner* addresses best-sequence-based decisions to manage the trade-off that sometimes occurs between the execution costs and the content degradations. If this trade-off happens, the *Planner* uses the user preferences to make a decision. This part of the chapter has also demonstrated how the user preferences can be used to make outcome-based static decisions. The user preferences for static decisions can be mapped to the MPEG-21 framework, but we have needed to make some extensions to the MPEG-21 schema.

The third part of the chapter shows how dynamic decisions are made in the pluggable *Adapter* modules during the execution phase. This part of the chapter demonstrates how to make quality-based decisions with scalable video. Several objective quality metrics can be used; however, the more challenging decision is over which quality metric maximizes the user's experience. To this end, we have to either use the adaptation engine preferences or accomplish some kind of subjective quality metric. This part of the chapter also demonstrates how to make semantic-based decisions. These kinds of decisions need to know both the meaning of the content and the user's preferences in order to improve the user's experience. In the demonstrations, the content is annotated (e.g., objective quality, ROIs, genre) so that the decision methods best understand the meaning of this content.

Chapter 8:

Contributions, conclusions and future work

Synopsis:

This final chapter highlights the main achievements and contributions of this thesis, analyses the consequences and limitations of the obtained results and, in light of these, proposes areas of future work.

1 Contributions

Overall, this thesis has contributed to systematic and automatic multimedia adaptation by means of metadata. The thesis has developed decision methods that decide which adaptation modules and parameters to execute in order to adapt different multimedia content in different contexts. In addition, the thesis contributes to the improvement of the MPEG-21 description tools by making a more precise description of the multimedia content and context. This section collects all the major contributions of each of the chapters in Part II, which are as follows:

Chapter 3: Architecture. This chapter describes the CAIN-21 multimedia adaptation engine. Subsequent chapters use this engine to study the methods that enable extensible, multi-step systematic and automatic multimedia adaptation. The major contributions of this chapter are:

- *Architecture*. There is no final consensus on the elements of a multimedia adaptation engine. Nevertheless, the chapter has highlighted the importance of differenciating between the decision and the execution phases.
- *Automatic decision*. To liberate the users from the responsibility of having to execute the adaptation operations, automatic decision methods have been analyzed. The first part of these decisions has been situated in the *Planner* module, which is executed before initiating the adaptation. The chapter also proposes that, during the execution of the adaptation, the *Adapter* modules be able to perform further automatic decisions.
- *Systematic adaptation*. Current multimedia adaptation engines are focused on the adaptation of a specific group of related content (e.g., scalable video, HTML pages, etc). Systematic adaptation addresses multimedia adaptation regardless of the multimedia content and context. The chapter has shown how the semantic-agnostic *Planner* module facilitates the use of the same decision methods with dissimilar multimedia elements.
- *Extensibility and interoperability*. The extensibility mechanism allows for the integration of pluggable adaptation modules. The adaptation modules of CAIN-21 are named *Adapters* and their adaptation capabilities are formally described in the *Adapter Capabilities DIs*. The chapter has shown how the *Adapters* progressively address wider ranges of adaptations. This wider range of adaptations means that more multimedia devices are able to consume new kinds of multimedia content and therefore extensibility facilitates interoperability.
- *Use of multimedia properties*. All information required in the adaptation process is homogeneously described by using the so-called *multimedia properties*. The chapter has demonstrated that multimedia properties presents several advantages with respect to using a standard XML representation: (a) all the decision-related information can efficiently be held in memory; and (b) if new symbols are added to the XML documents, the source code of the adaptation engine does not have to be changed. Subsection 2.2 of Chapter 3 explained why it only implies changes in the *Properties DI* document.

Chapter 4: Extensions to the MPEG-21 schema. This chapter addresses the representation of the multimedia elements involved in multimedia adaptation. To achieve interoperability, the MPEG-21 framework is used as much as possible. However, during this representation some limitations and ambiguities were identified. Subsequently, the chapter has proposed some extensions to the MPEG-21 schema that address these issues. The extensions are demonstrated in CAIN-21. The major contributions of this chapter are:

- *Standard representation*. The chapter has emphasized the use of the MPEG-21 framework to homogeneously describe the elements of the entire multimedia system. In addition, to support and communicate with external representation technology (i.e., non-MPEG-21 technology), the *Translator* module, which acts as a gateway, has been proposed.

- *Content DI, Context DI and Configuration DI*. The notion of *Content DI* and *Context DI* have been revitalized and extended with the notion of *Configuration DI*, which purpose is to decouple the *Content DI* and the *Context DI*. Instead of storing information of the adaptation state in the *Content DI* (as the MPEG-21 Part 2 *Choice* mechanism or MPEG-21 Part 7 DIA Configuration do), the *Configuration DI* sets apart all the information related to the adaptation that is to be carried out. In this way, the *Content DI* will not be modified when the target usage environment (located in a *Context DI*) changes.
- *ARC description tool*. The *Configuration DI* contains an *ARC* description. The ARC description extends the *DIA Configuration* to single out the ID of the current usage environment from the ones available in *Context DI*. In particular, when the adaptation process is launched, the ARC description indicates which terminal, network and user are the target of the adaptation.
- *Properties DI*. This is a systematic mechanism to manage multimedia properties. The chapter proposes, following a declarative approach, gathering in here all the information required by the multimedia adaptation process. In this way, changes in the metadata do not imply changes in the underlying source code. Instead, these changes imply only modifying the *Properties DI*. In particular, properties are represented as a label with an associated XPath expression. In addition, the *Properties DI* removes ambiguities because, if the same XML element appears in different parts of the document, its XPath expression will be different. Therefore, by providing different labels to semantically different properties the underlying ambiguity is avoided.
- *Alternative conversion capabilities*. The chapter develops a model that allows for describing alternative conversions capabilities. In particular, the *AdapterCapabilities* is divided into independent *ConversionCapabilities* elements. Each *ConversionCapabilities* element contains a *Preconditions* element describing the conditions under which it can be executed. In addition, the *Postconditions* element describes the changes in the properties that its execution can produce.
- *Binding mechanism*. A homogeneous and systematic mechanism to specify how the media is transferred between conversions in the multi-step adaptation process and between the last conversion and the target terminal.
- *Storable and Repeatable tests*. The chapter describes how to make the adaptation tests storable and repeatable. To this end, it uses the standard representation of the multimedia elements that MPEG-21 proposes. In addition, the chapter describes why we have needed to extend some of these elements.

Chapter 5: Selection of feasible adaptations. This chapter addresses the construction of a systematic and automatic *Planner* module. In particular, the module only relies on the multimedia format to decide which execution orders and which parameters of the conversions modules can perform the adaptation. This *Planner* is intentionally content agnostic (i.e., semantic-less) because in this way the metadata (multimedia properties) are the only elements necessary to make a decision. Nonetheless, to improve the adaptation results, the analysis of the content of the resource and its semantic can be achieved in a subsequent stage. The major contributions of this chapter are:

- *Sound multimedia adaptation planning*. This research has proven that AI planning techniques can make sound automatic decisions in the multimedia domain. The *Planner* algorithm is sound and produces a finite and complete plan. These features, in general, however, do not hold when a planning algorithm permits the removal of effects. For these reasons, Subsection 3.3 of Chapter 5 has proposed a machnism that never removes effects.

- *Bounded non-deterministic planning*. The *Planner* is well suited to represent parameterized conversion states and conversion capabilities of the conversion modules. We have replaced the notion of action with the notion of conversion in such a way that different parameters lead to different actions. Multi-valued parameters make it possible to gather related actions in a single conversion state. The source parameters (that may be multi-valued representing the selected inputs) represent the input to the conversion. The target parameters control the output of the conversion. The target parameters can be multi-valued in order for the *Adapter* to make dynamic decisions. One conversion can be comparable to a set of actions in a Graphplan-like planner. Subsection 2.4 of Chapter 5 explains peculiarities of conversions such as using selected conversion states to determine the source and target parameters.
- *Advantages of multimedia properties*. Several advantages of the multimedia properties have been demonstrated: (a) the multimedia properties directly represent the information that the *Planner* requires; (b) this information is represented homogeneously; (c) alternatives are easily represented by means of multi-valued properties; and lastly, (d) multi-valued properties allow for the representation of postponed decisions.
- *Matching process*. The matching process represents an easy and efficient way to check the relations between the MPEG-7 and MPEG-21 descriptions of the *Component* states (both selected and realized) and also of the conversion states that modify the *Component*. To check these relations, the classical and neoclassical planning approach – of modelling preconditions and effects with first order logic predicates – has been replaced by properties. The matching process has replaced the state transition function that performs unifications between predicates. Instead of computing intertwined states and actions, this replacement is the basis for the computation of conversion states and uses the parameters to implicitly represent the state of the *Component* being adapted.
- *Postponed decisions*. Most planners make all the decisions during the decision phase. We allow for the postponement of decisions that depend on the multimedia content. In contrast to continuous planning, the conversions of the *Planner* are bounded non-deterministic actions. As a result, the *Planner* does not perform further decisions that depend on the result of the dynamic decisions transferred to the *Adapter*. That is, the *Planner* computes all the sequences of conversions before the *Executer* can start executing the *Adapters*. The notion of postponed decisions cannot be found in existing multi-step multimedia adaptation planning algorithms.
- *Parameters for optimization*. Previous multi-step multimedia adaptation planning algorithms seldom search for more than one sequence of actions. Conversely, the bounded non-deterministic planner developed in this work searches for all the sequences of conversions capable of adapting the content to the usage environment. This feature allows further decisions in order to pick the sequence that optimizes some criterion (such as execution time or resulting spatial resolution). In addition, in contrast to a neoclassical planner, the bounded non-deterministic planner must find the source and target parameters that must be supplied to the non-deterministic conversion modules.
- *Tolerating partial description*. Traditional decision systems assume the complete description of the adaptation capabilities. This work has proposed the use of selected and realized states to represent the manifold states that non-deterministic actions can produce (instead of the belief states that planning under uncertainty typically has used). Moreover, Section 2 of Chapter 7 has demonstrated that the computational time and description size of the adaptation capabilities decrease significantly when partial description is allowed.
- *Absent properties*. The *Planner* deals with absent properties, which are useful in practical applications. Furthermore, the absent properties, in conjunction with multi-valued properties

allow the *Planner* to navigate through a set of conversion states. These are only partially determined using the same technique as the neoclassical planners. The experiments have proven that these sets significantly reduce the number of states that must be evaluated and therefore speed up the decision process. Previous multimedia adaptation planners do not take advantage of this idea.. Absent properties must not be confused with flexible planning: the former correspond to a lack of information, while the latter introduces soft constraints in the classical planning domain definition.
- *Partial decision*. The *Planner* partially decides the adaptation constraints that the conversion modules must comply with and postpones other decisions to the conversion modules. These conversion modules can use dissimilar decision techniques referred to as dynamic decisions. For instance, the *Image2VideoAdapter* can make decisions (using the ROIs stored in the *Content DI*) to show the faces in the image being adapted, instead of the whole subjects in the image.

Chapter 6: Best adaptation decision-making. This chapter brings into play the user preferences in order to decide which of the feasible adaptation maximizes the user's satisfaction. The major contributions of this chapter are:

- *Content-agnostic user preferences model*. Different authors have employed the preferences at different levels and in an ad-hoc manner. This chapter has developed a model to represent the user preferences that does not depend on the semantics of the preferences. To this end, the qualitative, quantitative and threshold preference model has been analyzed. After that, a systematic way to map these preference relations to the MPEG-21 schema has been discussed. Finally, the chapter has discussed how the best preference can be systematically and automatically identified by means of a preference graph.
- *Constraints hierarchy*. The chapter has brought to multimedia adaptation the idea of organizing the preferences in a constraint hierarchy. In this way, not all the constraint elements have the same importance, but rather elements in the hierarchy with higher priority levels are taken into account first.
- *Adaptation engine preferences*. We have found that, the MPEG-21 UED tools do not take into account the preferences of the adaptation engine. Therefore, we have proposed a solution to introduce the adaptation engine preferences in the decision process. In particular, we have proposed to put these preferences at the end of the constraints hierarchy.
- *Fallback mechanism*. We have brought to multimedia adaptation Danan's idea of allowing the end-user to provide only those preferences whose meaning the user *knows*. Otherwise, we would be making the user fill in preferences that do not describe the real wishes of the user. We have also described how, if we provide the whole set of preferences in the adaptation engine, unique and deterministic decision results are guaranteed.
- *Decision points*. The work has identified and clarified the points at which decisions can be taken. These points have been divided into three states: selection, static decision and dynamic decisions. The selection is executed during the planning phase and identifies all the feasible sequences of conversions and parameters. The static decisions are also executed in the decision phase just after the selection. They have been divided into best-sequence-based decisions and best-outcome-based decisions. The dynamic decisions are executed during the execution phase and are divided into quality-based-decisions and semantic-based-decisions. All these decision points have been demonstrated in CAIN-21.
- *Execution cost and content degradation*. The minimum length sequence idea proposed by Berhe et al. has been generalized by adding the notion of execution cost and content degradation. With this improvement, we no longer have to search for the shorter sequence, but rather

the user's preferences allows for searching for the sequence with a lower execution cost, content degradation, or both. In addition, the experiments have demonstrated that the results of adaptation can be improved in comparison with systems that do not take into account the execution cost, or the content degradation.

- *Dynamic decisions*. Our *Planner* transfers parts of the decision to the *Adapters*. In particular, if the *Planner* concludes that a conversion may be executed with more than one feasible value in the parameters, the selection of this value is transferred to the *Adapters*. The experiments have demonstrated the accuracy and soundness of this idea with the quality-based and the semantic-based decisions.

2 Conclusions

Current multimedia adaptation decision methods are focused on deciding how to adapt specific types of content rather than deciding how to address multimedia adaptation regardless of its content and context. In this regard, it is worth noting that MPEG-21 actually has envisioned and defined a very generic multimedia system description. In addition to the generic and systematic view, there is no consensus on the types of and temporal points at which multimedia automatic adaptation decisions can be made.

This thesis has demonstrated that it is possible to effectively use metadata in order to make systematic and automatic decisions for multimedia adaptation. Automatic decision mechanisms release the user from this responsibility. To make decisions we have developed a planning mechanism that is independent of the semantic of the multimedia content. In particular, the planner only utilizes metadata describing the media format and the usage environment constraints. In this way, the same method can be applied to manage different multimedia adaptation problems. In addition, the semantic of the content can be taken into account in a subsequent stage. This work has identified the decision points at which the computing system can make these decisions. In addition, to increase the user's experience, these automatic decisions use the user preferences to customize the content to his/her personal preferences.

Another important objective of this thesis is interoperability. In this regard, we have defined an extensibility mechanism for pluggable adaptation modules. Then we have shown how combining the existing pluggable adaptation modules progressively augments the range of feasible adaptation. We also have shown that the adaptation engine can automatically identify in which order and with which parameters the adaptation modules have to be executed. The resulting adaptation system is interoperable, progressively extensible and able to address a wide variety of multimedia adaptation problems.

To demonstrate our proposals, we have created a set of storable and repeatable adaptation tests. The description tools that formalize these tests are mostly extracted from the MPEG-21 framework. In this way, we achieve a higher level of interoperability and in fact, this framework is very generic and reusable. However, in some cases, we have identified ambiguities or limitations, and in these cases, we have had to modify these description tools.

At the end of the contributions stage, the thesis describes three major decision points: selection, static decisions and dynamic decisions. Clearly, this is not the only feasible organization for the decision points; a completely different organization of the decision points may also attain systematic and automatic adaptations. However, at least, this work has demonstrated that it is feasible to make systematic and automatic adaptation.

In the same manner, currently there is no consensus on the user preferences that help in making systematics and automatic multimedia adaptation decisions. To achieve a more systematic management of the user preferences, we have proposed organizing these preferences in a constraint hierarchy and consistently using preference relations to represent the qualitative or qualitative relationships between the outcomes of the entire set of preferences. From this semantic-agnostic representation, we are able to create a preference graph that explicitly arranges all these relations and enables the systematic and automatic search for the optimal outcomes.

3 Future work

The results of this work give rise to new research opportunities. In particular, we have organized our proposal for future research direction according to the results of the chapters in Part II:

Architecture. CAIN-21 allows for the dynamic addition of new *Adapters* and *Adapter Capabilities DIs*. In addition, the descriptions of the feasible binding modes determine the ways in which the media can be transferred throughout the chain of *Adapters*. Adaptation engines such as the one published in [58] have demonstrated distributed adaptation of scalable content in different nodes. Authors such as [16] have also demonstrated the distribution of their adaptation Web Services. From these ideas, we propose the following research areas for future work:

- *Distributed adaptation plans*. Adding these ideas to CAIN-21 to achieve distributed execution of the *Adapters*, and also to analyze whether multi-agent planning approaches can be added to the *Planner* module in order to create distributed adaptation plans.

Extensions to the MPEG-21 schema. We have proposed several extensions to the MPEG-21 description tools. In this regard, we propose the following idea for future work:

- *Adding these description tools to the MPEG-21 standard*. In this way, these description tools could be reused by industry and future researchers.

Selection of feasible adaptations. We have developed a *Planner* to systematically identify all the feasible sequences of conversions and to later make decisions regarding the best sequence and parameters. From the results of this chapter, we have found several opportunities for future work.

- *Standard planners and standard representation schemes*. To study whether the decision that our *Planner* achieves can be implemented by means of standard planners and standard plan representation schemes (such as PDDL). In this area we propose to study continuous orchestration of web services via planning [113]. It would be also interesting to investigate whether least commitment planning can be used to postpone our decisions [114].
- *To analyze the order of the sequences*. One opportunity for further inquiry is related to the conversion states. If there are several sequences that contain the same conversion states, but in a different order, which order should be the most appropriate? Here, the system should evaluate whether different orders provide the same outcome (the same outcome may mean the same adaptation quality) and/or whether different orders yield "cheaper" sequences. For example, if the last conversion state is lossy and the remaining ones are lossless (or nearly lossless), then a different order for the sequences of conversion states may be faster than this suboptimal one. In some cases, this lateral effect of conversion modules reusability can be observed regardless of the order in which we have decided to execute the conversion modules

(i.e., both conversion modules degrades the content). For instance, in *Demonstration 4* of Chapter 7 the video has been transcoded twice. First, the *small_image_2_video* conversion produces MPEG-2 video and secondly, the *ondemand_mp4_transcoder* conversion produces 3GPP video. For this purpose, approaches such as *Partial Order Planning* (POP) may be investigated [115].

- *To analyze whether the accumulated effect could be avoided*. A second opportunity for future development lies in the accumulated effects. During each step in the backwards search, our *Planner* forces the accumulated effects to increase or to remain in order to guarantee finiteness. This means that during the execution of the adaptation plan we never lose properties. However, sometimes, depending on the target terminal, properties make no sense. For instance, the *frame_rate* of the original video *Component* makes no sense if we are adapting to an only image- or audio-capable terminal. To avoid this problem, the *Adapters* and the terminal must not use properties that they don't declare. Nevertheless, it would be convenient to study whether it is possible to remove these properties from the adaptation plan.
- *To avoid duplicate states*. A third research area lies in the improvement of the performance of the reachability analysis. Our *Planner* builds a reachability tree that has been proven finite and complete. Graphplan introduced reachability ideas to reduce the computational spatial costs by avoiding the expansion of duplicate states. This idea could be incorporated into our *Planner*.
- *To consider desirable constraints*. A fourth research direction deals with flexible planning [64]. Having a set of decoding and transmission constraints, how should the *Planner* maximize the number of desirable constraints satisfied? One way is to maximize the number of desirable constraints (each desirable constraint has the same weight), but another option is to establish a ranking of constraints (each constraint may have a different weight).

Best adaptation decision-making. In reference to these decision approaches, we propose the following opportunities of future work:

- *To further elaborate on the content degradation metrics*. For the best-sequence-based decisions, we have proposed using the estimated execution costs and content degradations in order to select the best sequence. However, the model used to estimate the content degradations could be further elaborated. In this future research area, it would be important to consider that the content degradation depends on the parameters that the conversion module receives. For instance, if the parameters means that the frame size has to be considerably reduced, then the content degradation would be bigger than in cases in which the frame size is not modified.
- *To decide which objective quality metric is best for quality-based adaptation of scalable video*. In this regard, we are aware that the authors of [17] are conducting subjective experiments to shed some light on this topic.

APPENDIXES

Appendix A: Schema and description documents

This appendix gathers a set of description tools (XML Schema documents) and descriptions (XML documents). These documents are referenced in the thesis, but due to their length, they have been moved to this appendix. In addition, these documents are also publicly available at *cain21.sourceforge.net*. These elements have been restricted (using XML Schema derivation by restriction [25]) or extended (using XML Schema derivation by extension [25]) to reflect the current capabilities of CAIN-21.

Content DI

Listing 15 shows the MPEG-21 Part 2 DIDL elements that CAIN-21 utilizes. The target namespace of these elements is *urn:vpu:cain21-di*, which is shortened to *cdi*. The *cdi:ItemType* description tool has been restricted to contain up to one *Descriptor* and up to one *Component*. The *Descriptor* is used to optionally contain an instance of the *mpeg7:ClassificationType* description tool (see Listing 18 below). The *Component* can contain several *Descriptors* and *Resources*. There are several extensions of the *cdi:ComponentType* description tool to indicate its particular content (*cdi:ImageComponentType*, *cdi:AudioComponentType*, *cdi:VideoComponentType*, etc.) The *Descriptors* can be instances of either the *cdi:Mpeg7DescriptorType* or the *cdi:AdaptationQoSDescriptorType* description tools.

```
<?xml version="1.0" ?>
<xsd:schema targetNamespace="urn:vpu:cain21-di"
            elementFormDefault="qualified"
            attributeFormDefault="unqualified"
            xmlns="urn:vpu:cain21-di"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:didl="urn:mpeg:mpeg21:2002:02-DIDL-NS"
            xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
            id="cdi.xsd">

 <!-- The following import elements are just hints about the types
      and elements used by this schema -->
 <xsd:import namespace="urn:mpeg:mpeg21:2002:02-DIDL-NS"
             schemaLocation="didl.xsd"/>
 <xsd:import namespace="urn:mpeg:mpeg7:schema:2001"
             schemaLocation="Mpeg7-2001.xsd"/>

 <!-- Variation Descriptor -->

 <xsd:complexType name="VariationDescriptorType">
  <xsd:complexContent>
   <xsd:restriction base="didl:DescriptorType">
    <xsd:sequence>
     <xsd:element ref="didl:Statement" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="didl:ID_ATTRS"/>
   </xsd:restriction>
  </xsd:complexContent>
 </xsd:complexType>

 <!-- Item declarations -->

 <xsd:complexType name="ItemType">
  <xsd:complexContent>
   <xsd:restriction base="didl:ItemType">
    <xsd:sequence>
```

```
     <xsd:element ref="didl:Descriptor" minOccurs="0"
                  maxOccurs="1"/>
     <xsd:element ref="didl:Component" minOccurs="1"
                  maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="didl:ID_ATTRS"/>
   </xsd:restriction>
  </xsd:complexContent>
 </xsd:complexType>

 <!-- Component declaration -->

 <xsd:complexType name="ComponentType">
  <xsd:complexContent>
   <xsd:restriction base="didl:ComponentType">
    <xsd:sequence>
     <xsd:element ref="didl:Descriptor" minOccurs="0"
                  maxOccurs="unbounded"/>
     <xsd:element ref="didl:Resource" minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:ID" use="required"/>
   </xsd:restriction>
  </xsd:complexContent>
 </xsd:complexType>

 <xsd:complexType name="TextComponentType">
  <xsd:complexContent>
   <xsd:extension base="ComponentType">
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>

 <xsd:complexType name="ImageComponentType">
  <xsd:complexContent>
   <xsd:extension base="ComponentType">
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>

 <xsd:complexType name="AudioComponentType">
  <xsd:complexContent>
   <xsd:extension base="ComponentType">
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>

 <xsd:complexType name="VideoComponentType">
  <xsd:complexContent>
   <xsd:extension base="ComponentType">
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>

 <xsd:complexType name="HTMLComponentType">
  <xsd:complexContent>
   <xsd:extension base="ComponentType">
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>

 <xsd:complexType name="SMILComponentType">
  <xsd:complexContent>
   <xsd:extension base="ComponentType">
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>

 <!-- MPEG-7 Descriptors -->

 <xsd:complexType name="Mpeg7DescriptorType">
  <xsd:complexContent>
   <xsd:restriction base="didl:DescriptorType">
    <xsd:sequence>
     <xsd:element ref="didl:Statement" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="didl:ID_ATTRS"/>
   </xsd:restriction>
  </xsd:complexContent>
 </xsd:complexType>

 <!-- AdaptationQoS Description -->

 <xsd:complexType name="AdaptationQoSDescriptorType">
  <xsd:complexContent>
   <xsd:restriction base="didl:DescriptorType">
    <xsd:sequence>
     <xsd:element ref="didl:Statement" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="didl:ID_ATTRS"/>
   </xsd:restriction>
  </xsd:complexContent>
 </xsd:complexType>

</xsd:schema>
```

Listing 15: *Content DI* XML Schema of CAIN-21

Listing 16 shows an example of a *Content DI* containing a *cdi:ImageComponentType* description type along with an image resource and a *Descriptor* that describe the image media format using the standard *mpeg7:MediaDescriptionType* description tool.

```
<?xml version="1.0" ?>

<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:cdi="urn:vpu:cain21-di"
      xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004">
 <Item xsi:type="cdi:ItemType">
  <Component xsi:type="cdi:ImageComponentType" id="c1">
   <Descriptor xsi:type="cdi:Mpeg7DescriptorType">
    <Statement mimeType="text/xml">
     <Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2004">
      <Description xsi:type="MediaDescriptionType">
       <MediaInformation>
        <MediaProfile>
         <ComponentMediaProfile>
          <MediaFormat>
           <Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:4">
            <Name>Visual</Name>
           </Content>
           <VisualCoding>
            <Frame height="343" width="454"/>
           </VisualCoding>
          </MediaFormat>
         </ComponentMediaProfile>
         <MediaFormat>
          <Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:4.1">
           <Name>Image</Name>
          </Content>
          <FileFormat href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:1">
           <Name>jpeg</Name>
          </FileFormat>
         </MediaFormat>
        </MediaProfile>
       </MediaInformation>
      </Description>
     </Mpeg7>
    </Statement>
   </Descriptor>
   <!-- The resource itself -->
   <Resource mimeType="image/jpeg" ref="painting.jpg"/>
  </Component>
 </Item>
</DIDL>
```

Listing 16: Example of a *Content DI*

Listing 17 shows a *Content DI* in which the faces in the image have been annotated using ROIs. In particular, the standard *mpeg7:StillRegionType* description tool is used to indicate the rectangular boxes of the faces. The use of this *Content DI* is described in Subsection 4.2 of Chapter 6 and *Demonstration 4* of Chapter 7.

```
<?xml version="1.0" ?>

<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:cdi="urn:vpu:cain21-di"
      xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004">
 <Item xsi:type="cdi:ItemType">
  <Component xsi:type="cdi:ImageComponentType"
             id="c1">
   <Descriptor xsi:type="cdi:Mpeg7DescriptorType">
    <Statement mimeType="text/xml">
     <Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2004">
      <Description xsi:type="MediaDescriptionType">
       <MediaInformation>
        <MediaProfile>
         <ComponentMediaProfile>
          <MediaFormat>
           <Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:4">
            <Name>Visual</Name>
           </Content>
           <VisualCoding>
            <Frame height="343" width="454"/>
           </VisualCoding>
          </MediaFormat>
         </ComponentMediaProfile>
         <MediaFormat>
          <Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:4.1">
           <Name>Image</Name>
          </Content>
          <FileFormat href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:1">
           <Name>jpeg</Name>
          </FileFormat>
         </MediaFormat>
        </MediaProfile>
       </MediaInformation>
      </Description>
     </Mpeg7>
    </Statement>
   </Descriptor>
   <Descriptor xsi:type="cdi:Mpeg7DescriptorType">
```

```
    <Statement mimeType="text/xml">
     <Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2004">
      <DescriptionUnit xsi:type="StillRegionType" id="faces">
       <SpatialMask>
        <SubRegion>
         <Box mpeg7:dim="2 2"> 136 178 165 212</Box>
        </SubRegion>
        <SubRegion>
         <Box mpeg7:dim="2 2"> 157 108 181 135</Box>
        </SubRegion>
        <SubRegion>
         <Box mpeg7:dim="2 2"> 215 80 240 120</Box>
        </SubRegion>
        <SubRegion>
         <Box mpeg7:dim="2 2"> 269 81 293 108</Box>
        </SubRegion>
        <SubRegion>
         <Box mpeg7:dim="2 2"> 311 76 332 101</Box>
        </SubRegion>
        <SubRegion>
         <Box mpeg7:dim="2 2"> 357 73 378 98</Box>
        </SubRegion>
       </SpatialMask>
      </DescriptionUnit>
     </Mpeg7>
    </Statement>
   </Descriptor>
   <!-- The resource itself -->
   <Resource mimeType="image/jpeg"  ref="people.jpg"/>
  </Component>
 </Item>
</DIDL>
```

Listing 17: Example of a Content DI with ROIs

Listing 18 shows a *Content DI* containing a news item. The standard *mpeg7:ClassificationType* description tool indicates the genre of the news item. The standard *mpeg7:MediaDescriptionType* description tool indicates the format of the video. The instance of the *mpeg21:VariationDescriptionType* description type indicates that this is the original news item to be later summarized and adapted.

```
<?xml version="1.0" ?>

<!-- Example of DI that conveys only a news item -->

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

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

  <!-- Original content -->
  <Component xsi:type="cdi:VideoComponentType"
             id="original">
   <Descriptor xsi:type="cdi:Mpeg7DescriptorType">
    <Statement mimeType="text/xml">
     <Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2004">
      <Description xsi:type="MediaDescriptionType">
       <MediaInformation>
        <MediaProfile>
         <ComponentMediaProfile>
          <MediaFormat>
           <Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:4">
            <Name>Visual</Name>
           </Content>
           <BitRate>104857000</BitRate>
           <VisualCoding>
            <Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:1"
                    colorDomain="color">
             <Name xml:lang="en">MPEG-1 Video Coding Format</Name>
            </Format>
            <Pixel aspectRatio="0.75"/>
            <Frame height="576" width="720" rate="25"/>
           </VisualCoding>
          </MediaFormat>
         </ComponentMediaProfile>
         <ComponentMediaProfile>
          <MediaFormat>
           <Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:1">
            <Name>Audio</Name>
           </Content>
           <BitRate>384000</BitRate>
           <AudioCoding>
            <Format
             href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4">
             <Name xml:lang="en">MPEG-2 Audio Coding Format</Name>
            </Format>
           </AudioCoding>
          </MediaFormat>
         </ComponentMediaProfile>
         <MediaFormat>
          <Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:2">
           <Name>Audiovisual</Name>
          </Content>
          <FileFormat href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:3">
           <Name>mpeg</Name>
          </FileFormat>
          <FileSize>45361152</FileSize>
          <BitRate>4843000</BitRate>
         </MediaFormat>
        </MediaProfile>
       </MediaInformation>
      </Description>
      <Description xsi:type="VariationDescriptionType">
       <VariationSet>
        <Source xsi:type="AudioVisualType">
         <AudioVisual>
          <MediaLocator>
           <MediaUri>#original</MediaUri>
          </MediaLocator>
         </AudioVisual>
        </Source>
```

```
        <Variation priority="1">
         <Content xsi:type="AudioVisualType">
          <AudioVisual>
           <MediaLocator>
            <MediaUri>#summarization</MediaUri>
           </MediaLocator>
          </AudioVisual>
         </Content>
         <VariationRelationship>
          summarization
         </VariationRelationship>
        </Variation>
       </VariationSet>
      </Description>
     </Mpeg7>
    </Statement>
   </Descriptor>

   <Resource mimeType="video/mpeg" ref="../mesh/didl/mpeg2video.mpg"/>
  </Component>

 </Item>
</DIDL>
```

Listing 18: Original *Content DI* to be summarized and adapted

Listing 19 shows the *Content DI* that results from adapting the *Content DI* in Listing 18. This adaptation is described in *Demonstration 6* of Chapter 7. In particular, the original *Component* is preserved and three new variations are added. The standard *mpeg7:VariationDescriptionType* description tool describes these variations.

```
<?xml version="1.0" ?>

<!-- Example of DI that conveys a news item and its adapted summary -->

<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:cdi="urn:vpu:cain21-di"
      xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004">
 <Item xsi:type="cdi:ItemType">

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

  <!-- Original content -->
  <Component xsi:type="cdi:VideoComponentType"
             id="original">
   <Descriptor xsi:type="cdi:Mpeg7DescriptorType">
    <Statement mimeType="text/xml">
     <Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2004">
      <Description xsi:type="MediaDescriptionType">
```

```
       <MediaInformation>
        <MediaProfile>
         <ComponentMediaProfile>
          <MediaFormat>
           <Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:4">
            <Name>Visual</Name>
           </Content>
           <BitRate>104857000</BitRate>
           <VisualCoding>
            <Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:1"
                    colorDomain="color">
             <Name xml:lang="en">MPEG-1 Video Coding Format</Name>
            </Format>
            <Pixel aspectRatio="0.75"/>
            <Frame height="576" width="720" rate="25"/>
           </VisualCoding>
          </MediaFormat>
         </ComponentMediaProfile>
         <ComponentMediaProfile>
          <MediaFormat>
           <Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:1">
            <Name>Audio</Name>
           </Content>
           <BitRate>384000</BitRate>
           <AudioCoding>
            <Format
                   href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4">
             <Name xml:lang="en">MPEG-2 Audio Coding Format</Name>
            </Format>
           </AudioCoding>
          </MediaFormat>
         </ComponentMediaProfile>
         <MediaFormat>
          <Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:2">
           <Name>Audiovisual</Name>
          </Content>
          <FileFormat href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:3">
           <Name>mpeg</Name>
          </FileFormat>
          <FileSize>45361152</FileSize>
          <BitRate>4843000</BitRate>
         </MediaFormat>
        </MediaProfile>
       </MediaInformation>
      </Description>
      <Description xsi:type="VariationDescriptionType">
       <VariationSet>
        <Source xsi:type="AudioVisualType">
         <AudioVisual>
          <MediaLocator>
           <MediaUri>#original</MediaUri>
          </MediaLocator>
         </AudioVisual>
        </Source>
        <Variation priority="1">
         <Content xsi:type="AudioVisualType">
          <AudioVisual>
           <MediaLocator>
            <MediaUri>#big-sum</MediaUri>
           </MediaLocator>
          </AudioVisual>
         </Content>
         <VariationRelationship>
          summarization
         </VariationRelationship>
        </Variation>
                <Variation priority="2">
         <Content xsi:type="AudioVisualType">
          <AudioVisual>
           <MediaLocator>
            <MediaUri>#medium-sum</MediaUri>
           </MediaLocator>
          </AudioVisual>
         </Content>
         <VariationRelationship>
          summarization
         </VariationRelationship>
        </Variation>
                <Variation priority="3">
         <Content xsi:type="AudioVisualType">
          <AudioVisual>
           <MediaLocator>
            <MediaUri>#small-sum</MediaUri>
           </MediaLocator>
          </AudioVisual>
         </Content>
         <VariationRelationship>
          summarization
         </VariationRelationship>
        </Variation>
       </VariationSet>
      </Description>
     </Mpeg7>
    </Statement>
   </Descriptor>
   <Resource mimeType="video/mpeg"  ref="../mesh/didl/mpeg2video.mpg"/>
  </Component>

  <!-- Big size summarized content -->
  <Component xsi:type="cdi:VideoComponentType"
             id="big-sum">
   <Descriptor xsi:type="cdi:Mpeg7DescriptorType">
    <Statement mimeType="text/xml">
     <Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2004">
      <Description xsi:type="MediaDescriptionType">
       <MediaInformation>
        <MediaProfile>
         <ComponentMediaProfile>
          <MediaFormat>
           <Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:4">
            <Name>Visual</Name>
           </Content>
           <BitRate>104857000</BitRate>
           <VisualCoding>
            <Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:1"
                    colorDomain="color">
             <Name xml:lang="en">MPEG-1 Video Coding Format</Name>
            </Format>
            <Pixel aspectRatio="0.75"/>
            <Frame height="576" width="720" rate="25"/>
           </VisualCoding>
          </MediaFormat>
         </ComponentMediaProfile>
         <ComponentMediaProfile>
          <MediaFormat>
           <Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:1">
            <Name>Audio</Name>
           </Content>
```

```
           <BitRate>384000</BitRate>
           <AudioCoding>
            <Format
                   href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4">
             <Name xml:lang="en">MPEG-2 Audio Coding Format</Name>
            </Format>
           </AudioCoding>
          </MediaFormat>
         </ComponentMediaProfile>
         <MediaFormat>
          <Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:2">
           <Name>Audiovisual</Name>
          </Content>
          <FileFormat href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:3">
           <Name>mpeg</Name>
          </FileFormat>
          <FileSize>7197412</FileSize>
          <BitRate>2399000</BitRate>
         </MediaFormat>
        </MediaProfile>
       </MediaInformation>
      </Description>
     </Mpeg7>
    </Statement>
   </Descriptor>
   <Resource mimeType="video/mpeg"
             ref="../mesh/didl/mpeg2video.mpg"/>
  </Component>

<!-- Medium size summarized content -->
  <Component xsi:type="cdi:VideoComponentType"
             id="medium-sum">
   <Descriptor xsi:type="cdi:Mpeg7DescriptorType">
    <Statement mimeType="text/xml">
     <Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2004">
      <Description xsi:type="MediaDescriptionType">
       <MediaInformation>
        <MediaProfile>
         <ComponentMediaProfile>
          <MediaFormat>
           <Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:4">
            <Name>Visual</Name>
           </Content>
           <BitRate>104857000</BitRate>
           <VisualCoding>
            <Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:1"
                    colorDomain="color">
             <Name xml:lang="en">MPEG-1 Video Coding Format</Name>
            </Format>
            <Pixel aspectRatio="0.75"/>
            <Frame height="288" width="352" rate="25"/>
           </VisualCoding>
          </MediaFormat>
         </ComponentMediaProfile>
         <ComponentMediaProfile>
          <MediaFormat>
           <Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:1">
            <Name>Audio</Name>
           </Content>
           <BitRate>384000</BitRate>
           <AudioCoding>
            <Format
                   href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4">
             <Name xml:lang="en">MPEG-2 Audio Coding Format</Name>
            </Format>
           </AudioCoding>
          </MediaFormat>
         </ComponentMediaProfile>
         <MediaFormat>
          <Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:2">
           <Name>Audiovisual</Name>
          </Content>
          <FileFormat href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:3">
           <Name>mpeg</Name>
          </FileFormat>
          <FileSize>7197412</FileSize>
          <BitRate>2399000</BitRate>
         </MediaFormat>
        </MediaProfile>
       </MediaInformation>
      </Description>
     </Mpeg7>
    </Statement>
   </Descriptor>
   <Resource mimeType="video/mpeg"
             ref="../mesh/didl/mpeg2video.mpg"/>
  </Component>

<!-- Small size summarized content -->
  <Component xsi:type="cdi:VideoComponentType"
             id="small-sum">
   <Descriptor xsi:type="cdi:Mpeg7DescriptorType">
    <Statement mimeType="text/xml">
     <Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2004">
      <Description xsi:type="MediaDescriptionType">
       <MediaInformation>
        <MediaProfile>
         <ComponentMediaProfile>
          <MediaFormat>
           <Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:4">
            <Name>Visual</Name>
           </Content>
           <BitRate>104857000</BitRate>
           <VisualCoding>
            <Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:1"
                    colorDomain="color">
             <Name xml:lang="en">MPEG-1 Video Coding Format</Name>
            </Format>
            <Pixel aspectRatio="0.75"/>
            <Frame height="144" width="176" rate="25"/>
           </VisualCoding>
          </MediaFormat>
         </ComponentMediaProfile>
         <ComponentMediaProfile>
          <MediaFormat>
           <Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:1">
            <Name>Audio</Name>
           </Content>
           <BitRate>384000</BitRate>
           <AudioCoding>
            <Format
                   href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4">
             <Name xml:lang="en">MPEG-2 Audio Coding Format</Name>
            </Format>
           </AudioCoding>
          </MediaFormat>
         </ComponentMediaProfile>
         <MediaFormat>
```

```
          <Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:2">
           <Name>Audiovisual</Name>
          </Content>
          <FileFormat href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:3">
           <Name>mpeg</Name>
          </FileFormat>
          <FileSize>7197412</FileSize>
          <BitRate>2399000</BitRate>
         </MediaFormat>
        </MediaProfile>
       </MediaInformation>
      </Description>
     </Mpeg7>
    </Statement>
   </Descriptor>
   <Resource mimeType="video/mpeg"
             ref="../mesh/didl/mpeg2video.mpg"/>
  </Component>

 </Item>
</DIDL>
```

Listing 19: Summarized and adapted *Content DI*

Listing 20 of Appendix A shows a *Content DI* with two resources named *he_asked.svc* and *he_asked.aac*. The visual stream uses the scalable extension for the H.264/AVC media format and the audio stream uses the AAC media format. Specifically, the visual stream is encoded using scalable format at three spatial resolutions (namely, QCIF: 176x144 pixels, CIF: 352x288 pixels, 4CIF: 704x576 pixels) and frame rates of 30, 15, 7.5 and 3.75 fps. The visual stream was encoded with one base layer and up to two quality enhancement layers. The *cdi:AdaptationQoSDescriptorType* description tool (see Listing 15) includes an instance of the standard *mpeg21:AdaptationQoSType* description tool to describe the layers of the scalable visual stream. An instance of *mpeg21:UtilityFunctionType* description tool defines the relationships among the terminal constraints, scalable video adaptation operations and their objective quality. The only constraint is the bitrate. There are four adaptation operations: *HORIZONTAL_SIZE*, *VERTICAL_SIZE*, *FRAME_RATE*, *QUALITY_LAYER*. Finally, there are three utility metrics: PSNR, VQM and SSIM[27].

In addition, we have added an instance of *cdi:VariationDescriptionType* description tool to indicate that there is an alternative *Component* that uses a standard MP4 video container for the video. In particular, as of the time of the writing of this thesis, there is no web player plug-ins for the scalable extension for H.264/AVC, so we added this variation to provide a visual feedback of the *Content DI*. Although the MP4 variation has similar quality (but not equal), it is useful to facilitate the visualisation of the content that is going to be adapted in the CAIN-21 demo.

```
<?xml version="1.0" ?>

<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:cdi="urn:vpu:cain21-di"
      xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004">
 <Item xsi:type="cdi:ItemType">
  <Component xsi:type="cdi:VideoComponentType"
             id="scalable">
   <Descriptor xsi:type="cdi:Mpeg7DescriptorType">
```

[27] The ILab of the University of Surrey, Guildford, UK has calculated the values for the *AdaptationQoS* description. I thank S. Dogan, G. Nur, M. Mrak, H. K. Arachchi for calculating this helpful information.

```
    <Statement mimeType="text/xml">
     <Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2004">
      <Description xsi:type="MediaDescriptionType">
       <MediaInformation>
        <MediaProfile>
         <ComponentMediaProfile>
          <MediaFormat>
           <Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:4">
            <Name>Visual</Name>
           </Content>
           <BitRate>7785472</BitRate>
           <VisualCoding>
            <Format href="urn:vpu:cs:VisualCodingFormatCS:2009:svc" colorDomain="color">
             <Name xml:lang="en">SVC</Name>
            </Format>
            <Pixel aspectRatio="0.81"/>
            <Frame height="576" width="704" rate="30"/>
           </VisualCoding>
          </MediaFormat>
         </ComponentMediaProfile>
         <ComponentMediaProfile>
          <MediaFormat>
           <Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:1">
            <Name>Audio</Name>
           </Content>
           <BitRate>64000</BitRate>
           <AudioCoding>
            <Format
                   href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.3.1">
             <Name xml:lang="en">MPEG-2 Audio AAC Low Complexity Profile</Name>
            </Format>
           </AudioCoding>
          </MediaFormat>
         </ComponentMediaProfile>
         <MediaFormat>
          <Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:2">
           <Name>Audiovisual</Name>
          </Content>
          <FileFormat href="urn:vpu:cs:FileFormatCS:2009:svc_audio">
           <Name>SVC with audio resource</Name>
          </FileFormat>
         </MediaFormat>
        </MediaProfile>
       </MediaInformation>
      </Description>
      <Description xsi:type="VariationDescriptionType">
       <VariationSet>
        <Source xsi:type="AudioVisualType">
         <AudioVisual>
          <MediaLocator>
           <MediaUri>#scalable</MediaUri>
          </MediaLocator>
         </AudioVisual>
        </Source>
        <Variation priority="1">
         <Content xsi:type="AudioVisualType">
          <AudioVisual>
           <MediaLocator>
            <MediaUri>#playable</MediaUri>
           </MediaLocator>
          </AudioVisual>
         </Content>
         <VariationRelationship>
          substitution
          </VariationRelationship>
         </Variation>
        </VariationSet>
       </Description>
      </Mpeg7>
     </Statement>
    </Descriptor>
    <Descriptor xsi:type="cdi:AdaptationQoSDescriptorType">
     <Statement mimeType="text/xml">
      <Description xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS" xsi:type="AdaptationQoSType">
       <Module xsi:type="UtilityFunctionType">
        <Constraint iOPinRef="BITRATE">
          <Values xsi:type="FloatVectorType">
           <Vector>27.40 37.60 48.70 58.50 47.40 60.90 66.40 82.90 88.10 107.80 108.60
                   132.10 130.20 178.30 231.20 281.90 186.10 214.90 258.50 293.20 340.20
                   381.30 419.10 467.90 404.20 560.60 735.40 907.50 404.90 567.30 601.70
                   811.90 841.10 1096.60 1094.30 1389.40</Vector>
         </Values>
        </Constraint>
        <AdaptationOperator iOPinRef="HORIZONTAL_SIZE">
         <Values xsi:type="IntegerVectorType">
          <Vector>176 176 176 176 176 176 176 176 176 176 176 176
                  352 352 352 352 352 352 352 352 352 352 352 352
                  704 704 704 704 704 704 704 704 704 704 704 704
          </Vector>
         </Values>
        </AdaptationOperator>
        <AdaptationOperator iOPinRef="VERTICAL_SIZE">
         <Values xsi:type="IntegerVectorType">
          <Vector>144 144 144 144 144 144 144 144 144 144 144 144
                  288 288 288 288 288 288 288 288 288 288 288 288
                  576 576 576 576 576 576 576 576 576 576 576 576
          </Vector>
         </Values>
        </AdaptationOperator>
        <AdaptationOperator iOPinRef="FRAME_RATE">
         <Values xsi:type="FloatVectorType">
          <Vector>3.75 7.5 15 30 3.75 3.75 7.5 7.5 15 15 30 30 3.75 7.5 15 30 3.75 3.75
                  7.5 7.5 15 15 30 30 3.75 7.5 15 30 3.75 3.75 7.5 7.5 15 15 30 30
          </Vector>
         </Values>
        </AdaptationOperator>
        <AdaptationOperator iOPinRef="QUALITY_LAYER">
         <Values xsi:type="IntegerVectorType">
          <Vector>0 0 0 0 1 2 1 2 1 2 1 2 0 0 0 0 1 2 1 2 1 2 1 2
                  0 0 0 0 1 2 1 2 12 1 2</Vector>
         </Values>
        </AdaptationOperator>
        <Utility iOPinRef="PSNR">
         <Values xsi:type="FloatVectorType">
          <Vector>37.230 36.614 36.245 36.091 39.092 40.608 38.463 39.702 38.061
                  39.103 37.906 38.848 38.719 38.175 37.818 37.681 40.681 41.907
                  40.046 41.013 39.636 40.465 39.476 40.188 41.381 40.820 40.513
                  40.379 42.261 43.389 41.872 42.704 41.628 42.325 40.552 42.164
          </Vector>
         </Values>
        </Utility>
        <Utility iOPinRef="VQM">
         <Values xsi:type="FloatVectorType">
          <Vector>0.240 0.259 0.273 0.276 0.185 0.167 0.203 0.187 0.215 0.201
                  0.274 0.206 0.200 0.213 0.246 0.228 0.153 0.143 0.166 0.158
                  0.178 0.171 0.183 0.176 0.169 0.181 0.195 0.201 0.168 0.127
                  0.180 0.140 0.194 0.152 0.193 0.157
          </Vector>
        </Values>
       </Utility>
       <Utility iOPinRef="SSIM">
        <Values xsi:type="FloatVectorType">
         <Vector>0.949 0.943 0.940 0.937 0.966 0.974 0.962 0.969 0.959 0.966
                 0.958 0.964 0.955 0.951 0.948 0.947 0.970 0.976 0.966 0.971
                 0.964 0.968 0.963 0.967 0.960 0.958 0.955 0.953 0.961 0.974
                 0.957 0.970 0.956 0.968 0.956 0.967
         </Vector>
        </Values>
       </Utility>
      </Module>
      <IOPin semantics="urn:mpeg:mpeg21:2003:01-DIA-AdaptationQoSCS:1.3.4.1"
             id="HORIZONTAL_SIZE"/>
      <IOPin semantics="urn:mpeg:mpeg21:2003:01-DIA-AdaptationQoSCS:1.3.4.1"
             id="VERTICAL_SIZE"/>
      <IOPin semantics="urn:vpu:cs:AdaptationQoS:frame_rate"
             id="FRAME_RATE"/>
      <IOPin semantics="urn:mpeg:mpeg21:2003:01-DIA-AdaptationQoSCS:1.3.9.4"
             id="QUALITY_LAYER"/>
      <IOPin semantics="urn:vpu:cs:AdaptationQoS:bit_rate" id="BITRATE"/>
      <IOPin semantics="urn:vpu:cs:AdaptationQoS:y_psnr" id="PSNR"/>
      <IOPin semantics="urn:vpu:cs:AdaptationQoS:y_psnr" id="VQM"/>
      <IOPin semantics="urn:vpu:cs:AdaptationQoS:y_psnr" id="SSIM"/>
     </Description>
    </Statement>
   </Descriptor>
   <Resource mimeType="video/svc" ref="he_asked.svc"/>
   <Resource mimeType="audio/aac" ref="he_asked.aac"/>
  </Component>

  <!-- Playable content -->
  <Component xsi:type="cdi:VideoComponentType"
             id="playable">
   <Descriptor xsi:type="cdi:Mpeg7DescriptorType">
    <Statement mimeType="text/xml">
     <Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2004">
      <Description xsi:type="MediaDescriptionType">
       <MediaInformation>
        <MediaProfile>
         <ComponentMediaProfile>
          <MediaFormat>
           <Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:4">
            <Name>Visual</Name>
           </Content>
           <BitRate>306176</BitRate>
           <VisualCoding>
            <Format href="urn:vpu:cs:VisualCodingFormatCS:2007:1" colorDomain="color">
             <Name xml:lang="en">H.264 Baseline Profile @ Level 1.1</Name>
            </Format>
            <Frame height="576" width="704" rate="30"/>
           </VisualCoding>
          </MediaFormat>
         </ComponentMediaProfile>
         <ComponentMediaProfile>
          <MediaFormat>
           <Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:1">
            <Name>Audio</Name>
           </Content>
           <BitRate>64000</BitRate>
           <AudioCoding>
            <Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.3.1">
             <Name xml:lang="en">MPEG-2 Audio AAC Low Complexity Profile</Name>
            </Format>
           </AudioCoding>
          </MediaFormat>
         </ComponentMediaProfile>
         <MediaFormat>
          <Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:2">
           <Name>Audiovisual</Name>
          </Content>
          <FileFormat href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:5">
           <Name>MP4</Name>
          </FileFormat>
          <FileSize>995902</FileSize>
          <BitRate>420864</BitRate>
         </MediaFormat>
        </MediaProfile>
       </MediaInformation>
      </Description>
     </Mpeg7>
    </Statement>
   </Descriptor>
   <Resource mimeType="video/avi" ref="he_asked_playable.mp4"/>
  </Component>

 </Item>
</DIDL>
```

Listing 20: *Scalable resource annotated with the AdaptationQoS*

UED DI

Listing 21 shows the MPEG-21 Part 7 UED tools that CAIN-21 utilizes. The target namespace of these elements is *urn:vpu:cain21-description-elements*, which is shortened to *cde*. The *cde:TerminalsType* description tool restricts the cardinality of the standard *dia:TerminalsType* description tool to force the existence of at least one terminal. In contrast, the *cde:NetworksType* description tool and *cde:UsersType* description tool maintain the cardinality of the base *dia:NetworksType* and *dia:UsersType* description tools (i.e., from 0 to *n* instances).

In the terminal description, CAIN-21 adds the *cde:HandlerCapabilitiesType* description tool to indicate the binding mode. The *cde:CodecCapabilitiesType* description tool is restricted to accepting only the *Decoding* description element. This is because currently, in CAIN-21, terminals only decode media. The standard *dia:CodecCapabilityBaseType* description tool has been extended to indicate the type of the media. In particular, CAIN-21 extends this element with the *cde:ImageCapabilitiesType*, *cde:VideoCapabilitiesType* and *cde:AudioCapabilitiesType* description tools. The cardinality of the *Display* element in the *DisplaysType* description tool has been restricted to accepting one and only one display. This change was made because currently CAIN-21 assumes exactly one display per terminal.

In the network description, the *cde:NetworkType* description tool has been restricted to accepting one and only one instance of the standard *dia:NetworkCharacteristic*. In addition the description tool of this element (i.e., *cde:NetworkCapabilityType*) has been restricted to accepting only the elements that CAIN-21 uses (namely, the attributes *minGuaranteed* and *maxCapacity*).

In reference to the user, the content of the *UsagePreferences* element has been restricted. In particular, the *cde:FilteringAndSearchPreferencesType* description tool restricts the standard *mpeg7:FilteringAndSearchPreferencesType* description tool to accepting at most one instance of the *cde:SourcePreferencesType* and *cde:AdaptationPreferencesType* description tools. These elements, in turn, restrict the standard *mpeg7:SourcePreferencesType* and *mpeg7:AdaptationPreferencesType* description tools to refer only to the elements that CAIN-21 is able to utilize. With respect to the media format preferences, we have extended the existing *mpeg7:MediaFormat*, *mpeg7:VisualCoding* and *mpeg7:AudioCoding* descriptors with the inclusion of a new attribute named *preferenceValue*.

The content of the *mpeg21:ConverstionPreferenceType* description tool has been restricted because CAIN-21 currently only accepts one instance of the *mpeg21:GeneralResourceConversions* element (instances of the *mpeg21:SpecificResourceConversions* element are not allowed). Finally, the *mpeg21:ConversionType* description tool has been restricted in order to remove the *weight* attribute, i.e., only use the *order* attribute (see Subsection 3.2.3 of Chapter 6).

```
<?xml version="1.0" ?>
<xsd:schema targetNamespace="urn:vpu:cain21-description-elements"
            elementFormDefault="qualified"
            attributeFormDefault="unqualified"
            xmlns="urn:vpu:cain21-description-elements"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS"
            xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
            id="cde.xsd">

 <!-- The following import elements are just hints for
      the types and elements used by this schema -->
 <!-- ConvCapab-AMD1.xsd includes DIA.xsd" -->
 <xsd:import namespace="urn:mpeg:mpeg21:2003:01-DIA-NS"
             schemaLocation="ConvCapab-AMD1.xsd"/>
 <xsd:import namespace="urn:mpeg:mpeg21:2003:01-DIA-NS"
             schemaLocation="cde-aux.xsd"/>
 <xsd:import namespace="urn:mpeg:mpeg7:schema:2001"
             schemaLocation="mpeg7-udp-2004.xsd"/>

 <!-- Terminal capabilities -->

 <xsd:complexType name="TerminalsType">
  <xsd:complexContent>
   <xsd:restriction base="dia:TerminalsType">
    <xsd:sequence>
     <xsd:element ref="dia:Terminal" minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
   </xsd:restriction>
  </xsd:complexContent>
 </xsd:complexType>

 <xsd:complexType name="TerminalType">
  <xsd:complexContent>
   <xsd:restriction base="dia:TerminalType">
    <xsd:sequence>
     <xsd:element ref="dia:TerminalCapability"
                  minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:ID" use="required"/>
   </xsd:restriction>
  </xsd:complexContent>
 </xsd:complexType>

 <!-- Terminal capabilities:  HandlerCapabilitiesType-->

 <xsd:complexType name="HandlerCapabilitiesType">
  <xsd:complexContent>
```

```
   <xsd:extension base="dia:TerminalCapabilityBaseType">
    <xsd:sequence>
     <xsd:element ref="dia:Handler"
                  minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>

 <!-- Terminal capabilities:  CodecCapabilityBaseType-->

 <xsd:complexType name="CodecCapabilityBaseType" abstract="true">
    <xsd:complexContent>
      <xsd:extension base="dia:DIABaseType">
        <xsd:sequence minOccurs="0" maxOccurs="unbounded">
          <xsd:element name="Format" type="mpeg7:ControlledTermUseType"/>
          <xsd:element name="CodecParameter" type="dia:CodecParameterBaseType"
                       minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="optional" type="xsd:boolean" use="optional"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <!-- Terminal capabilities:  CodecCapabilitiesType-->

 <xsd:complexType name="CodecCapabilitiesType">
  <xsd:complexContent>
   <xsd:restriction base="dia:CodecCapabilitiesType">
    <xsd:sequence>
     <xsd:element name="Decoding" type="CodecCapabilityBaseType" minOccurs="1" maxOc-
curs="unbounded"/>
    </xsd:sequence>
   </xsd:restriction>
  </xsd:complexContent>
 </xsd:complexType>

  <!-- File format capabilities:  TransportCapabilitiesType -->

 <xsd:complexType name="TransportCapabilitiesType">
    <xsd:complexContent>
      <xsd:extension base="CodecCapabilityBaseType"/>
    </xsd:complexContent>
  </xsd:complexType>

  <!-- Image capabilities:  ImageCapabilitiesType -->

 <xsd:complexType name="ImageCapabilitiesType">
    <xsd:complexContent>
      <xsd:extension base="CodecCapabilityBaseType"/>
    </xsd:complexContent>
  </xsd:complexType>

  <!-- Video capabilities:  VideoCapabilitiesType -->

 <xsd:complexType name="VideoCapabilitiesType">
    <xsd:complexContent>
      <xsd:extension base="CodecCapabilityBaseType"/>
    </xsd:complexContent>
  </xsd:complexType>

  <!-- Audio capabilities:  VideoCapabilitiesType -->

 <xsd:complexType name="AudioCapabilitiesType">
```

```
    <xsd:complexContent>
      <xsd:extension base="CodecCapabilityBaseType"/>
    </xsd:complexContent>
  </xsd:complexType>

<!-- Terminal capabilities:  DisplaysType-->

<xsd:complexType name="DisplaysType">
  <xsd:complexContent>
   <xsd:restriction base="dia:DisplaysType">
    <xsd:sequence>
     <xsd:element ref="dia:Display" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
   </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="DisplayType">
  <xsd:complexContent>
   <xsd:restriction base="dia:DisplayType">
    <xsd:sequence>
     <xsd:element ref="dia:DisplayCapability"
                  minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
   </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="DisplayCapabilityType">
  <xsd:complexContent>
   <xsd:restriction base="dia:DisplayCapabilityType">
    <xsd:sequence>
     <xsd:element ref="dia:Mode"
                  minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
   </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>

<!-- Network description -->

<xsd:complexType name="NetworksType">
  <xsd:complexContent>
   <xsd:restriction base="dia:NetworksType">
    <xsd:sequence>
     <xsd:element ref="dia:Network"
                  minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
   </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="NetworkType">
  <xsd:complexContent>
   <xsd:restriction base="dia:NetworkType">
    <xsd:sequence>
     <xsd:element ref="dia:NetworkCharacteristic"
                  minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
   </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="NetworkCapabilityType">
```

```
  <xsd:complexContent>
   <xsd:restriction base="dia:NetworkCapabilityType">
    <xsd:attribute name="maxCapacity" type="xsd:nonNegativeInteger"
                   use="optional"/>
    <xsd:attribute name="minGuaranteed"
                   type="xsd:nonNegativeInteger" use="optional"/>
   </xsd:restriction>
  </xsd:complexContent>
 </xsd:complexType>

 <!-- User caracteristics -->

 <xsd:complexType name="UsersType">
  <xsd:complexContent>
   <xsd:restriction base="dia:UsersType">
    <xsd:sequence>
     <xsd:element ref="dia:User" minOccurs="0"
                  maxOccurs="unbounded"/>
    </xsd:sequence>
   </xsd:restriction>
  </xsd:complexContent>
 </xsd:complexType>

 <!-- User caracteristics: UsagePreferencesType -->

 <xsd:complexType name="FilteringAndSearchPreferencesType">
  <xsd:complexContent>
   <xsd:extension base="mpeg7:FilteringAndSearchPreferencesType">
    <xsd:sequence>
     <xsd:element name="SourcePreferences" type="SourcePreferencesType"
                  minOccurs="0" maxOccurs="1"/>
     <xsd:element name="AdaptationPreferences" type="AdaptationPreferencesType"
                  minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>

 <!-- SourcePreferencesType -->

 <xsd:complexType name="SourcePreferencesType">
  <xsd:complexContent>
   <xsd:extension base="mpeg7:SourcePreferencesType">
    <xsd:sequence>
     <xsd:element name="MediaFormat" type="MediaFormatType"
                  minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>

 <xsd:complexType name="MediaFormatType">
  <xsd:complexContent>
   <xsd:extension base="mpeg7:MediaFormatType">
   <xsd:sequence>
    <xsd:element name="FileFormat" type="FileFormatType"
                 minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="VisualCoding" type="CodingType"
                    minOccurs="0" maxOccurs="1"/>
    <xsd:element name="AudioCoding" type="CodingType"
                    minOccurs="0" maxOccurs="1"/>
   </xsd:sequence>
   <xsd:attribute name="preferenceValue"
                   type="mpeg7:preferenceValueType" use="required"/>
 </xsd:extension>
 </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="FileFormatType">
 <xsd:complexContent>
  <xsd:extension base="mpeg7:ControlledTermUseType">
   <xsd:sequence>
    <xsd:element name="Name" type="xsd:string"
                 minOccurs="1" maxOccurs="1"/>
   </xsd:sequence>
   <xsd:attribute name="preferenceValue"
                  type="mpeg7:preferenceValueType" use="required"/>
  </xsd:extension>
 </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="CodingType">
  <xsd:sequence>
   <xsd:element name="Format" type="FormatType"
                minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="preferenceValue"
                 type="mpeg7:preferenceValueType" use="required"/>
</xsd:complexType>

<xsd:complexType name="FormatType">
 <xsd:sequence>
  <xsd:element name="Name"
               type="mpeg7:TextualType" minOccurs="1" maxOccurs="1"/>
 </xsd:sequence>
 <xsd:attribute name="href"
                type="mpeg7:termReferenceType" use="required"/>
 <xsd:attribute name="preferenceValue"
                type="mpeg7:preferenceValueType" use="required"/>
</xsd:complexType>

<!-- AdaptationPreferencesType -->

<xsd:complexType name="AdaptationPreferencesType">
  <xsd:sequence>
   <xsd:element name="ContentDegradation" type="AdaptationPreferenceType"
                  minOccurs="0" maxOccurs="1"/>
   <xsd:element name="Online" type="AdaptationPreferenceType"
                  minOccurs="0" maxOccurs="1"/>
   <xsd:element name="ExecutionCost" type="AdaptationPreferenceType"
                  minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="AdaptationPreferenceType">
 <xsd:attribute name="preferenceValue"
                type="mpeg7:preferenceValueType" use="required"/>
</xsd:complexType>

<!-- User caracteristics: ConversionPreferenceType -->

<xsd:complexType name="ConversionPreferenceType">
 <xsd:complexContent>
  <xsd:restriction base="dia:ConversionPreferenceType">
   <xsd:sequence>
    <xsd:element ref="dia:GeneralResourceConversions"
                 minOccurs="1" maxOccurs="1"/>
   </xsd:sequence>
   </xsd:restriction>
  </xsd:complexContent>
 </xsd:complexType>

<xsd:complexType name="ConversionType">
 <xsd:complexContent>
  <xsd:restriction base="dia:ConversionType">
   <xsd:sequence>
    <xsd:element name="From" type="mpeg7:ControlledTermUseType"
                 minOccurs="1" maxOccurs="1"/>
    <xsd:element name="To" type="mpeg7:ControlledTermUseType"
                 minOccurs="1" maxOccurs="1"/>
   </xsd:sequence>
   <xsd:attribute name="order" type="xsd:nonNegativeInteger" use="required"/>
   <xsd:attribute name="weight" type="mpeg7:nonNegativeReal" use="prohibited"/>
  </xsd:restriction>
 </xsd:complexContent>
</xsd:complexType>

</xsd:schema>
```

Listing 21: UED XML Schema of CAIN-21

Listing 22 shows the current usage environment description elements of CAIN-21. Under the *cde:TerminalsType* description tool we find the description of 29 different terminals.

There are 5 network profiles (namely *modem*, *umts_3g*, *wifi*, *adsl* and *ethernet*), which represent different network conditions. Some of them provide a minimum bandwidth guarantee and others only provide the maximum capacity of the network.

Finally, There are 3 user profiles to describe the user's preferences. The *roi_faces_preference* shows how to indicate that the user is interested in the ROIs. The *fast_adaptation* preference shows how to indicate that, if possible, the user prefers online adaptation (*preferenceValue="80"*), rather than reducing the number of conversion steps (*preferenceValue="50"*) or reducing the content degradation (*preferenceValue="30"*). The *cain21* profile contains the adaptation engine default preferences. In these preferences, CAIN-21 states that (by default) it is preferable to produce an MPEG file format rather than an AVI or MP4 file format. It also states that the MPEG-2 visual stream is preferable to other MPEG-4 visual streams, and that the MPEG-2 Audio is preferable to the AMR audio. In addition, CAIN-21 defines as more preferable reducing the content degradation (*preferenceValue="80"*) than minimizing the number of conversions (*preferenceValue="50"*) or executing the adaptation online (*preferenceValue="30"*).

```
<?xml version="1.0" ?>

<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"
     xmlns:cde="urn:vpu:cain21-description-elements">
 <Description xsi:type="UsageEnvironmentType">
  <!-- Terminals descriptions -->
  <UsageEnvironmentProperty xsi:type="cde:TerminalsType">

  <Terminal id="multi_audio_device">
    <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
     <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
```

```
     <cde:Decoding xsi:type="cde:AudioCapabilitiesType">
      <cde:Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.3">
       <mpeg7:Name xml:lang="en">
        MPEG-2 Audio AAC
       </mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
       <BitRate>96000</BitRate>
      </cde:CodecParameter>
      <cde:Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:1">
       <mpeg7:Name xml:lang="en">
        Dolby AC3
       </mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
       <BitRate>96000</BitRate>
      </cde:CodecParameter>
     </cde:Decoding>
    </TerminalCapability>
   </Terminal>

  <Terminal id="mp3_only_audio_device">
    <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
     <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
     <cde:Decoding xsi:type="cde:AudioCapabilitiesType">
      <cde:Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.4">
       <mpeg7:Name xml:lang="en">
        MP3 Audio Coding Format
       </mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
       <BitRate>96000</BitRate>
      </cde:CodecParameter>
     </cde:Decoding>
    </TerminalCapability>
   </Terminal>

  <Terminal id="aac_only_audio_device">
    <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
     <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
     <cde:Decoding xsi:type="cde:AudioCapabilitiesType">
      <cde:Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.3">
       <mpeg7:Name xml:lang="en">
        MPEG-2 Audio AAC
       </mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
       <BitRate>96000</BitRate>
      </cde:CodecParameter>
     </cde:Decoding>
    </TerminalCapability>
   </Terminal>

  <Terminal id="ac3_only_audio_device">
    <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
     <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
     <cde:Decoding xsi:type="cde:AudioCapabilitiesType">
      <cde:Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:1">
      <mpeg7:Name xml:lang="en">
       Dolby AC3
      </mpeg7:Name>
     </cde:Format>
     <cde:CodecParameter xsi:type="CodecParameterBitRateType">
      <BitRate>96000</BitRate>
     </cde:CodecParameter>
    </cde:Decoding>
   </TerminalCapability>
  </Terminal>

 <Terminal id="wav_only_audio_device">
   <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
    <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
   </TerminalCapability>
   <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
    <cde:Decoding xsi:type="cde:AudioCapabilitiesType">
     <cde:Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:8">
      <mpeg7:Name xml:lang="en">
       Linear PCM
      </mpeg7:Name>
     </cde:Format>
     <cde:CodecParameter xsi:type="CodecParameterBitRateType">
      <BitRate>96000</BitRate>
     </cde:CodecParameter>
    </cde:Decoding>
   </TerminalCapability>
  </Terminal>

  <Terminal id="gray_images_viewer" xsi:type="cde:TerminalType">
   <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
    <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
   </TerminalCapability>
   <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
    <cde:Decoding xsi:type="cde:ImageCapabilitiesType">
     <cde:Format
            href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:4">
      <mpeg7:Name xml:lang="en">
       JPEG
      </mpeg7:Name>
     </cde:Format>
     <cde:CodecParameter xsi:type="CodecParameterBitRateType">
      <BitRate >32000</BitRate>
     </cde:CodecParameter>
    </cde:Decoding>
   </TerminalCapability>
   <TerminalCapability xsi:type="cde:DisplaysType">
    <Display>
     <DisplayCapability xsi:type="cde:DisplayCapabilityType"
                   colorCapable="false">
      <Mode>
       <Resolution horizontal="176" vertical="144"/>
      </Mode>
     </DisplayCapability>
    </Display>
   </TerminalCapability>
  </Terminal>

  <Terminal id="jpeg_images_viewer" xsi:type="cde:TerminalType">
   <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
    <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
   </TerminalCapability>
   <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
    <cde:Decoding xsi:type="cde:ImageCapabilitiesType">
```

```
      <cde:Format
              href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:4">
       <mpeg7:Name xml:lang="en">
        JPEG
       </mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
       <BitRate >32000</BitRate>
      </cde:CodecParameter>
     </cde:Decoding>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:DisplaysType">
     <Display>
      <DisplayCapability xsi:type="cde:DisplayCapabilityType"
                         colorCapable="true">
       <Mode>
        <Resolution horizontal="352" vertical="288"/>
       </Mode>
      </DisplayCapability>
     </Display>
    </TerminalCapability>
   </Terminal>

   <Terminal id="bmp_images_viewer" xsi:type="cde:TerminalType">
    <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
     <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
     <cde:Decoding xsi:type="cde:ImageCapabilitiesType">
      <cde:Format
              href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:bmp">
       <mpeg7:Name xml:lang="en">
        BMP
       </mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
       <BitRate >32000</BitRate>
      </cde:CodecParameter>
     </cde:Decoding>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:DisplaysType">
     <Display>
      <DisplayCapability xsi:type="cde:DisplayCapabilityType"
                         colorCapable="true">
       <Mode>
        <Resolution horizontal="352" vertical="288"/>
       </Mode>
      </DisplayCapability>
     </Display>
    </TerminalCapability>
   </Terminal>

  <Terminal id="gif_images_viewer" xsi:type="cde:TerminalType">
    <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
     <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
     <cde:Decoding xsi:type="cde:ImageCapabilitiesType">
      <cde:Format
              href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:gif">
       <mpeg7:Name xml:lang="en">
        GIF
       </mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
       <BitRate >32000</BitRate>
      </cde:CodecParameter>
     </cde:Decoding>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:DisplaysType">
     <Display>
      <DisplayCapability xsi:type="cde:DisplayCapabilityType"
                        colorCapable="true">
       <Mode>
        <Resolution horizontal="640" vertical="480"/>
       </Mode>
      </DisplayCapability>
     </Display>
    </TerminalCapability>
   </Terminal>

  <Terminal id="png_images_viewer" xsi:type="cde:TerminalType">
    <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
     <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
     <cde:Decoding xsi:type="cde:ImageCapabilitiesType">
      <cde:Format
              href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:png">
       <mpeg7:Name xml:lang="en">
        PNG
       </mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
       <BitRate >32000</BitRate>
      </cde:CodecParameter>
     </cde:Decoding>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:DisplaysType">
     <Display>
      <DisplayCapability xsi:type="cde:DisplayCapabilityType"
                        colorCapable="true">
       <Mode>
        <Resolution horizontal="510" vertical="320"/>
       </Mode>
      </DisplayCapability>
     </Display>
    </TerminalCapability>
   </Terminal>

   <Terminal id="images_viewer_duplicate_resolution" xsi:type="cde:TerminalType">
    <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
     <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
     <cde:Decoding xsi:type="cde:ImageCapabilitiesType">
      <cde:Format
           href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:gif">
       <mpeg7:Name xml:lang="en">
        GIF
       </mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
       <BitRate >32000</BitRate>
      </cde:CodecParameter>
     </cde:Decoding>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:DisplaysType">
    <Display>
     <DisplayCapability xsi:type="cde:DisplayCapabilityType"
                       colorCapable="true">
      <Mode>
       <Resolution horizontal="510" vertical="320"/>
                 <Resolution horizontal="320" vertical="256"/>
      </Mode>
     </DisplayCapability>
    </Display>
   </TerminalCapability>
  </Terminal>

 <Terminal id="images_viewer_width_resolution" xsi:type="cde:TerminalType">
   <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
    <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
   </TerminalCapability>
   <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
    <cde:Decoding xsi:type="cde:ImageCapabilitiesType">
     <cde:Format
          href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:bmp">
      <mpeg7:Name xml:lang="en">
       BMP
      </mpeg7:Name>
     </cde:Format>
     <cde:CodecParameter xsi:type="CodecParameterBitRateType">
      <BitRate >32000</BitRate>
     </cde:CodecParameter>
    </cde:Decoding>
   </TerminalCapability>
   <TerminalCapability xsi:type="cde:DisplaysType">
    <Display>
     <DisplayCapability xsi:type="cde:DisplayCapabilityType"
                       colorCapable="true">
      <Mode>
       <Resolution horizontal="120" vertical="340"/>

      </Mode>
     </DisplayCapability>
    </Display>
   </TerminalCapability>
  </Terminal>

<Terminal id="images_viewer_height_resolution" xsi:type="cde:TerminalType">
   <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
    <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
   </TerminalCapability>
   <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
    <cde:Decoding xsi:type="cde:ImageCapabilitiesType">
     <cde:Format
          href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:png">
      <mpeg7:Name xml:lang="en">
       PNG
      </mpeg7:Name>
     </cde:Format>
     <cde:CodecParameter xsi:type="CodecParameterBitRateType">
      <BitRate >32000</BitRate>
     </cde:CodecParameter>
    </cde:Decoding>
   </TerminalCapability>
   <TerminalCapability xsi:type="cde:DisplaysType">
    <Display>
     <DisplayCapability xsi:type="cde:DisplayCapabilityType"
                       colorCapable="true">
      <Mode>
```

```
        <Resolution horizontal="300" vertical="100"/>

       </Mode>
      </DisplayCapability>
     </Display>
    </TerminalCapability>
   </Terminal>

  <Terminal id="images_viewer_several_formats" xsi:type="cde:TerminalType">
    <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
     <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
     <cde:Decoding xsi:type="cde:ImageCapabilitiesType">
      <cde:Format
              href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:png">
       <mpeg7:Name xml:lang="en">
        PNG
       </mpeg7:Name>
      </cde:Format>
              <cde:Format
              href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:gif">
       <mpeg7:Name xml:lang="en">
        GIF
       </mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
       <BitRate >32000</BitRate>
      </cde:CodecParameter>
     </cde:Decoding>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:DisplaysType">
     <Display>
      <DisplayCapability xsi:type="cde:DisplayCapabilityType"
                        colorCapable="true">
       <Mode>
              <Resolution horizontal="100" vertical="100"/>
       </Mode>
      </DisplayCapability>
     </Display>
    </TerminalCapability>
   </Terminal>

  <Terminal id="images_viewer_without_resolution" xsi:type="cde:TerminalType">
    <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
     <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
     <cde:Decoding xsi:type="cde:ImageCapabilitiesType">
      <cde:Format
              href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:png">
       <mpeg7:Name xml:lang="en">
        PNG
       </mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
       <BitRate >32000</BitRate>
      </cde:CodecParameter>
     </cde:Decoding>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:DisplaysType">
     <Display>
      <DisplayCapability xsi:type="cde:DisplayCapabilityType"
                        colorCapable="true">
      </DisplayCapability>
     </Display>
    </TerminalCapability>
   </Terminal>

   <Terminal id="iphone" xsi:type="cde:TerminalType">
    <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
     <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
     <cde:Decoding xsi:type="cde:TransportCapabilitiesType">
                <cde:Format href="urn:vpu:cs:FileFormatCS:2009:3gp">
       <mpeg7:Name xml:lang="en">
        3GPP file format
       </mpeg7:Name>
      </cde:Format>
               </cde:Decoding>
     <cde:Decoding xsi:type="cde:VideoCapabilitiesType">
      <cde:Format
              href="urn:vpu:cs:VisualCodingFormatCS:2007:1">
       <mpeg7:Name xml:lang="en">
          H.264 Baseline Profile @ Level 1.1
       </mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
       <BitRate >32000</BitRate>
     </cde:CodecParameter>
     </cde:Decoding>
     <cde:Decoding xsi:type="cde:AudioCapabilitiesType" optional="true">
      <cde:Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.3.1">
       <mpeg7:Name xml:lang="en">
        MPEG-2 Audio AAC Low Complexity Profile
       </mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
       <BitRate>7950</BitRate>
     </cde:CodecParameter>
     </cde:Decoding>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:DisplaysType">
     <Display>
      <DisplayCapability xsi:type="cde:DisplayCapabilityType"
                         colorCapable="true">
       <Mode refreshRate="15">
        <Resolution horizontal="240" vertical="160"/>
       </Mode>
      </DisplayCapability>
     </Display>
    </TerminalCapability>
   </Terminal>

  <Terminal id="http_nokia_n95" xsi:type="cde:TerminalType">
    <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
     <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:HTTP"/>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
     <cde:Decoding xsi:type="cde:TransportCapabilitiesType">
                <cde:Format
                        href="urn:vpu:cs:FileFormatCS:2009:3gp">
                 <mpeg7:Name xml:lang="en">
                  3GPP file format
                 </mpeg7:Name>
                </cde:Format>
               </cde:Decoding>
     <cde:Decoding xsi:type="cde:VideoCapabilitiesType">
      <cde:Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.1">
       <mpeg7:Name xml:lang="en">MPEG-4 Visual Simple Profile</mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
       <BitRate>192000</BitRate>
     </cde:CodecParameter>
     </cde:Decoding>
     <cde:Decoding xsi:type="cde:AudioCapabilitiesType" optional="true">
      <cde:Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.3.1">
       <mpeg7:Name xml:lang="en">
        MPEG-2 Audio AAC Low Complexity Profile
       </mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
       <BitRate>48000</BitRate>
     </cde:CodecParameter>
     </cde:Decoding>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:DisplaysType">
     <Display>
      <DisplayCapability xsi:type="cde:DisplayCapabilityType"
                         colorCapable="true">
       <Mode refreshRate="15">
        <Resolution horizontal="160" vertical="120"/>
       </Mode>
      </DisplayCapability>
     </Display>
    </TerminalCapability>
   </Terminal>

   <Terminal id="flash_player" xsi:type="cde:TerminalType">
    <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
     <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
     <cde:Decoding xsi:type="cde:TransportCapabilitiesType">
                <cde:Format
                        href="urn:vpu:cs:FileFormatCS:2009:swf">
                 <mpeg7:Name xml:lang="en">
                  Flash file format
                 </mpeg7:Name>
                </cde:Format>
               </cde:Decoding>
     <cde:Decoding xsi:type="cde:VideoCapabilitiesType">
      <cde:Format href="urn:vpu:cs:VisualCodingFormatCS:2009:flv">
       <mpeg7:Name xml:lang="en">Macromedia Flash video stream</mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
       <BitRate>192000</BitRate>
     </cde:CodecParameter>
     </cde:Decoding>
     <cde:Decoding xsi:type="cde:AudioCapabilitiesType" optional="true">
      <cde:Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.4">
       <mpeg7:Name xml:lang="en">
        MPEG-2 Audio AAC Low Complexity Profile
       </mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
       <BitRate>32000</BitRate>
     </cde:CodecParameter>
     </cde:Decoding>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:DisplaysType">
    <Display>
     <DisplayCapability xsi:type="cde:DisplayCapabilityType"
                        colorCapable="true">
      <Mode refreshRate="25">
       <Resolution horizontal="320" vertical="240"/>
      </Mode>
     </DisplayCapability>
    </Display>
   </TerminalCapability>
  </Terminal>

  <Terminal id="mp4_desktop">
   <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
    <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
    <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:HTTP"/>
   </TerminalCapability>
   <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
    <cde:Decoding xsi:type="cde:TransportCapabilitiesType">
               <cde:Format
                       href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:5">
                <mpeg7:Name xml:lang="en">
                 MPEG-4 file format
                </mpeg7:Name>
               </cde:Format>
              </cde:Decoding>
    <cde:Decoding xsi:type="cde:VideoCapabilitiesType">
     <cde:Format
             href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.1">
      <mpeg7:Name xml:lang="en">
       MPEG-4 Visual Simple Profile
      </mpeg7:Name>
     </cde:Format>
     <cde:CodecParameter xsi:type="CodecParameterBitRateType">
      <BitRate>320000</BitRate>
    </cde:CodecParameter>
     <cde:Format
             href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.3">
      <mpeg7:Name xml:lang="en">
       MPEG-4 Visual Advanced Simple Profile
      </mpeg7:Name>
     </cde:Format>
     <cde:CodecParameter xsi:type="CodecParameterBitRateType">
      <BitRate>320000</BitRate>
    </cde:CodecParameter>
    </cde:Decoding>
    <cde:Decoding xsi:type="cde:AudioCapabilitiesType" optional="true">
     <cde:Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.3.1">
      <mpeg7:Name xml:lang="en">
       MPEG-2 Audio AAC
      </mpeg7:Name>
     </cde:Format>
     <cde:CodecParameter xsi:type="CodecParameterBitRateType">
      <BitRate>64000</BitRate>
    </cde:CodecParameter>
    </cde:Decoding>
   </TerminalCapability>
   <TerminalCapability xsi:type="cde:DisplaysType">
    <Display>
     <DisplayCapability xsi:type="cde:DisplayCapabilityType"
                        colorCapable="true">
      <Mode refreshRate="30">
       <Resolution horizontal="352" vertical="288"/>
       <Resolution horizontal="176" vertical="144"/>
      </Mode>
     </DisplayCapability>
    </Display>
   </TerminalCapability>
  </Terminal>

  <Terminal id="h264_desktop">
   <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
    <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
    <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:HTTP"/>
   </TerminalCapability>
   <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
    <cde:Decoding xsi:type="cde:TransportCapabilitiesType">
               <cde:Format
                       href="urn:vpu:cs:FileFormatCS:2009:3gp">
                <mpeg7:Name xml:lang="en">
                 3GPP file format
                </mpeg7:Name>
               </cde:Format>
              </cde:Decoding>
    <cde:Decoding xsi:type="cde:VideoCapabilitiesType">
     <cde:Format href="urn:vpu:cs:VisualCodingFormatCS:2007:1">
      <mpeg7:Name xml:lang="en">H.264 Baseline Profile @ Level 1.1</mpeg7:Name>
     </cde:Format>
     <cde:CodecParameter xsi:type="CodecParameterBitRateType">
      <BitRate>672000</BitRate>
    </cde:CodecParameter>
    </cde:Decoding>
    <cde:Decoding xsi:type="cde:AudioCapabilitiesType" optional="true">
     <cde:Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.3.1">
      <mpeg7:Name xml:lang="en">
       MPEG-2 Audio AAC Low Complexity Profile
      </mpeg7:Name>
     </cde:Format>
     <cde:CodecParameter xsi:type="CodecParameterBitRateType">
      <BitRate>96000</BitRate>
    </cde:CodecParameter>
    </cde:Decoding>
   </TerminalCapability>
   <TerminalCapability xsi:type="cde:DisplaysType">
    <Display>
     <DisplayCapability xsi:type="cde:DisplayCapabilityType"
                        colorCapable="true">
      <Mode refreshRate="25">
       <Resolution horizontal="240" vertical="180"/>
      </Mode>
     </DisplayCapability>
    </Display>
   </TerminalCapability>
  </Terminal>

   <Terminal id="mpeg1_desktop">
   <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
    <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
   </TerminalCapability>
   <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
    <cde:Decoding xsi:type="cde:TransportCapabilitiesType">
               <cde:Format
                       href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:3">
                <mpeg7:Name xml:lang="en">
                 MPEG file format
                </mpeg7:Name>
               </cde:Format>
              </cde:Decoding>
    <cde:Decoding xsi:type="cde:VideoCapabilitiesType">
    <cde:Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:1">
     <mpeg7:Name xml:lang="en">MPEG-1 Video Coding Format</mpeg7:Name>
    </cde:Format>
    <cde:CodecParameter xsi:type="CodecParameterBitRateType">
     <BitRate>672000</BitRate>
    </cde:CodecParameter>
   </cde:Decoding>
   <cde:Decoding xsi:type="cde:AudioCapabilitiesType" optional="true">
    <cde:Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:3.2">
     <mpeg7:Name xml:lang="en">
    MPEG-1 Audio Layer II
     </mpeg7:Name>
    </cde:Format>
    <cde:CodecParameter xsi:type="CodecParameterBitRateType">
     <BitRate>64000</BitRate>
    </cde:CodecParameter>
   </cde:Decoding>
  </TerminalCapability>
  <TerminalCapability xsi:type="cde:DisplaysType">
   <Display>
    <DisplayCapability xsi:type="cde:DisplayCapabilityType"
                       colorCapable="true">
     <Mode refreshRate="25">
      <Resolution horizontal="360" vertical="288"/>
     </Mode>
    </DisplayCapability>
   </Display>
  </TerminalCapability>
 </Terminal>

<Terminal id="mpeg2_medium_desktop">
  <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
   <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
  </TerminalCapability>
  <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
   <cde:Decoding xsi:type="cde:TransportCapabilitiesType">
              <cde:Format
                      href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:7">
               <mpeg7:Name xml:lang="en">
                AVI file format
               </mpeg7:Name>
              </cde:Format>
             </cde:Decoding>
   <cde:Decoding xsi:type="cde:VideoCapabilitiesType">
    <cde:Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:2.2.2">
     <mpeg7:Name xml:lang="en">MPEG-2 Video Main Profile @ Main Level</mpeg7:Name>
    </cde:Format>
    <cde:CodecParameter xsi:type="CodecParameterBitRateType">
     <BitRate>100000</BitRate>
    </cde:CodecParameter>
   </cde:Decoding>
   <cde:Decoding xsi:type="cde:AudioCapabilitiesType" optional="true">
    <cde:Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4">
     <mpeg7:Name xml:lang="en">
      MP2 Audio Coding Format
     </mpeg7:Name>
    </cde:Format>
    <cde:CodecParameter xsi:type="CodecParameterBitRateType">
     <BitRate>96000</BitRate>
    </cde:CodecParameter>
   </cde:Decoding>

  </TerminalCapability>
  <TerminalCapability xsi:type="cde:DisplaysType">
```

```
     <Display>
      <DisplayCapability xsi:type="cde:DisplayCapabilityType"
                         colorCapable="true">
       <Mode refreshRate="25">
        <Resolution horizontal="352" vertical="288"/>
       </Mode>
      </DisplayCapability>
     </Display>
    </TerminalCapability>
   </Terminal>

   <Terminal id="mpeg2_small_desktop">
    <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
     <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
     <cde:Decoding xsi:type="cde:TransportCapabilitiesType">
                <cde:Format
                        href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:7">
                 <mpeg7:Name xml:lang="en">
                  AVI file format
                 </mpeg7:Name>
                </cde:Format>
               </cde:Decoding>
     <cde:Decoding xsi:type="cde:VideoCapabilitiesType">
      <cde:Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:2.2.2">
       <mpeg7:Name xml:lang="en">MPEG-2 Video Main Profile @ Main Level</mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
       <BitRate>20000</BitRate>
      </cde:CodecParameter>
     </cde:Decoding>
     <cde:Decoding xsi:type="cde:AudioCapabilitiesType" optional="true">
      <cde:Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4">
       <mpeg7:Name xml:lang="en">
        MP2 Audio Coding Format
       </mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
       <BitRate>96000</BitRate>
      </cde:CodecParameter>
     </cde:Decoding>

    </TerminalCapability>
    <TerminalCapability xsi:type="cde:DisplaysType">
     <Display>
      <DisplayCapability xsi:type="cde:DisplayCapabilityType"
                         colorCapable="true">
       <Mode refreshRate="25">
        <Resolution horizontal="176" vertical="144"/>
       </Mode>
      </DisplayCapability>
     </Display>
    </TerminalCapability>
   </Terminal>

   <Terminal id="mpeg2_without_audio">
    <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
     <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
     <cde:Decoding xsi:type="cde:TransportCapabilitiesType">
                <cde:Format
                        href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:7">
                 <mpeg7:Name xml:lang="en">
                  AVI file format
                 </mpeg7:Name>
                </cde:Format>
               </cde:Decoding>
     <cde:Decoding xsi:type="cde:VideoCapabilitiesType">
                <cde:Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:2.2.2">
                 <mpeg7:Name xml:lang="en">MPEG-2 Video Main Profile @ Main
Level</mpeg7:Name>
                </cde:Format>
                <cde:CodecParameter xsi:type="CodecParameterBitRateType">
                 <BitRate>20000</BitRate>
                </cde:CodecParameter>
               </cde:Decoding>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:DisplaysType">
     <Display>
      <DisplayCapability xsi:type="cde:DisplayCapabilityType"
                         colorCapable="true">
       <Mode refreshRate="25">
        <Resolution horizontal="352" vertical="288"/>
       </Mode>
      </DisplayCapability>
     </Display>
    </TerminalCapability>
   </Terminal>

<Terminal id="mpeg1_720x576_adapted_online_web">
    <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
     <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:HTTP"/>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
     <cde:Decoding xsi:type="cde:TransportCapabilitiesType">
                <cde:Format
                        href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:7">
                 <mpeg7:Name xml:lang="en">
                  AVI file cde:Format
                 </mpeg7:Name>
                </cde:Format>
               </cde:Decoding>
     <cde:Decoding xsi:type="cde:VideoCapabilitiesType">
      <cde:Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:1">
       <mpeg7:Name xml:lang="en">MPEG-1 Video Coding cde:Format</mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
      </cde:CodecParameter>
     </cde:Decoding>
     <cde:Decoding xsi:type="cde:AudioCapabilitiesType">
      <cde:Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4">
       <mpeg7:Name xml:lang="en">
        MPEG-2 Audio Coding cde:Format
       </mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
      </cde:CodecParameter>
     </cde:Decoding>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:DisplaysType">
     <Display>
      <DisplayCapability xsi:type="cde:DisplayCapabilityType"
                         colorCapable="true">
       <Mode refreshRate="25">
        <Resolution horizontal="720" vertical="576"/>
       </Mode>
      </DisplayCapability>
     </Display>
    </TerminalCapability>
   </Terminal>

   <Terminal id="mpeg1_352x288_adapted_online_web">
    <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
     <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:HTTP"/>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
     <cde:Decoding xsi:type="cde:TransportCapabilitiesType">
                <cde:Format
                        href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:7">
                 <mpeg7:Name xml:lang="en">
                  AVI file cde:Format
                 </mpeg7:Name>
                </cde:Format>
               </cde:Decoding>
     <cde:Decoding xsi:type="cde:VideoCapabilitiesType">
      <cde:Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:1">
       <mpeg7:Name xml:lang="en">MPEG-1 Video Coding cde:Format</mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
      </cde:CodecParameter>
     </cde:Decoding>
     <cde:Decoding xsi:type="cde:AudioCapabilitiesType">
      <cde:Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4">
       <mpeg7:Name xml:lang="en">
        MPEG-2 Audio Coding cde:Format
       </mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
      </cde:CodecParameter>
     </cde:Decoding>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:DisplaysType">
     <Display>
      <DisplayCapability xsi:type="cde:DisplayCapabilityType"
                         colorCapable="true">
       <Mode refreshRate="25">
        <Resolution horizontal="352" vertical="288"/>
       </Mode>
      </DisplayCapability>
     </Display>
    </TerminalCapability>
   </Terminal>

   <Terminal id="mpeg1_176x144_adapted_online_web">
    <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
     <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:HTTP"/>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
     <cde:Decoding xsi:type="cde:TransportCapabilitiesType">
                <cde:Format
                        href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:7">
                 <mpeg7:Name xml:lang="en">
                  AVI file cde:Format
                 </mpeg7:Name>
                </cde:Format>
               </cde:Decoding>
     <cde:Decoding xsi:type="cde:VideoCapabilitiesType">
      <cde:Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:1">
       <mpeg7:Name xml:lang="en">MPEG-1 Video Coding cde:Format</mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
      </cde:CodecParameter>
     </cde:Decoding>
     <cde:Decoding xsi:type="cde:AudioCapabilitiesType">
      <cde:Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4">
       <mpeg7:Name xml:lang="en">
        MPEG-2 Audio Coding cde:Format
       </mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
      </cde:CodecParameter>
     </cde:Decoding>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:DisplaysType">
     <Display>
      <DisplayCapability xsi:type="cde:DisplayCapabilityType"
                         colorCapable="true">
       <Mode refreshRate="25">
        <Resolution horizontal="176" vertical="144"/>
       </Mode>
      </DisplayCapability>
     </Display>
    </TerminalCapability>
   </Terminal>
   <Terminal id="svc_no_audio_176x144_15fps">
    <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
     <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
     <cde:Decoding xsi:type="cde:TransportCapabilitiesType">
                <cde:Format
                        href="urn:vpu:cs:FileFormatCS:2009:svc">
                 <mpeg7:Name xml:lang="en">
                  SVC format
                 </mpeg7:Name>
                </cde:Format>
               </cde:Decoding>
     <cde:Decoding xsi:type="cde:VideoCapabilitiesType">
      <cde:Format href="urn:vpu:cs:VisualCodingFormatCS:2009:svc">
       <mpeg7:Name xml:lang="en">SVC visual format</mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
       <BitRate>20000</BitRate>
      </cde:CodecParameter>
     </cde:Decoding>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:DisplaysType">
     <Display>
      <DisplayCapability xsi:type="cde:DisplayCapabilityType"
                         colorCapable="true">
       <Mode refreshRate="15">
        <Resolution horizontal="176" vertical="144"/>
       </Mode>
      </DisplayCapability>
     </Display>
    </TerminalCapability>
   </Terminal>

   <Terminal id="svc_with_audio_352x288_15fps">
    <TerminalCapability xsi:type="cde:HandlerCapabilitiesType">
     <Handler handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:CodecCapabilitiesType">
     <cde:Decoding xsi:type="cde:TransportCapabilitiesType">
```

```
                <cde:Format
                        href="urn:vpu:cs:FileFormatCS:2009:svc">
                 <mpeg7:Name xml:lang="en">
                  SVC format
                 </mpeg7:Name>
                </cde:Format>
               </cde:Decoding>
               <cde:Decoding xsi:type="cde:TransportCapabilitiesType">
                <cde:Format
                        href="urn:vpu:cs:FileFormatCS:2009:svc_audio">
                 <mpeg7:Name xml:lang="en">
                  SVC format
                 </mpeg7:Name>
                </cde:Format>
               </cde:Decoding>
     <cde:Decoding xsi:type="cde:VideoCapabilitiesType">
      <cde:Format href="urn:vpu:cs:VisualCodingFormatCS:2009:svc">
       <mpeg7:Name xml:lang="en">SVC visual format</mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
       <BitRate>1500000</BitRate>
      </cde:CodecParameter>
     </cde:Decoding>
     <cde:Decoding xsi:type="cde:AudioCapabilitiesType" optional="true">
      <cde:Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.3.1">
       <mpeg7:Name xml:lang="en">
        MPEG-2 Audio AAC Low Complexity Profile
       </mpeg7:Name>
      </cde:Format>
      <cde:CodecParameter xsi:type="CodecParameterBitRateType">
       <BitRate>96000</BitRate>
     </cde:CodecParameter>
     </cde:Decoding>
    </TerminalCapability>
    <TerminalCapability xsi:type="cde:DisplaysType">
     <Display>
      <DisplayCapability xsi:type="cde:DisplayCapabilityType"
                         colorCapable="true">
       <Mode refreshRate="15">
        <Resolution horizontal="352" vertical="288"/>
       </Mode>
      </DisplayCapability>
     </Display>
    </TerminalCapability>
   </Terminal>

  </UsageEnvironmentProperty>

  <!-- Networks descriptions -->
  <UsageEnvironmentProperty xsi:type="cde:NetworksType">

   <Network id="modem" xsi:type="cde:NetworkType">
    <NetworkCharacteristic xsi:type="cde:NetworkCapabilityType"
                           maxCapacity="56000"/>
   </Network>

   <Network id="umts_3g" xsi:type="cde:NetworkType">
    <NetworkCharacteristic xsi:type="cde:NetworkCapabilityType"
                           maxCapacity="2000000"/>
   </Network>

   <Network id="wifi" xsi:type="cde:NetworkType">
    <NetworkCharacteristic xsi:type="cde:NetworkCapabilityType"
                           maxCapacity="11540000"/>
  </Network>

  <Network id="adsl" xsi:type="cde:NetworkType">
   <NetworkCharacteristic xsi:type="cde:NetworkCapabilityType"
                          maxCapacity="104857600"
                          minGuaranteed="96000"/>
  </Network>

  <Network id="ethernet" xsi:type="cde:NetworkType">
   <NetworkCharacteristic xsi:type="cde:NetworkCapabilityType"
                          maxCapacity="1024000000"
                          minGuaranteed="8000000"/>
  </Network>

 </UsageEnvironmentProperty>

 <!-- Users descriptions -->
 <UsageEnvironmentProperty xsi:type="cde:UsersType">

  <User id="roi_faces_preference" xsi:type="UserType">
   <UserCharacteristic xsi:type="FocusOfAttentionType">
    <ROI uri="#faces"/>
   </UserCharacteristic>
  </User>

  <User id="fast_adaptation" xsi:type="UserType">
   <UserCharacteristic xsi:type="UsagePreferencesType">
    <UsagePreferences>
     <mpeg7:FilteringAndSearchPreferences
        xsi:type="cde:FilteringAndSearchPreferencesType">
      <cde:AdaptationPreferences>
       <cde:ContentDegradation preferenceValue="30"/>
       <cde:Online preferenceValue="80"/>
       <cde:ExecutionCost preferenceValue="50"/>
      </cde:AdaptationPreferences>
     </mpeg7:FilteringAndSearchPreferences>
    </UsagePreferences>
   </UserCharacteristic>
  </User>

  <User id="cain21" xsi:type="UserType">
   <UserCharacteristic xsi:type="UsagePreferencesType">
    <UsagePreferences>
     <mpeg7:FilteringAndSearchPreferences
        xsi:type="cde:FilteringAndSearchPreferencesType">
      <cde:SourcePreferences>
       <cde:MediaFormat preferenceValue="50">
        <mpeg7:Content href="urn:mpeg:mpeg7:cs:ContentCS:2001:2">
         <mpeg7:Name>Audiovisual</mpeg7:Name>
        </mpeg7:Content>
        <cde:FileFormat href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:3"
                        preferenceValue="50">
         <cde:Name>MPEG file format</cde:Name>
        </cde:FileFormat>
        <cde:FileFormat href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:7"
                        preferenceValue="0">
         <cde:Name>Audio video interleave format</cde:Name>
        </cde:FileFormat>
        <cde:FileFormat href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:5"
                        preferenceValue="-50">
         <cde:Name>MPEG-4 file format</cde:Name>
        </cde:FileFormat>
        <cde:VisualCoding preferenceValue="30">
         <cde:Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:2.2.2"
```

```
                      preferenceValue="50">
           <cde:Name>MPEG-2 Video Main Profile @ Main Level</cde:Name>
          </cde:Format>
          <cde:Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.1"
                      preferenceValue="0">
           <cde:Name>MPEG-4 Visual Simple Profile</cde:Name>
          </cde:Format>
          <cde:Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.2"
                      preferenceValue="-10">
           <cde:Name>MPEG-4 Visual Advanced Simple Profile</cde:Name>
          </cde:Format>
         </cde:VisualCoding>
         <cde:AudioCoding preferenceValue="30">
          <cde:Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.3"
                      preferenceValue="50">
           <cde:Name>MPEG-2 Audio AAC</cde:Name>
          </cde:Format>
          <cde:Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:6"
                      preferenceValue="0">
           <cde:Name>AMR</cde:Name>
          </cde:Format>
         </cde:AudioCoding>
        </cde:MediaFormat>
       </cde:SourcePreferences>
       <cde:AdaptationPreferences>
        <cde:ContentDegradation preferenceValue="80"/>
        <cde:Online preferenceValue="50"/>
        <cde:ExecutionCost preferenceValue="30"/>
       </cde:AdaptationPreferences>
      </mpeg7:FilteringAndSearchPreferences>
     </UsagePreferences>
    </UserCharacteristic>
    <UserCharacteristic xsi:type="cde:ConversionPreferenceType">
     <GeneralResourceConversions>
      <Conversion order="1">
       <From href="urn:mpeg:mpeg7:cs:ContentCS:2001:4.2">
        <mpeg7:Name>Video</mpeg7:Name>
       </From>
       <To href="urn:mpeg:mpeg7:cs:ContentCS:2001:4.2">
        <mpeg7:Name>Video</mpeg7:Name>
       </To>
      </Conversion>
      <Conversion order="2">
       <From href="urn:mpeg:mpeg7:cs:ContentCS:2001:4.2">
        <mpeg7:Name>Video</mpeg7:Name>
       </From>
       <To href="urn:mpeg:mpeg7:cs:ContentCS:2001:4.1">
        <mpeg7:Name>Image</mpeg7:Name>
       </To>
      </Conversion>
      <Conversion order="3">
       <From href="urn:mpeg:mpeg7:cs:ContentCS:2001:4.2">
        <mpeg7:Name>Video</mpeg7:Name> </From>
       <To href="urn:mpeg:mpeg7:cs:ContentCS:2001:1">
        <mpeg7:Name>Audio</mpeg7:Name>
       </To>
      </Conversion>
     </GeneralResourceConversions>
    </UserCharacteristic>
   </User>

  </UsageEnvironmentProperty>

 </Description>
```

```
</DIA>
```

Listing 22: Usage Environment of CAIN-21

ARC description tool

Listing 23 shows the XML Schema of the ARC description tool. This description tool has been explained in Section 3 of Chapter 4. This subsection also discusses a usage example for this description tool.

```
<?xml version="1.0" ?>
<xsd:schema targetNamespace="urn:mpeg:mpeg21:2003:01-DIA-DIAC-NS"
            elementFormDefault="qualified"
            attributeFormDefault="unqualified"
            xmlns="urn:mpeg:mpeg21:2003:01-DIA-DIAC-NS"
            xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            id="carc.xsd">

 <!-- The following import elements are just hints about the types
        and elements used by this schema -->
<xsd:import namespace="urn:mpeg:mpeg21:2003:01-DIA-NS"
             schemaLocation="DIA.xsd"/>

<xsd:complexType name="ARCElementType">
 <xsd:complexContent>
  <xsd:extension base="dia:DIABaseType">
   <xsd:sequence>
    <xsd:element name="Id" type="xsd:string"
                 minOccurs="1" maxOccurs="1"/>
    <xsd:element name="OperationMode" type="xsd:string"
                 minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
  </xsd:extension>
 </xsd:complexContent>
</xsd:complexType>

 <xsd:complexType name="ARCType">
  <xsd:complexContent>
   <xsd:extension base="dia:DIADescriptionType">
    <xsd:sequence>
     <xsd:element name="Terminal" type="ARCElementType"
                  minOccurs="1" maxOccurs="1"/>
     <xsd:element name="Network" type="ARCElementType"
                  minOccurs="0" maxOccurs="1"/>
     <xsd:element name="User" type="ARCElementType"
                  minOccurs="0" maxOccurs="1"/>
     <xsd:element name="NaturalEnvironment"
                  type="ARCElementType" minOccurs="0"
                  maxOccurs="1"/>
   </xsd:sequence>
  </xsd:extension>
 </xsd:complexContent>
</xsd:complexType>

</xsd:schema>
```

Listing 23: ARC XML Schema of CAIN-21

Adapter Capabilities DI

Listing 24 shows the XML Schema that the *Adapter Capabilities DIs* use. The target namespace of these elements is *urn:vpu:cain21-adapter-capabilities*, which is shortened to *cac*. Each instance of the *AdapterCapabilitiesType* description tool describes the adaptation capabilities for a specific *Adapter*. The instances of the *ConversionCapabilitiesType* description tool describe the multiple conversions that the *Adapter* may implement. Certain capabilities such as the *Adapter-ClassName* or the *Platform* are provided at the *AdapterCapabilitiesType* level. Other capabilities such as the *ContentDegradation* or *ExecutionCost* are provided for each instance of the *Conver-sionCapabilityType* description tool.

```
<?xml version="1.0" ?>
<xsd:schema targetNamespace="urn:vpu:cain21-adapter-capabilities"
            elementFormDefault="qualified"
            attributeFormDefault="unqualified"
            xmlns="urn:vpu:cain21-adapter-capabilities"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS"
            id="cac.xsd">

 <!-- The following import elements are just hints about the types
      and elements used by this schema -->
 <xsd:import namespace="urn:mpeg:mpeg21:2003:01-DIA-NS"
             schemaLocation="ConvCapab-AMD1.xsd"/>

 <!-- Adapter Capabilities -->

 <xsd:complexType name="AdapterCapabilitiesType">
  <xsd:complexContent>
   <xsd:restriction base="dia:DIADescriptionType">
    <xsd:sequence>
     <xsd:element name="AdapterClassName" type="xsd:string" minOccurs="0" maxOccurs="1"/>
     <xsd:element name="Description" type="xsd:string" minOccurs="0" maxOccurs="1"/>
     <xsd:element name="Platform" type="ValuesType" minOccurs="1" maxOccurs="1"/>
     <xsd:element name="ConversionCapability" type="ConversionCapabilityType"
                  minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
   </xsd:restriction>
  </xsd:complexContent>
 </xsd:complexType>

 <xsd:complexType name="ConversionCapabilityType">
  <xsd:complexContent>
   <xsd:restriction base="dia:ConversionCapabilityType">
    <xsd:sequence>
     <xsd:element name="Description" type="xsd:string" minOccurs="0" maxOccurs="1"/>
     <xsd:element name="ContentDegradation" type="xsd:float"
                  minOccurs="1" maxOccurs="1"/>
     <xsd:element name="ExecutionCost" type="xsd:float" minOccurs="1" maxOccurs="1"/>
     <xsd:element name="Preconditions" type="PropertiesType"
                  minOccurs="1" maxOccurs="unbounded"/>
     <xsd:element name="Postconditions" type="PropertiesType"
                  minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:ID" use="required"/>
   </xsd:restriction>
  </xsd:complexContent>
 </xsd:complexType>

 <xsd:complexType name="PropertiesType">
  <xsd:sequence>
   <xsd:element name="URL" type="ValuesType" minOccurs="1" maxOccurs="1"/>
   <xsd:element name="Binding" type="BindingsType" minOccurs="1" maxOccurs="1"/>
   <xsd:element name="Content" type="ValuesType" minOccurs="1" maxOccurs="1"/>
   <xsd:element name="FileFormat" type="ValuesType" minOccurs="0" maxOccurs="1"/>
   <xsd:element name="MIMEType" type="ValuesType" minOccurs="0" maxOccurs="1"/>
   <xsd:element name="Bitrate" type="ValuesType" minOccurs="0" maxOccurs="1"/>
   <xsd:element name="VisualCoding" type="VisualCodingType" minOccurs="0" maxOccurs="1"/>
   <xsd:element name="AudioCoding" type="AudioCodingType" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="VisualCodingType">
  <xsd:sequence>
   <xsd:element name="Format" type="ValuesType" minOccurs="0" maxOccurs="1"/>
   <xsd:element name="ColorDomain" type="ValuesType" minOccurs="0" maxOccurs="1"/>
   <xsd:element name="Frame" type="FrameType" minOccurs="0" maxOccurs="1"/>
   <xsd:element name="Bitrate" type="ValuesType" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="FrameType">
  <xsd:sequence>
   <xsd:element name="Rate" type="ValuesType" minOccurs="0" maxOccurs="1"/>
   <xsd:element name="Width" type="ValuesType" minOccurs="0" maxOccurs="1"/>
   <xsd:element name="Height" type="ValuesType" minOccurs="0" maxOccurs="1"/>
   <xsd:element name="AspectRatio" type="ValuesType" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="AudioCodingType">
  <xsd:sequence>
   <xsd:element name="Format" type="ValuesType" minOccurs="1" maxOccurs="1"/>
   <xsd:element name="Bitrate" type="ValuesType" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="ValuesType">
  <xsd:choice>
   <xsd:element name="AnyValue" type="AnyValueType" minOccurs="0" maxOccurs="1"/>
   <xsd:element name="ValueSet" type="ValueSetType" minOccurs="0" maxOccurs="1"/>
   <xsd:element name="RangeValueSet" type="RangeValueSetType" minOccurs="0" maxOc-
curs="1"/>
  </xsd:choice>
 </xsd:complexType>

 <xsd:simpleType name="AnyValueType">
  <xsd:restriction base="xsd:string">
  </xsd:restriction>
 </xsd:simpleType>

 <xsd:complexType name="ValueSetType">
  <xsd:sequence>
   <xsd:element name="Value" type="ValueType" minOccurs="1" maxOccurs="unbounded"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="ValueType" mixed="true">
  <xsd:attribute name="href" type="xsd:string" use="required"/>
 </xsd:complexType>

 <xsd:complexType name="RangeValueSetType">
  <xsd:attribute name="from" type="xsd:string" use="required"/>
  <xsd:attribute name="to" type="xsd:string" use="required"/>
 </xsd:complexType>

 <xsd:complexType name="BindingsType">
  <xsd:complexContent>
   <xsd:restriction base="ValuesType">
    <xsd:sequence>
     <xsd:element name="ValueSet" type="BindingSetType" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
   </xsd:restriction>
  </xsd:complexContent>
 </xsd:complexType>

 <xsd:complexType name="BindingSetType">
  <xsd:complexContent>
   <xsd:restriction base="ValueSetType">
    <xsd:sequence>
     <xsd:element name="Value" type="BindingType" minOccurs="1" maxOccurs="4"/>
    </xsd:sequence>
   </xsd:restriction>
  </xsd:complexContent>
 </xsd:complexType>

 <xsd:complexType name="BindingType" mixed="true">
  <xsd:attribute name="href" type="BindingOptionType" use="required"/>
 </xsd:complexType>

 <xsd:simpleType name="BindingOptionType">
  <xsd:restriction base = "xsd:string">
   <xsd:enumeration value = "urn:mpeg:mpeg21:2007:01-BBL-NS:handler:INPROCESS"/>
   <xsd:enumeration value = "urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE"/>
   <xsd:enumeration value = "urn:mpeg:mpeg21:2007:01-BBL-NS:handler:TCP"/>
   <xsd:enumeration value = "urn:mpeg:mpeg21:2007:01-BBL-NS:handler:HTTP"/>
   <xsd:enumeration value = "urn:mpeg:mpeg21:2007:01-BBL-NS:handler:RTSP"/>
  </xsd:restriction>
 </xsd:simpleType>

</xsd:schema>
```

Listing 24: *Adapter Capabilities DI* XML Schema of CAIN-21

Listing 25 shows the adaptation capabilities of the *OnDemandVideoTranscoderAdapter*. This adapter includes two conversion modules: *ondemand_mpeg_transcoder* (for MPEG, and AVI file formats) and *ondemand_mp4_transcoder* (for 3GPP and MP4 file formats). This division was performed because the sets of input and output formats that each conversion module accepts and produces varied.

```
<?xml version="1.0" ?>

<!-- Video Transcoder Adapter Capabilities -->

<dia:DIA xmlns="urn:vpu:cain21-adapter-capabilities"
         xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS"
         xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <dia:Description xsi:type="AdapterCapabilitiesType"
                  id="ondemand_video_transcoder_adapter">
  <AdapterClassName>es.vpu.cain21.adapters.OnDemandVideoTranscoderAdapter
  </AdapterClassName>
  <Description>
   Transcodes video formats and sizes
  </Description>
  <Platform>
   <ValueSet>
    <Value href="Windows XP">Windows XP</Value>
    <Value href="Windows 2003">Windows 2003</Value>
    <Value href="Linux">Linux</Value>
    <Value href="Mac OS X">Mac OS X</Value>
   </ValueSet>
  </Platform>
  <!-- On Demand conversion using the ffmpeg command -->
  <ConversionCapability xsi:type="ConversionCapabilityType"
```

```
                    id="ondemand_mpeg_transcoder">
  <Description>
   This conversion performs transcoding using the ffmpeg command
  </Description>
  <ContentDegradation>0</ContentDegradation>
  <ExecutionCost>1.0</ExecutionCost>
  <Preconditions>
   <URL>
    <AnyValue/>
   </URL>
   <Binding>
    <ValueSet>
     <Value href="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:HTTP">HTTP</Value>
     <Value href="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE">FILE</Value>
    </ValueSet>
   </Binding>
   <Content>
    <ValueSet>
     <Value href="urn:mpeg:mpeg7:cs:ContentCS:2001:2">Audiovisual</Value>
     <Value href="urn:mpeg:mpeg7:cs:ContentCS:2001:4.2">Video</Value>
    </ValueSet>
   </Content>
   <FileFormat>
    <ValueSet>
     <Value href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:3">MPEG file format</Value>
     <Value href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:5">MPEG-4 file format</Value>
     <Value href="urn:vpu:cs:FileFormatCS:2009:3gp">3GPP file format</Value>
     <Value href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:7">
      Audio video interleave format
     </Value>
     <Value href="urn:vpu:cs:FileFormatCS:2009:swf">SWF format</Value>
    </ValueSet>
   </FileFormat>
   <MIMEType>
    <ValueSet>
     <Value href="video/mpeg">MPEG video</Value>
     <Value href="video/mpg">MPEG video</Value>
     <Value href="video/mp4">MP4 video</Value>
     <Value href="video/3gp">3GP video</Value>
     <Value href="video/avi">AVI video</Value>
     <Value href="video/swf">Flash video</Value>
    </ValueSet>
   </MIMEType>
   <VisualCoding>
    <Format>
     <ValueSet>
      <Value href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:1">
       MPEG-1 Video Coding Format
      </Value>
      <Value href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:2">
       MPEG-2 Video
      </Value>
      <Value href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:2.1">
       MPEG-2 Video Simple Profile
      </Value>
      <Value href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:2.2">
       MPEG-2 Video Main Profile
      </Value>
      <Value href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:2.2.2">
       MPEG-2 Video Main Profile @ Main Level
      </Value>
      <Value href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.1">
       MPEG-4 Visual Simple Profile
      </Value>
```

```
        <Value href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.1.2">
         H.264 Baseline Profile
        </Value>
        <Value href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.2">
         MPEG-4 Visual Advanced Simple Profile
        </Value>
        <Value href="urn:vpu:cs:VisualCodingFormatCS:2007:1">
         H.264 Baseline Profile @ Level 1.1
        </Value>
        <Value href="urn:vpu:cs:VisualCodingFormatCS:2007:2">
         H.264 Video Main Profile @ Level 3
        </Value>
        <Value href="urn:vpu:cs:VisualCodingFormatCS:2009:flv">
         Macromedia Flash video stream
        </Value>
        <Value href="urn:vpu:cs:VisualCodingFormatCS:2009:yuv420p">
         YUP420P
        </Value>
       </ValueSet>
       </Format>
     </VisualCoding>
     <AudioCoding>
      <Format>
       <ValueSet>
        <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:3">
         MPEG-1 Audio
        </Value>
        <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:3.1">
         MPEG-1 Audio Layer I
        </Value>
        <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:3.2">
         MPEG-1 Audio Layer II
        </Value>
        <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:3.3">
         MPEG-1 Audio Layer III
        </Value>
        <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4">
         MPEG-2 Audio
        </Value>
        <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.1">
         MPEG-2 Audio Low Sampling Rate
        </Value>
        <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.1.1">
         MPEG-2 Audio Low Sampling Rate Layer I
        </Value>
        <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.1.2">
         MPEG-2 Audio Low Sampling Rate Layer II
        </Value>
        <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.1.3">
         MPEG-2 Audio Low Sampling Rate Layer III
        </Value>
        <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.2">
         MPEG-2 Backward Compatible Multi-Channel
        </Value>
        <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.3">
         MPEG-2 Audio AAC
        </Value>
        <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.3.1">
         MPEG-2 Audio AAC Low Complexity Profile
        </Value>
        <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.4">
         MP3
        </Value>
        <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:6">
        AMR
       </Value>
      </ValueSet>
     </Format>
    </AudioCoding>
   </Preconditions>

   <Postconditions>
    <URL>
     <AnyValue/>
    </URL>
    <Binding>
     <ValueSet>
      <Value href="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE">FILE</Value>
     </ValueSet>
    </Binding>
    <Content>
     <ValueSet>
      <Value href="urn:mpeg:mpeg7:cs:ContentCS:2001:2">Audioviual</Value>
      <Value href="urn:mpeg:mpeg7:cs:ContentCS:2001:4.2">Video</Value>
     </ValueSet>
    </Content>
    <FileFormat>
     <ValueSet>
      <Value href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:3">MPEG file format</Value>
      <Value href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:7">
       Audio video interleave format
      </Value>
     </ValueSet>
    </FileFormat>
    <MIMEType>
     <ValueSet>
      <Value href="video/mpeg">MPEG video</Value>
      <Value href="video/mpg">MPEG video</Value>
      <Value href="video/avi">AVI video</Value>
     </ValueSet>
    </MIMEType>
    <Bitrate>
     <RangeValueSet from="20000" to="10000000"/>
    </Bitrate>
    <VisualCoding>
     <Format>
      <ValueSet>
       <Value href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:1">
        MPEG-1 Video Coding Format
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:2">
        MPEG-2 Video
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:2.1">
        MPEG-2 Video Simple Profile
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:2.2">
        MPEG-2 Video Main Profile
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:2.2.2">
        MPEG-2 Video Main Profile @ Main Level
       </Value>
      </ValueSet>
     </Format>
     <ColorDomain>
      <ValueSet>
       <Value href="color"/>
      </ValueSet>
     </ColorDomain>
     <Frame>
      <Rate>
       <RangeValueSet from="1" to="30"/>
      </Rate>
      <Width>
        <AnyValue/>
       </Width>
      <Height>
        <AnyValue/>
      </Height>
      <AspectRatio>
       <AnyValue/>
      </AspectRatio>
     </Frame>
     <Bitrate>
      <RangeValueSet from="5000" to="1000000"/>
     </Bitrate>
    </VisualCoding>
    <AudioCoding>
     <Format>
      <ValueSet>
       <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:3.2">
        MPEG-1 Audio Layer II
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4">
        MPEG-2 Audio
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.4">
        MP3
       </Value>
      </ValueSet>
     </Format>
     <Bitrate>
      <RangeValueSet from="2000" to="800000"/>
     </Bitrate>
    </AudioCoding>
   </Postconditions>
  </ConversionCapability>

<!-- On demand MP4 conversion using the ffmpeg command -->
  <ConversionCapability xsi:type="ConversionCapabilityType" id="ondemand_mp4_transcoder">
   <Description>
    This conversion performs transcoding using the ffmpeg command
   </Description>
   <ContentDegradation>0</ContentDegradation>
   <ExecutionCost>1.0</ExecutionCost>
   <Preconditions>
    <URL>
     <AnyValue/>
    </URL>
    <Binding>
     <ValueSet>
      <Value href="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:HTTP">HTTP</Value>
      <Value href="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE">FILE</Value>
     </ValueSet>
    </Binding>
    <Content>
     <ValueSet>
      <Value href="urn:mpeg:mpeg7:cs:ContentCS:2001:2">Audiovisual</Value>
      <Value href="urn:mpeg:mpeg7:cs:ContentCS:2001:4.2">Video</Value>
     </ValueSet>
    </Content>
    <FileFormat>
```

```
     <ValueSet>
      <Value href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:3">MPEG file format</Value>
      <Value href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:5">MPEG-4 file format</Value>
      <Value href="urn:vpu:cs:FileFormatCS:2009:3gp">3GPP file format</Value>
      <Value href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:7">
       Audio video interleave format
      </Value>
      <Value href="urn:vpu:cs:FileFormatCS:2009:swf">SWF format</Value>
     </ValueSet>
    </FileFormat>
    <MIMEType>
     <ValueSet>
      <Value href="video/mpeg">MPEG video</Value>
      <Value href="video/mpg">MPEG video</Value>
      <Value href="video/mp4">MP4 video</Value>
      <Value href="video/3gp">3GP video</Value>
      <Value href="video/avi">AVI video</Value>
      <Value href="video/swf">Flash video</Value>
     </ValueSet>
    </MIMEType>
    <VisualCoding>
     <Format>
      <ValueSet>
       <Value href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:1">
        MPEG-1 Video Coding Format
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:2">
        MPEG-2 Video
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:2.1">
        MPEG-2 Video Simple Profile
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:2.2">
        MPEG-2 Video Main Profile
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:2.2.2">
        MPEG-2 Video Main Profile @ Main Level
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.1">
        MPEG-4 Visual Simple Profile
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.1.2">
        H.264 Baseline Profile
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.2">
        MPEG-4 Visual Advanced Simple Profile
       </Value>
       <Value href="urn:vpu:cs:VisualCodingFormatCS:2007:1">
        H.264 Baseline Profile @ Level 1.1
       </Value>
       <Value href="urn:vpu:cs:VisualCodingFormatCS:2007:2">
        H.264 Video Main Profile @ Level 3
       </Value>
       <Value href="urn:vpu:cs:VisualCodingFormatCS:2009:flv">
        Macromedia Flash video stream
       </Value>
       <Value href="urn:vpu:cs:VisualCodingFormatCS:2009:yuv420p">
        YUP420P
       </Value>
      </ValueSet>
      </Format>
    </VisualCoding>
    <AudioCoding>
     <Format>
```

```
      <ValueSet>
       <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:3">
        MPEG-1 Audio
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:3.1">
        MPEG-1 Audio Layer I
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:3.2">
        MPEG-1 Audio Layer II
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:3.3">
        MPEG-1 Audio Layer III
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4">
        MPEG-2 Audio
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.1">
        MPEG-2 Audio Low Sampling Rate
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.1.1">
        MPEG-2 Audio Low Sampling Rate Layer I
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.1.2">
        MPEG-2 Audio Low Sampling Rate Layer II
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.1.3">
        MPEG-2 Audio Low Sampling Rate Layer III
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.2">
        MPEG-2 Backward Compatible Multi-Channel
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.3">
        MPEG-2 Audio AAC
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.3.1">
        MPEG-2 Audio AAC Low Complexity Profile
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.4">
        MP3
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:6">
        AMR
       </Value>
      </ValueSet>
     </Format>
    </AudioCoding>
   </Preconditions>

   <Postconditions>
    <URL>
     <AnyValue/>
    </URL>
    <Binding>
     <ValueSet>
      <Value href="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE">FILE</Value>
     </ValueSet>
    </Binding>
    <Content>
     <ValueSet>
      <Value href="urn:mpeg:mpeg7:cs:ContentCS:2001:2">Audioviual</Value>
      <Value href="urn:mpeg:mpeg7:cs:ContentCS:2001:4.2">Video</Value>
     </ValueSet>
    </Content>
    <FileFormat>
```

```
     <ValueSet>
      <Value href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:5">MPEG-4 file format</Value>
      <Value href="urn:vpu:cs:FileFormatCS:2009:3gp">3GPP file format</Value>
      <Value href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:7">
       Audio video interleave format
      </Value>
      <Value href="urn:vpu:cs:FileFormatCS:2009:swf">SWF format</Value>
     </ValueSet>
    </FileFormat>
    <MIMEType>
     <ValueSet>
      <Value href="video/mp4">MP4 video</Value>
      <Value href="video/3gp">3GP video</Value>
      <Value href="video/avi">AVI video</Value>
      <Value href="video/swf">Flash video</Value>
     </ValueSet>
    </MIMEType>
    <Bitrate>
     <RangeValueSet from="20000" to="10000000"/>
    </Bitrate>
    <VisualCoding>
     <Format>
      <ValueSet>
       <Value href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.1">
        MPEG-4 Visual Simple Profile
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.1.2">
        >MPEG-4 Visual Simple Profile @ Level 1
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.2">
        MPEG-4 Visual Advanced Simple Profile
       </Value>
       <Value href="urn:vpu:cs:VisualCodingFormatCS:2007:1">
        H.264 Baseline Profile @ Level 1.1
       </Value>
       <Value href="urn:vpu:cs:VisualCodingFormatCS:2007:2">
        H.264 Video Main Profile @ Level 3
       </Value>
       <Value href="urn:vpu:cs:VisualCodingFormatCS:2009:flv">
        Macromedia Flash video stream
       </Value>
      </ValueSet>
     </Format>
     <ColorDomain>
      <ValueSet>
       <Value href="color"/>
      </ValueSet>
     </ColorDomain>
     <Frame>
      <Rate>
       <RangeValueSet from="1" to="30"/>
      </Rate>
      <Width>
        <AnyValue/>
       </Width>
      <Height>
        <AnyValue/>
      </Height>
      <AspectRatio>
       <AnyValue/>
      </AspectRatio>
     </Frame>
     <Bitrate>
      <RangeValueSet from="5000" to="1000000"/>
     </Bitrate>
    </VisualCoding>
    <AudioCoding>
     <Format>
      <ValueSet>
       <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:6">
        AMR
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4">
        MPEG-2 Audio
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.3">
        MPEG-2 Audio AAC
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.3.1">
        MPEG-2 Audio AAC Low Complexity Profile
       </Value>
       <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.4">
        MP3
       </Value>
      </ValueSet>
     </Format>
     <Bitrate>
      <RangeValueSet from="2000" to="800000"/>
     </Bitrate>
    </AudioCoding>
   </Postconditions>
  </ConversionCapability>
 </dia:Description>
</dia:DIA>
```

Listing 25: *Adapter Capabilities DI* example for on-demand video transcoding

Listing 26 shows another *Adapter Capabilities DI* that *Demonstration 6* of Chapter 7 uses. This adapter efficiently combines raw audio and visual stream into a compressed MPEG-1 video container. Note that the binding mode of the input is *TCP* and the binding mode of the output is *MEMORY*. This means that subsequent *Adapters* can efficiently retrieve the output that this *Adapter* produces (i.e., in main memory).

```
<?xml version="1.0" ?>

<!-- Raw Video Combiner Adapter Capabilities -->

<dia:DIA xmlns="urn:vpu:cain21-adapter-capabilities"
         xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS"
         xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <dia:Description xsi:type="AdapterCapabilitiesType" id="video_transcoder_adapter">
  <AdapterClassName>es.vpu.cain21.adapters.RawVideoCombinerAdapter</AdapterClassName>
  <Description>
   Read raw video and audio and produces MPEG-1 video
  </Description>
  <Platform>
   <ValueSet>
    <Value href="Windows XP">Windows XP</Value>
    <Value href="Windows 2003">Windows 2003</Value>

   </ValueSet>
  </Platform>
  <!-- Online combiner from raw video and audio to MPEG-1 video -->
  <ConversionCapability xsi:type="ConversionCapabilityType" id="raw_video_combiner">
   <Description>
    Online combiner from raw video and audio to MPEG-1 video
   </Description>
   <ContentDegradation>0</ContentDegradation>
   <ExecutionCost>1.0</ExecutionCost>
   <Preconditions>
    <URL>
     <AnyValue/>
    </URL>
    <Binding>
     <ValueSet>
      <Value href="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:TCP">TCP</Value>
      <Value href="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:FILE">FILE</Value>
     </ValueSet>
    </Binding>
    <Content>
     <ValueSet>
      <Value href="urn:mpeg:mpeg7:cs:ContentCS:2001:2">Audiovisual</Value>
     </ValueSet>
    </Content>
    <FileFormat>
     <ValueSet>
      <Value href="uam:vpu:cs:FileformatCS:2008:summarized-video">
       VPU summarized video
      </Value>
     </ValueSet>
    </FileFormat>
    <MIMEType>
     <ValueSet>
      <Value href="video/x-vpu-summarized-video">Raw summarized video</Value>
     </ValueSet>
    </MIMEType>
    <Bitrate>
     <RangeValueSet from="40000" to="2000000"/>
    </Bitrate>
    <VisualCoding>
     <Format>
      <ValueSet>
       <Value href="urn:vpu:cs:VisualCodingFormatCS:2008:raw">
        Raw visual video
       </Value>
      </ValueSet>
      </Format>
      <Frame>
       <Rate>
        <RangeValueSet from="1" to="30"/>
       </Rate>
       <Width>
        <AnyValue/>
       </Width>
       <Height>
        <AnyValue/>
       </Height>
       <AspectRatio>
        <ValueSet>
         <Value href="0.75"/>
        </ValueSet>
        </AspectRatio>
     </Frame>
     <Bitrate>
      <RangeValueSet from="20000" to="1000000"/>
     </Bitrate>
    </VisualCoding>
    <AudioCoding>
     <Format>
      <ValueSet>
       <Value href="urn:vpu:cs:AudioCodingFormatCS:2008:wav">
```

```
        WAV audio
       </Value>
      </ValueSet>
     </Format>
     <Bitrate>
      <RangeValueSet from="4000" to="400000"/>
     </Bitrate>
    </AudioCoding>
   </Preconditions>

   <Postconditions>
    <URL>
     <AnyValue/>
    </URL>
    <Binding>
     <ValueSet>
      <Value href="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:INPROCESS">MEMORY</Value>
     </ValueSet>
    </Binding>
    <Content>
     <ValueSet>
      <Value href="urn:mpeg:mpeg7:cs:ContentCS:2001:2">Audioviual</Value>
      <Value href="urn:mpeg:mpeg7:cs:ContentCS:2001:4.2">Video</Value>
     </ValueSet>
    </Content>
    <FileFormat>
     <ValueSet>
      <Value href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:3">MPEG file format</Value>
      <Value href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:7">
       Audio video interleave format
      </Value>
     </ValueSet>
    </FileFormat>
    <MIMEType>
     <ValueSet>
      <Value href="video/mpeg">MPEG video</Value>
      <Value href="video/mpg">MPEG video</Value>
      <Value href="video/avi">AVI video</Value>
     </ValueSet>
    </MIMEType>
    <Bitrate>
     <RangeValueSet from="20000" to="10000000"/>
    </Bitrate>
    <VisualCoding>
     <Format>
      <ValueSet>
       <Value href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:1">
        MPEG-1 Video Coding Format
       </Value>
      </ValueSet>
     </Format>
     <Frame>
      <Rate>
       <RangeValueSet from="1" to="30"/>
      </Rate>
      <Width>
        <AnyValue/>
       </Width>
      <Height>
        <AnyValue/>
      </Height>
      <AspectRatio>
       <ValueSet>
        <Value href="0.75"/>
       </ValueSet>
      </AspectRatio>
     </Frame>
     <Bitrate>
      <RangeValueSet from="5000" to="1000000"/>
     </Bitrate>
    </VisualCoding>
    <AudioCoding>
     <Format>
      <ValueSet>
       <Value href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4">
        MPEG-2 Audio
       </Value>
      </ValueSet>
     </Format>
     <Bitrate>
      <RangeValueSet from="2000" to="800000"/>
     </Bitrate>
    </AudioCoding>
   </Postconditions>
  </ConversionCapability>

 </dia:Description>
</dia:DIA>
```

Listing 26: *Adapter Capabilities DI* example for the raw video combiner adapter

Properties DI

Listing 27 shows the XML Schema of the *Properties DI* that CAIN-21 utilizes. The target namespace of these elements is *urn:vpu:cain21-properties.di*, which is shortened to *cpd*. The *PropertiesDIType* description tool is defined as a derivation by restriction of the standard *mpeg21:DIADescriptionType* description tool. This type includes four important elements that correspond to the five groups of properties: *DIProperties*, *ComponentProperties*, *AdapterProperties*, *ConversionProperties* and *UsageEnvProperties*.

```
<?xml version="1.0" ?>
<xsd:schema targetNamespace="urn:vpu:cain21-properties-di"
            elementFormDefault="qualified"
            attributeFormDefault="unqualified"
            xmlns="urn:vpu:cain21-properties-di"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS"
            id="cpr.xsd">

  <!-- The following import elements are just hints about the types
       and elements used by this schema -->
 <xsd:import namespace="urn:mpeg:mpeg21:2003:01-DIA-NS"
             schemaLocation="DIA.xsd"/>

 <!-- Properties DI general structure -->

 <xsd:complexType name="PropertiesDIType">
  <xsd:complexContent>
   <xsd:restriction base="dia:DIADescriptionType">
    <xsd:sequence>
     <xsd:element name="DIProperties" type="ComponentPropertiesType"
                  minOccurs="1" maxOccurs="1"/>
     <xsd:element name="ComponentProperties" type="ComponentPropertiesType"
                  minOccurs="1" maxOccurs="1"/>
     <xsd:element name="AdapterProperties" type="AdapterPropertiesType"
                  minOccurs="1" maxOccurs="1"/>
     <xsd:element name="ConversionProperties" type="ConversionPropertiesType"
```

```
                  minOccurs="1" maxOccurs="1"/>
    <xsd:element name="UsageEnvProperties" type="UsageEnvPropertiesType"
                  minOccurs="1" maxOccurs="1"/>
   </xsd:sequence>
  </xsd:restriction>
 </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="DIPropertiesType">
 <xsd:choice  minOccurs="0" maxOccurs="unbounded">
  <xsd:element name="Property" type="PropertyType"/>
  <xsd:element name="ComposedProperty" type="ComposedPropertyType"/>
 </xsd:choice>
</xsd:complexType>

<xsd:complexType name="ComponentPropertiesType">
 <xsd:choice  minOccurs="0" maxOccurs="unbounded">
  <xsd:element name="Property" type="PropertyType"/>
  <xsd:element name="ComposedProperty" type="ComposedPropertyType"/>
 </xsd:choice>
</xsd:complexType>

<xsd:complexType name="ConversionPropertiesType">
 <xsd:choice  minOccurs="0" maxOccurs="unbounded">
  <xsd:element name="Property" type="PropertyType"/>
  <xsd:element name="ComposedProperty" type="ComposedPropertyType"/>
 </xsd:choice>
</xsd:complexType>

<xsd:complexType name="AdapterPropertiesType">
 <xsd:sequence>
  <xsd:element name="Property" type="PropertyType" minOccurs="0" maxOccurs="unbounded"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="UsageEnvPropertiesType">
 <xsd:sequence>
  <xsd:element name="TerminalProperties" minOccurs="1" maxOccurs="1">
   <xsd:complexType>
    <xsd:choice  minOccurs="0" maxOccurs="unbounded">
     <xsd:element name="Property" type="PropertyType"/>
     <xsd:element name="ComposedProperty" type="ComposedPropertyType"/>
    </xsd:choice>
   </xsd:complexType>
  </xsd:element>
  <xsd:element name="NetworkProperties" minOccurs="1" maxOccurs="1">
   <xsd:complexType>
    <xsd:choice  minOccurs="0" maxOccurs="unbounded">
     <xsd:element name="Property" type="PropertyType"/>
     <xsd:element name="ComposedProperty" type="ComposedPropertyType"/>
    </xsd:choice>
   </xsd:complexType>
  </xsd:element>
  <xsd:element name="UserProperties" minOccurs="1" maxOccurs="1">
   <xsd:complexType>
    <xsd:choice  minOccurs="0" maxOccurs="unbounded">
     <xsd:element name="Property" type="PropertyType"/>
     <xsd:element name="ComposedProperty" type="ComposedPropertyType"/>
    </xsd:choice>
   </xsd:complexType>
  </xsd:element>
 </xsd:sequence>
</xsd:complexType>
```

```
 <xsd:complexType name="PropertyType">
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="required" type="xsd:boolean" use="required"/>
  <xsd:attribute name="xpath" type="xsd:string" use="required"/>
 </xsd:complexType>

 <xsd:complexType name="ComposedPropertyType">
  <xsd:sequence>
   <xsd:element name="Value" minOccurs="1" maxOccurs="unbounded">
    <xsd:complexType>
     <xsd:attribute name="xpath" type="xsd:string" use="required"/>
    </xsd:complexType>
   </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="required" type="xsd:boolean" use="required"/>
 </xsd:complexType>

</xsd:schema>
```

Listing 27: *Properties DI* XML Schema of CAIN-21

Listing 28 shows the current Properties DI of CAIN-21. Note that each property has a *name*, an attribute that indicates whether the property is *required* (must exist) and an *xpath* expression.

```
<?xml version="1.0" ?>

<!-- Properties DI -->

<dia:DIA xmlns="urn:vpu:cain21-properties-di"
         xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS"
         xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <dia:Description xsi:type="PropertiesDIType">
  <DIProperties>
    <Property name="genre" required="false"
             xpath="/Item/Descriptor/Statement/Mpeg7/DescriptionUnit/Genre/@href"/>
  </DIProperties>
  <ComponentProperties>
   <Property name="id" required="true"
             xpath="/@id"/>
   <!-- Component descriptions -->
   <Property name="url" required="true"
             xpath="/Resource/@ref"/>
   <Property name="content" required="true"
             xpath="//Mpeg7/Description/MediaInformation/MediaProfile
                    /MediaFormat/Content/@href"/>
   <Property name="format" required="false"
             xpath="//Mpeg7/Description/MediaInformation/MediaProfile
                    /MediaFormat/FileFormat/@href"/>
   <Property name="mime_type" required="false"
             xpath="/Resource/@mimeType"/>
   <Property name="file_size" required="false"
             xpath="//Mpeg7/Description/MediaInformation
                    /MediaProfile/MediaFormat/FileSize"/>
   <Property name="bitrate" required="false"
             xpath="//Mpeg7/Description/MediaInformation
                    /MediaProfile/MediaFormat/BitRate"/>
   <Property name="visual_format" required="false"
             xpath="//Mpeg7/Description/MediaInformation
                    /MediaProfile//MediaFormat/VisualCoding/Format/@href"/>
   <Property name="visual_color_domain" required="false"
             xpath="//Mpeg7/Description/MediaInformation/MediaProfile
                    //MediaFormat/VisualCoding/Format/@colorDomain"/>
   <Property name="visual_bitrate" required="false"
            xpath="//Mpeg7/Description/MediaInformation/MediaProfile
                   /ComponentMediaProfile/MediaFormat[Content
                   /@href='urn:mpeg:mpeg7:cs:ContentCS:2001:4']/BitRate"/>
   <Property name="visual_frame_rate" required="false"
            xpath="//Mpeg7/Description/MediaInformation/MediaProfile
                   /ComponentMediaProfile/MediaFormat/VisualCoding/Frame/@rate"/>
   <Property name="visual_frame_aspect_ratio" required="false"
            xpath="//Mpeg7/Description/MediaInformation/MediaProfile
                   //MediaFormat/VisualCoding/Pixel/@aspectRatio"/>
   <ComposedProperty name="visual_frame" required="false">
    <Value xpath="//Mpeg7/Description/MediaInformation/MediaProfile
                 //MediaFormat/VisualCoding/Frame/@width"/>
    <Value xpath="//Mpeg7/Description/MediaInformation/MediaProfile
                 //MediaFormat/VisualCoding/Frame/@height"/>
  </ComposedProperty>
   <Property name="audio_format" required="false"
            xpath="//Mpeg7/Description/MediaInformation/MediaProfile
                   /ComponentMediaProfile/MediaFormat/AudioCoding/Format/@href"/>
   <Property name="audio_bitrate" required="false"
            xpath="//Mpeg7/Description/MediaInformation/MediaProfile
                   /ComponentMediaProfile/MediaFormat[Content
                  /@href='urn:mpeg:mpeg7:cs:ContentCS:2001:1']/BitRate"/>
   <!-- Creation descriptions -->
   <Property name="original_variation" required="false"
            xpath="//Mpeg7/Description/VariationSet/Source//MediaUri"/>
   <ComposedProperty name="variation" required ="false">
    <Value xpath="//Mpeg7/Description/VariationSet/Variation//VariationRelationship"/>
    <Value xpath="//Mpeg7/Description/VariationSet/Variation//MediaUri"/>
  </ComposedProperty>
   <Property name="spoken_lang" required="false"
            xpath="//Mpeg7/Description/CreationInformation/Classification/Language"/>
   <!-- ROI properties -->
   <Property name="roi_id" required="false"
            xpath="//Mpeg7/DescriptionUnit[@type='StillRegionType']/@id"/>
   <Property name="roi_box" required="false"
            xpath="//Mpeg7/DescriptionUnit[@type='StillRegionType']
                   /SpatialMask/SubRegion/Box"/>
   <!-- AdaptationQoS -->
   <ComposedProperty name="aqos_constraint" required ="false">
    <Value xpath="//Description//Module/Constraint/@iOPinRef"/>
    <Value xpath="//Description//Module/Constraint/Values/Vector"/>
   </ComposedProperty>
   <ComposedProperty name="aqos_adaptation_operator" required ="false">
    <Value xpath="//Description//Module/AdaptationOperator/@iOPinRef"/>
    <Value xpath="//Description//Module/AdaptationOperator/Values/Vector"/>
   </ComposedProperty>
   <ComposedProperty name="aqos_utility" required ="false">
    <Value xpath="//Description//Module/Utility/@iOPinRef"/>
    <Value xpath="//Description//Module/Utility/Values/Vector"/>
   </ComposedProperty>
  </ComponentProperties>
  <AdapterProperties>
   <Property name="id" required="true"
            xpath="/@id"/>
   <Property name="adapter_class_name" required="true"
            xpath="/AdapterClassName"/>
   <Property name="description" required="false"
            xpath="/Description"/>
   <Property name="platform" required="true"
                  xpath="/Platform"/>
  </AdapterProperties>
  <ConversionProperties>
   <!-- General properties -->
   <Property name="id" required="true"
             xpath="/@id"/>
   <Property name="conversion_description" required="false"
             xpath="/Description"/>
   <Property name="content_degradation" required="true"
             xpath="/ContentDegradation"/>
   <Property name="execution_cost" required="true"
             xpath="/ExecutionCost"/>
   <!-- Input properties -->
   <Property name="pre_url" required="true"
             xpath="/Preconditions/URL"/>
   <Property name="pre_binding" required="true"
             xpath="/Preconditions/Binding"/>
   <Property name="pre_content" required="true"
             xpath="/Preconditions/Content"/>
   <Property name="pre_mime_type" required="false"
             xpath="/Preconditions/MIMEType"/>
   <Property name="pre_format" required="false"
             xpath="/Preconditions/FileFormat"/>
   <Property name="pre_bitrate" required="false"
             xpath="/Preconditions/Bitrate"/>
   <Property name="pre_visual_format" required="false"
             xpath="/Preconditions/VisualCoding/Format"/>
   <Property name="pre_visual_color_domain" required="false"
             xpath="/Preconditions/VisualCoding/ColorDomain"/>
   <Property name="pre_visual_frame_rate" required="false"
             xpath="/Preconditions/VisualCoding/Frame/Rate"/>
   <ComposedProperty name="pre_visual_frame" required ="false">
    <Value xpath="/Preconditions/VisualCoding/Frame/Width"/>
    <Value xpath="/Preconditions/VisualCoding/Frame/Height"/>
  </ComposedProperty>
   <Property name="pre_visual_frame_aspect_ratio" required="false"
             xpath="/Preconditions/VisualCoding/Frame/AspectRatio"/>
   <Property name="pre_visual_bitrate" required="false"
             xpath="/Preconditions/VisualCoding/Bitrate"/>
   <Property name="pre_audio_format" required="false"
             xpath="/Preconditions/AudioCoding/Format"/>
   <Property name="pre_audio_bitrate" required="false"
             xpath="/Preconditions/AudioCoding/Bitrate"/>
   <!-- Output properties -->
   <Property name="post_url" required="true"
             xpath="/Postconditions/URL"/>
   <Property name="post_binding" required="true"
             xpath="/Postconditions/Binding"/>
   <Property name="post_content" required="true"
             xpath="/Postconditions/Content"/>
   <Property name="post_mime_type" required="false"
             xpath="/Postconditions/MIMEType"/>
   <Property name="post_format" required="true"
             xpath="/Postconditions/FileFormat"/>
   <Property name="post_bitrate" required="false"
             xpath="/Postconditions/Bitrate"/>
   <Property name="post_visual_format" required="false"
             xpath="/Postconditions/VisualCoding/Format"/>
   <Property name="post_visual_color_domain" required="false"
             xpath="/Postconditions/VisualCoding/ColorDomain"/>
   <Property name="post_visual_frame_rate" required="false"
             xpath="/Postconditions/VisualCoding/Frame/Rate"/>
   <ComposedProperty name="post_visual_frame" required ="false">
    <Value xpath="/Postconditions/VisualCoding/Frame/Width"/>
    <Value xpath="/Postconditions/VisualCoding/Frame/Height"/>
  </ComposedProperty>
   <Property name="post_visual_frame_aspect_ratio" required="false"
             xpath="/Postconditions/VisualCoding/Frame/AspectRatio"/>
   <Property name="post_visual_bitrate" required="false"
             xpath="/Postconditions/VisualCoding/Bitrate"/>
   <Property name="post_audio_format" required="false"
             xpath="/Postconditions/AudioCoding/Format"/>
   <Property name="post_audio_bitrate" required="false"
             xpath="/Postconditions/AudioCoding/Bitrate"/>
  </ConversionProperties>

  <UsageEnvProperties>
   <TerminalProperties>
    <Property name="id" required="true"
              xpath="/@id"/>
    <Property name="binding" required="true"
              xpath="/TerminalCapability[@type='cde:HandlerCapabilitiesType']
                     /Handler/@handlerURI"/>
    <Property name="transport_decoding_format" required="false"
              xpath="/TerminalCapability[@type='cde:CodecCapabilitiesType']
                     /Decoding[@type='cde:TransportCapabilitiesType']/Format/@href"/>
    <Property name="image_decoding_format" required="false"
              xpath="/TerminalCapability[@type='cde:CodecCapabilitiesType']
                     /Decoding[@type='cde:ImageCapabilitiesType']/Format/@href"/>
    <Property name="video_decoding_format" required="false"
              xpath="/TerminalCapability[@type='cde:CodecCapabilitiesType']
                     /Decoding[@type='cde:VideoCapabilitiesType']/Format/@href"/>
    <Property name="audio_decoding_format" required="false"
              xpath="/TerminalCapability[@type='cde:CodecCapabilitiesType']
                     /Decoding[@type='cde:AudioCapabilitiesType']/Format/@href"/>
    <Property name="display_color_capable" required="false"
              xpath="/TerminalCapability[@type='cde:DisplaysType']/Display
                   /DisplayCapability[@type='cde:DisplayCapabilityType']/@colorCapable"/>
    <Property name="display_refresh_rate" required="false"
              xpath="/TerminalCapability[@type='cde:DisplaysType']/Display
               /DisplayCapability[@type='cde:DisplayCapabilityType']/Mode/@refreshRate"/>
    <ComposedProperty name="display_resolution" required ="false">
     <Value xpath="/TerminalCapability[@type='cde:DisplaysType']
                   /Display/DisplayCapability[@type='cde:DisplayCapabilityType']
                   /Mode/Resolution/@horizontal"/>
     <Value xpath="/TerminalCapability[@type='cde:DisplaysType']
                   /Display/DisplayCapability[@type='cde:DisplayCapabilityType']
                   /Mode/Resolution/@vertical"/>
    </ComposedProperty>
    <ComposedProperty name="display_active_resolution" required ="false">
     <Value xpath="/TerminalCapability[@type='cde:DisplaysType']
                   /Display/DisplayCapability/Mode
                   /Resolution[@activeResolution='true']/@horizontal"/>
     <Value xpath="/TerminalCapability[@type='cde:DisplaysType']
                   /Display/DisplayCapability/Mode
                   /Resolution[@activeResolution='true']/@vertical"/>
    </ComposedProperty>
    <Property name="video_bitrate" required="false"
              xpath="/TerminalCapability/Decoding[@type='cde:VideoCapabilitiesType']
                     /CodecParameter/BitRate"/>
    <Property name="audio_bitrate" required="false"
              xpath="/TerminalCapability/Decoding[@type='cde:AudioCapabilitiesType']
                     /CodecParameter/BitRate"/>
   </TerminalProperties>
   <NetworkProperties>
    <Property name="id" required="true"
              xpath="/@id"/>
    <Property name="max_capacity" required="false"
              xpath="/NetworkCharacteristic/@maxCapacity"/>
    <Property name="min_guaranteed" required="false"
              xpath="/NetworkCharacteristic/@minGuaranteed"/>
   </NetworkProperties>
```

```
   <UserProperties>
    <Property name="id" required="true"
              xpath="/@id"/>
    <!-- File format preferences -->
    <Property name="pref_file_format_utility" required="false"
              xpath="/UserCharacteristic/UsagePreferences/FilteringAndSearchPreferences
                     /SourcePreferences/MediaFormat/@preferenceValue"/>
    <Property name="pref_file_format_value_href" required="false"
              xpath="/UserCharacteristic/UsagePreferences/FilteringAndSearchPreferences
                     /SourcePreferences/MediaFormat/FileFormat/@href"/>
    <Property name="pref_file_format_value_utility" required="false"
              xpath="/UserCharacteristic/UsagePreferences/FilteringAndSearchPreferences
                     /SourcePreferences/MediaFormat/FileFormat/@preferenceValue"/>
    <!-- Visual format preferences -->
    <Property name="pref_visual_coding_utility" required="false"
              xpath="/UserCharacteristic/UsagePreferences/FilteringAndSearchPreferences
                     /SourcePreferences/MediaFormat/VisualCoding/@preferenceValue"/>
    <Property name="pref_visual_coding_value_href" required="false"
              xpath="/UserCharacteristic/UsagePreferences/FilteringAndSearchPreferences
                     /SourcePreferences/MediaFormat/VisualCoding/Format/@href"/>
    <Property name="pref_visual_coding_value_utility" required="false"
              xpath="/UserCharacteristic/UsagePreferences/FilteringAndSearchPreferences
                   /SourcePreferences/MediaFormat/VisualCoding/Format/@preferenceValue"/>
    <!-- Audio format preferences -->
    <Property name="pref_audio_coding_utility" required="false"
              xpath="/UserCharacteristic/UsagePreferences/FilteringAndSearchPreferences
                     /SourcePreferences/MediaFormat/AudioCoding/@preferenceValue"/>
    <Property name="pref_audio_coding_value_href" required="false"
              xpath="/UserCharacteristic/UsagePreferences/FilteringAndSearchPreferences
                     /SourcePreferences/MediaFormat/AudioCoding/Format/@href"/>
    <Property name="pref_audio_coding_value_utility" required="false"
              xpath="/UserCharacteristic/UsagePreferences/FilteringAndSearchPreferences
                     /SourcePreferences/MediaFormat/AudioCoding/Format/@preferenceValue"/>
    <!-- Content degradation preferences -->
    <Property name="pref_content_degradation_utility" required="false"
              xpath="/UserCharacteristic/UsagePreferences/FilteringAndSearchPreferences
                     /AdaptationPreferences/ContentDegradation/@preferenceValue"/>
    <!-- Online preferences -->
    <Property name="pref_online_utility" required="false"
              xpath="/UserCharacteristic/UsagePreferences/FilteringAndSearchPreferences
                     /AdaptationPreferences/Online/@preferenceValue"/>
    <!-- Minimization of conversions preferences -->
    <Property name="pref_execution_cost_utility" required="false"
              xpath="/UserCharacteristic/UsagePreferences/FilteringAndSearchPreferences
                     /AdaptationPreferences/ExecutionCost/@preferenceValue"/>
    <!-- FocusOfAttention preferences -->
    <Property name="pref_focus_of_attention" required="false"
              xpath="/UserCharacteristic/ROI/@uri"/>
   </UserProperties>
  </UsageEnvProperties>

 </dia:Description>
</dia:DIA>
```

Listing 28: *Properties DI* of CAIN-21

Appendix B: The implicit ontology of CAIN-21

This appendix compares the description capabilities of CAIN-21 with those of the semantic web. As explained in Section 4.1 of Chapter 2, different description languages have different level of expressiveness (i.e., capability to describe knowledge). Description languages at the top of the Semantic Web Stack [43] allow representing high-level concepts and relationships as well as automatically inferring knowledge from existing knowledge using AI reasoning techniques. In exchange, as we come closer to the top of the stack the complexity of the underlying algorithms also increases.

XML-based standards provide an elemental syntax for the content structure within documents. The major limitation of simple XML-based standards is they associate no semantics with the meaning of the content. As a result, identical XML description elements may mean very different things depending on the context in which they are used. MPEG-21 relies on the XML Schema to define the structure of the content and therefore have this limitation. For example, if the semantic were not given, two instances of the *Format* element inside the *VideoCapabilitiesType* and *AudioCapabilitiesType* description tools (which refer to different media streams) would be confused.

In this research, we make a distinction between *explicit ontologies* (aka formal ontologies) and *implicit ontologies* (aka informal ontologies). Explicit ontologies, in addition to describing the content structure of the elements, use a formal description language (such as RDF or OWL) to describe the ontology elements meaning and relationships. Implicit ontologies lacks of a formal description of the meanings and relationships of the ontology elements, but describe these elements in the text of the standard (usually, with natural language).

MPEG-21 is one of such standards that lacks of an explicit ontology but defines an implicit ontology. The implicit ontology of MPEG-21 have been criticised because it is informally described and therefore is not a good choice for tasks such as automatically inferring knowledge [116]. Support for reasoning (e.g., subsumption) without an explicit ontology is very complicate. However, the multimedia research community has frequently accepted and used the MPEG-21 implicit ontology. Research has demonstrated that a wide variety of multimedia algorithms can be effectively developed within the MPEG-21 framework. In the area of multimedia systems, the MPEG-21 description tools enable the representation a large set of concepts and relationships. To perform automatic knowledge inference and remove ambiguities, some parts of the standard have been extended with description languages that provide explicit ontologies with a higher level of expressiveness. Particularly, the explicit ontology is represented using semantic description languages such as OWL.

Authors have extended MPEG-21 with OWL to address tasks such as automatic multimedia adaptation [16][54] or intellectual property management [117]. However, the increase in expressiveness of using an explicit ontology came at the cost of increasing the complexity. We hypothesise that in the case of multimedia adaptation decision, the results obtained in [16][54] can be attained easily with an implicit ontology such as MPEG-21. To study this hypothesis, this chapter investigates an alternative approach that relies on MPEG-21 to create a seamless description of the multimedia adaptation problem. Subsequently, Chapter 5 develops a method that demonstrates how to perform automatic adaptation decision-making in the MPEG-21 domain. This adaptation is simpler and more effective than the one proposes in [16][54] and without the ambiguities that an implicit ontology may produce.

Even though, in contrast with OWL based approaches, XML Schema based approaches demand simpler algorithms, the introduction of MPEG-21 in a multimedia system is not an easy task. One of the reasons is that the high variety of multimedia description tools complicates their use. The identification of the description tool for each concept is not effortless. However, the main difficulty in using MPEG-21 is not it all-inclusive nature but its incompleteness: there are multimedia concepts in the real world that do not correspond to any MPEG-21 description tool. In fact, it is still not clear whether it is possible to represent every real world concept by means of any description language [43]. At the time of writing this thesis, MPEG-21 has been the more complete endeavour to describe and standardise multimedia elements.

Removing ambiguities with properties

Subsection 4.1 of Chapter 2 introduced the Semantic Web. Semantic Web languages such as OWL allow explicitly representing and storing concepts and their relationships in a semantic graph. Frequently, automatic reasoning techniques use this graph to search for relationships among the values and to infer additional information.

In CAIN-21, the concepts are represented by means of properties and the relationships are limited. Specifically, relationships are just intended to assist the matching algorithm developed in Chapter 5. In this sense, CAIN-21 uses a delimited subset of the rich relationships that the Semantic Web provides.

The *Properties DI* complies with the KISS principle and avoids the ambiguities that the use of implicit ontologies produces. The *Properties DI* has two main purposes:

- To elude changes in the decision algorithm when the metadata under consideration evolves. In contrast, depending on the reasoning techniques, changes in the relationships of the semantic graph of an explicit ontology may imply changes in the underlying reasoning algorithms.
- To let the use of an implicit ontology (such as MPEG-21), but at the same time, avoiding ambiguities. The ambiguities are removed because the elements are references with an XPath expression. If the same XML element appears in different parts of the document, its XPath expression will be different. By providing different labels to semantic-different properties, we elude ambiguities.

That being said, it is important to highlight that the use of properties to remove ambiguities work best in systems where the meaning of all the properties is centralized in a *Properties DI*. Therefore, the use of implicit ontologies (in contrast with explicit ontologies) would become more difficult in distributed systems. In these decentralized systems, advertisers and requesters have very different perspectives about the meaning of the description elements and the use of an explicit ontology may become necessary.

In a nutshell, the matching mechanism described in Chapter 5 tests whether the input of one *Adapter* accepts the output of the previous *Adapter* in the sequence. The matching mechanism also tests whether the terminal accepts the output of the last *Adapter*. In order to perform these tests, usually the simple properties-based representation mechanism has demonstrated to be enough. Section 2 of Chapter 7 demonstrates its suitability and also demonstrates that the matching mechanism operates efficiently. However, during the tests, we encountered that occasionally it is convenient to consider more complicate relationships between properties. For instance, to maintain the ratio in the adapted media, the width and height should be considered together. In these cases, we use *composed properties*. Listing 8 shows an example of these composed properties. The *visual_frame* property uses the *ComposedProperty* element to gather the width and height elemental values. In the representation schema of CAIN-21, these elemental values can be represented by means of ranges or as a placeholder accepting any value.

Appendix C: Acronyms

Acronym	Description
3GPP	3rd Generation Partnership Project
4CIF	4 times Common Intermediate Format
AAC	Advanced Audio Coding
aceMedia	Integrating Knowledge, Semantics and Content for User-centred Intelligent Media Services
AI	Artifitial Intelligence
AMR	Adaptive Multi-Rate
AOD	Audio On Demand
API	Application Programming Interface
ARC	Adaptation Request Configuration
AVC	Advanced Video Coding
AVI	Audio Video Interleaved
BBL	Bitstream Binding Language
BIFS	Binary Format for Scenes
BSD	Bitstream Syntax Description
CAT	Content Adaptation Tool
CAIN	Content Adaptation INtegrator
CAIN-21	Content Adaptation INtegrator in the MPEG-21 framework
CC/PP	Composite Capabilities / Preference Profiles
CIF	Common Intermediate Format
CM	Comunidad de Madrid
CSP	Contraint Satisfaction Problem
DAE	Description Adaptation Engine
DAG	Directed Acyclic Graph
dB	Decibel
DCAF	Dynamic Content Adaptation Framework
DF	Degree of Freedom
DI	Digital Item
DIA	Digital Item Adaptation
DIAC	Digital Item Adaptation Configuration
DIAE	Digial Item Adaptation Engine
DID	Digital Item Description
DIDL	Digital Item Description Language
DM	Decision Module
DOI	Digital Object Identifier
Ds	Descriptors
DSs	Description Schemes
DSP	Digital Signal Processing
DVD	Digital Versatile Disc
EBNF	Extended Backus-Naur Form
FPU	Formación de Personal Investigador
GIF	Graphic Interchange Format

GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
ID	IDentifier
IOPE	Inputs, Outputs, Preconditions Effects
IST-IP	Information Society Technologies - Integrated Project
IST-FP6	Information Society Technologies - 6th Framework Programme
JNI	Java Native Interface
JPEG	Joint Photographic Experts Group
JSVM	Joint Scalable Video Model
KISS	Keep It Short and Simple
koMMa	Knowledge-based Multimedia Adaptation
MAGG	Multimedia Adaptation Graph Service
MESH	Multimedia Semantic Syndication for Enhanced News Services
MMC	Multimedia Communication research group
MOD	Media On Demand
MP3	MPEG-1 Audio Layer 3
MP4	MPEG-4 File Format
MPEG	Moving Picture Expert Group
OWL	Web Ontology Language
OWL-S	OWL-Services
PDDL	Planning Domain Definition Language
PNG	Portable Network Graphics
PSNR	Peak Signal-to-Noise Ratio
QCIF	Quarter Common Intermediate Format
RAE	Resource Adaptation Engine
RAM	Random-Access Memory
RDF	Resource Description Framework
ROIs	Regions of Interest
RTP	Real-time Transport Protocol
RTSP	Real-Time Streaming Protocol
SMIL	Synchronized Multimedia Integration Language
SPARQL	Simple Protocol and RDF Query Language
SE	Standard Error
SSIM	Structural SIMilarity
soc	Sequence of Conversions
SOIs	Segments of Interest
SSOC	Set of Sequences of Conversions
STRIPS	Stanford Research Institute Problem Solver
SVC	Scalable Video Coding
TCP	Transport Control Protocol
TIC	Tecnologías de la Información y la Comunicación
TIFF	Tagged Image File Format
TV	Tele Vision
UAM	Universidad Autónoma de Madrid
UAProf	User Agent Profile
UCD	Universal Contraints Description

UED	Usage Environment Description
UMA	Universal Multimedia Access
UME	Universal Multimedia Experience
URI	Universal Resource Identifier
URL	Universal Resource Locator
VOD	Video In Demand
VPU Lab	Video Processing and Understanding Lab
VQM	Video Quality Metrics
VRT	Vlaamse Radio- en Televisieomroep
WAP Forum	Wireless Access Protocol Forum
WAV	Waveform Audio Format
WMA	Windows Media Audio
WMF	Windows Media Format
XML	Extensible Markup Language
XSL	eXtensible Stylesheet Language
XSLT	XML Stylesheet Language for Transformations

Appendix D: Achievements and contributions

This appendix gathers the achievements and contributions that support the work in this thesis.

Publications

This subsection provides our list of publications related to the content of the thesis. After each publication, we indicate the chapter or chapters to which the publication of the contribution's content corresponds.

Articles in international journals evaluated and accepted for publication:

- F. López, J. M. Martínez, N. Garcia, "A Model for Preference-Driven Multimedia Adaptation Decision-Making In The MPEG-21 Framework". Journal of Multimedia Tools and Applications, Springer, DOI: 10.1007/s11042-010-0507-1. Online Mar. 2010. [related to Chapter 6]
- F. López, D. Jannach, J. M. Martínez, C. Timmerer, N. García, H. Hellwagner, "Bounded Non-Deterministic Planning for Multimedia Adaptation", Applied Intelligence, Springer, DOI: 10.1007/s10489-010-0242-3. Online Jul. 2010. [related to Chapter 5]

Articles in international conferences evaluated and accepted for publication:

- F. López, J. M. Martínez, V. Valdés, "Multimedia Content Adaptation within the CAIN Framework via Constraints Satisfaction and Optimization". Springer Lecture Notes in Computer Science vol. 4398, pp. 149-163, Jul. 2006. [related to Chapter 5]
- J. Molina, J. M. Martínez, V. Valdés, F. López, "Extensibility of Adaptation Capabilities in the CAIN Content Adaptation Engine", Proc. of the 1st International Conference on Semantic and Digital Media Technologies, SAMT 2006, Dec. 2006. [related to Chapter 3]
- F. López, J. M. Martinez, "Multimedia Content Adaptation Modelled as a Constraints Matching Problem with Optimisation", Proc. of the International Workshop on Image Analysis for Multimedia Interactive Services, WIAMIS'2007, pp. 82-85 (ISBN 0-7695-2818-X), Jun. 2007. [related to Chapter 5]
- F. López, D. Jannach, J. M. Martínez, C. Timmerer, H. Hellwagner, N. García, "Multimedia Adaptation Decisions Modelled as Non-Deterministic Operations", Proc. of the 9th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS'08), pp. 46-49, Klagenfurt, Austria, May 2008. [related to Chapter 5]
- F. López, J. M. Martínez and N. García, "Towards a Fully MPEG-21 Compliant Adaptation Engine: Complementary Description Tools and Architectural Models", Springer Lecture Notes in Computer Science under preparation, Proc. of 6th International Workshop on Adaptive Multimedia Retrieval (AMR 2008), Berlin, German, Jun. 2008. [related to Chapter 4]
- F. López, J. M. Martínez, N. García. "Automatic Adaptation Decision-Making Using an Image to Video Adaptation Tool in The MPEG-21 Framework". Proc. of the 10th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS'09), pp. 222-225, May 2009. [related to Chapter 6]
- A. García, J. Molina, F. López, V. Valdés, F. Tiburzi, J. M. Martínez, J. Bescós. "Instant Customized Summaries Streaming: A Service for Immediate Awareness of New Video Content".

Lectures Notes in Computer Science, under preparation, Proc. of 7th International Workshop on Adaptive Multimedia Retrieval (AMR 2009), Sep. 2009. [related to Chapter 6]
- F. López, J. M. Martínez, N. García. "CAIN-21: An Extensible and Metadata-Driven Multimedia Adaptation Engine in the MPEG-21 Framework", Lectures Notes in Computer Science vol. 5887, pp. 114-125, Dec. 2009. [related to Chapter 3]
- F. López, G. Nur, S. Dogan, H.K. Arachchi, M. Mrak, J. M. Martínez, N. García, A. Kondoz. "Improving Scalable Video Adaptation in a Knowledge-Based Framework". Proc. of the 11th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS 2010), under publication. Apr. 2010. [related to Chapter 6]
- F. López, J. M. Martínez, N. García. "Automatic Adaptation Decision Making in the MPEG-21 Framework: Mechanisms and Complementary Description Tools". Proc. of the 11th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS 2010), under publication. Apr. 2010. [related to Chapter 4]

Book chapters:

- Cantador, F. López, J. Bescós, P. Castells, and J. M. Martínez. "Chapter 7: Enhanced Media Descriptions for the Automatic Adaptation of Audiovisual Content Retrieval". Book: Personalization of Interactive Multimedia Services: A Research and Development Perspective - CB, ISBN: 978-1-60456-680-2. Editors: J. J. Pazos-Arias, C. Delgado Kloos and N. Martin Lopez. Publisher: Nova Publishers. 2008. [related to Chapter 3, Chapter 4]

Research projects

- IST-IP-001765: Integrating Knowledge, Semantics and Content for User-centred Intelligent Media Services (aceMedia) (Mar 2005 -Dec 2007)
- IST-FP6-027685: Multimedia Semantic Syndication for Enhanced News Services (MESH) (Mar 2006- Mar 2009)
- CAIN-21: Content Adaptation INtegrator in the MPEG-21 framework (Open source) (2005-2010)

Appendix E: Conclusiones

Los métodos de decisión de la adaptación multimedia actuales se centran en decidir cómo adaptar tipos concretos de contenido multimedia en vez de en decidir cómo llevar a cavo la adaptación multimedia independientemente de su contenido y contexto. Respecto a este tema conviene resaltar que MPEG-21 sí que ha realizado una descripción bastante genérica de un sistema multimedia. Además de no haber una visión genérica y sistemática, tampoco hay consenso respecto a los tipos y puntos temporales en los cuales se puede realizar la decisión de la adaptación de contenidos multimedia.

Esta tesis ha mostrado que es posible usar con efectividad metadatos para hacer decisiones sistemáticas y automáticas de adaptación multimedia. Los mecanismos de decisión automática liberan al usuario de esta responsabilidad. Para hacer estas decisiones hemos desarrollado un mecanismo de planificación que es independiente de la semántica del contenido multimedia. En concreto, el planificador sólo utiliza metadatos que describen el formato del contenido a adaptar y las restricciones del entorno de uso. De esa forma, el mismo método se puede aplicar para gestionar diferentes problemas de adaptación multimedia. Además, la semántica del contenido se puede tener en cuenta durante una etapa posterior. Este trabajo ha identificado los puntos de decisión en los cuales un sistema de computación puede tomar las decisiones. Además, para mejorar la experiencia de usuario, las decisiones automáticas utilizan las preferencias del usuario para personalizar el contenido a sus preferencias personales.

Otro objetivo importante de esta tesis es la interoperabilidad. Para tratar este tema, hemos definido un mecanismo de extensibilidad para los módulos de adaptación. Después hemos mostrado cómo al combinar los módulos de adaptación aumenta progresivamente el rango de posibles adaptaciones. También hemos mostrado que el motor de adaptación puede identificar automáticamente en qué orden y con qué parámetros se pueden ejecutar los módulos de adaptación. El motor de adaptación resultante es interoperativo, se puede extender progresivamente y es capaz de resolver una gran variedad de problemas de adaptación multimedia.

Para demostrar nuestra propuesta, hemos creado un conjunto de tests de adaptación almacenables y repetibles. Las herramientas de descripción que formalizan estos tests están extraídas en su mayoría del marco MPEG-21. De esta forma conseguimos un mayor nivel de interoperabilidad y de hecho, este marco es muy genérico y reutilizable. Sin embargo, en algunos casos hemos identificado ambigüedades o limitaciones, y en estos casos hemos tenido que modificar estas herramientas de descripción.

Al final de la etapa de contribuciones se describen tres puntos de decisión principales: selección, decisiones estáticas y decisiones dinámicas. Claramente, esta no es la única organización posible para los puntos de decisión; una organización completamente diferente de los puntos de decisión podría también lograr adaptaciones sistemáticas y automáticas. Sin embargo, al menos, este trabajo ha mostrado que es posible hacer adaptaciones sistemáticas y automáticas.

Asimismo, actualmente no hay consenso en las preferencias de usuario que ayudan a hacer decisiones de adaptación sistemáticas y automáticas. Para lograr una gestión más sistemática de las preferencias de usuario, hemos propuesto organizar estas preferencias en una jerarquía de restricciones usando relaciones de preferencias para representar las relaciones cualitativas y cuantitativas entre los resultados del conjunto completo de preferencias. A partir de esta representación podemos crear un grafo de preferencias que organice explícitamente las relaciones y permita la búsqueda sistemática y automática de resultados óptimos.

References

[1] F. Pereira, I. Burnett, “Universal multimedia experiences for tomorrow”, IEEE Signal Processing Magazine, Special Issue on Universal Multimedia Access, vol. 20(2), pp. 63-73, Mar. 2003.

[2] J. Lachner, A. Lorenz, B. Reiterer, A. Zimmermann, H. Hellwagner. "Challenges Toward User-Centric Multimedia". Proc. of Second International Workshop on Semantic Media Adaptation and Personalization (SMAP'07), pp. 159-164, Dec. 2007.

[3] M. Almaoui, A. Kushki, K. Plataniotis, "Metadata-driven multimedia transcoding for distance learning", Multimedia Systems, vol. 12(6) pp. 505-520, May 2007.

[4] I. S. Burnett, F. Pereira, R. Van de Walle, R. Koenen, "The MPEG-21 Book", Ed. John Wiley and Sons. 2006.

[5] International Press Telecommunication Council. "NewsML 2.0 Specification". Technical Report. 2007.

[6] ISO/IEC 15938, "Information Technology - Multimedia Content Description Interface (MPEG-5) - Part 5: Multimedia Description Schemes", 2003.

[7] A. A. Sofokleous, M. C. Angelides, “DCAF: An MPEG-21 Dynamic Content Adaptation Framework”. Multimedia Tools and Applications, vol. 40(2) pp. 151-182, 2008.

[8] S. Dogan, S. Eminsoy, A. H. Sadka, A. M. Kondoz, "Video Content Adaptation using Transcoding for Enabling UMA over UMTS", Proc of 5th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS 04). Apr. 2004.

[9] I. Ahmad, X. Wei, Y. Sun, Y. Qin, "Video Transcoding: An Overview of Various Techniques and Research Issues", IEEE Transactions on Multimedia, vol. 7(5):793-804, 2005.

[10] D. Van Deursen, W. Van Lancker, S. De Bruyne, W. De Neve, E. Mannens, R. Van de Walle, "Format-independent and metadata-driven media resource adaptation using semantic web technologies", Multimedia Systems vol. 16(2) pp. 1432-1882, Mar. 2010.

[11] C-S. Li, R. Mohan and J.R. Smith, "Multimedia Content Description in the InfoPyramid", Proc. of Special session on Signal Processing in Modern Multimedia Standards (ICASP’98), May 1998.

[12] M. Kimiaei Asadi. “Adaptation de Contenu Multimédia avec MPEG-21: Conversion de Ressources et Adaptation Sémantique de Scènes”. PhD. Thesis. Ecole Nationale Supérieure des Télécommunications. Jun. 2005.

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

[14] M. Shahidi, A. Attou, K. Moessner, H. Aghvami, "Content Adaptation: Requirements and Architecture", Proc. of Proc. of 10th International Conference on Information Integration and Web-based Applications and Services (iiWAS2008), pp. 626-629, Nov. 2008.

[15] C. Timmerer, “Generic Adaptation of Scalable Multimedia Resources”. Verlag Dr. Muller, 2008.

[16] D. Jannach, K. Leopold, Ch. Timmerer, H. Hellwagner. “A knowledge-based framework for multimedia adaptation”. International Journal on Applied Intelligence, Springer, vol. 24(2), pp. 109 - 125, Apr. 2006.

[17] G. Nur, H. Kodikara Arachchi, S. Dogan, A. M. Kondoz, "Evaluation of Quality Scalability Layer Selection for Bit Rate Adaptation of Scalable Video Content", Proc. of 10th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS 2009), pp. 218-221, May 2009.

[18] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, "Image quality assessment from error visibility to structural similarity", IEEE Transactions on Image Processing, vol. 13:4. 2004.

[19] F. López, G. Nur, S. Dogan, H.K. Arachchi, M. Mrak, J. M. Martínez, N. García, A. Kondoz. “Improving Scalable Video Adaptation in a Knowledge-Based Framework”. Proc. of the 11th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS 2010), under publication. Apr. 2010.

[20] Lee, J. Y. B.Sons, “Scalable continuous media streaming systems”, Ed. Wiley. 2007.

[21] L. Chiariglione. "MPEG: a technological basis for multimedia applications", IEEE Mltimedia, vol 2(1), pp. 85-89. 1995.

[22] Nadeem Iftikhar, Muhammad Abdul Qadir, and Omara Abdul Hamid, "Group Profile and Ontology-based Semantic Annotation of Multimedia Data for Efficient Retrieval," Proc. of the 2nd International Workshop on Context-Based Information Retrieval 2007 in conjunction with Sixth International and Interdisciplinary Conference on Modeling and Using Context, Aug. 2007.

[23] B.S. Manjunath, P. Salenbier, T. Sikora, "Introduction to MPEG-7 Multimedia Content Description Interface", Ed. John Wiley & Sons. 2003.

[24] WWW Consortium (W3C). "Extensible Markup Language (XML) 1.0 (2nd edition)". Oct. 2000.

[25] WWW Consortium (W3C). "XML Schema". W3C Recommendation, May 2001.

[26] H. Kosch, L. Boszormenyi, M. Doller, M. Libsie, P. Schojer, A. Kofler, "The Life Cycle of Multimedia Metadata". IEEE Multimedia, vol. 12(1) pp. 80-86, Jan. 2005.

[27] ISO/IEC 14977:1996, Information Technology - Syntactic Metalanguage - Extended BNF.

[28] L. Rong, I. Burnett, "Dynamic multimedia adaptation and updating of media streams with MPEG-21", Proc. of the First IEEE Conference on Consumer Communications and Networking (CCNC 2004), pp. 436-441, Jan. 2004.

[29] C. Tsinaraki, S. Christodoulakis, "A multimedia user preference model that supports semantics and its application to MPEG 7/21". Proc. of 12th International Multi-Media Model-ling Conference (MMM'06), Jan. 2006.

[30] B. Köhncke, W. Balke. “Preference-driven personalization for flexible digital item adaptation”, Multimedia Systems, vol. 13(2) pp. 119-130, Aug. 2007.

[31] L. Rong, I. Burnett, “Facilitating Universal Multimedia Adaptation (UMA) in a Heterogeneous Peer-to-Peer Network”. Second International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution, AXMEDIS 2006, pp. 105-109. Dec. 2006.

[32] WWW Consortium (W3C), “XML Path Language (XPath)”. November 1999.

[33] WWW Consortium (W3C). "XSL Transformation. Version 1.0". November 1999.

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

[35] J. Thomas-Kerr, I- Burnett, C. Ritz, "Format-Independent Multimedia Streaming", Proc. of IEEE International Conference on Multimedia and Expo, pp. 1509-151, Jul. 2006.

[36] ISO/IEC 21000-18:2008 FDAM 1“Information technology – Multimedia framework (MPEG-21) – Part 18: Digital Item Streaming”. 2008.

[37] Y. Wang, J. Kim, S. Chang, H. Kim, "Utility-based video adaptation for universal multimedia access (UMA) and content-based utility function prediction for real-time video transcoding", IEE transactions of multimedia, vol. 9(2), pp. 213-200, Feb. 2007.

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

[39] ISO/IEC 21000-6:2001. "Information technology - Multimedia framework (MPEG-21) - Part 6: Right Data Dictionary", 2001.

[40] Open Mobile Alliance. "User Agent Profile". Technical Report. 2006.

[41] W3C Consortium, "Resource Description Framework (RDF)". Feb. 2004.

[42] C. Timmerer, J. Jaborning, H. Hellwagner, "A Survey on Delivery Context Description Formats - A Comparison and Mapping Model", Journal of Digital Information Management - Special issue on context-aware and mobile multimedia databases and services, vol. 8(1), pp. 15-26, Feb. 2010.

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

[44] WWW Consortium (W3C). "Multimedia Vocabularies on the Semantic Web". http://www.w3.org/2005/Incubator/mmsem/

[45] S. Russell, Stuart J. "Artificial Intelligence: A modern approach". Prentice Hall. 2nd ed. 2003.

[46] V. Barbosa. M.T. Andrade, "Multicao: a semantic approach to context-aware adaptation decision taking", Proc. of the 10th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS'09), pp. 133-136, May 2009.

[47] S. De Bruyne, D. De Schrijver, W. De Neve, D. Van Deursen, R. Van de Walle. "Enhanced Shot-Based Video Adaptation using MPEG-21 generic Bitstream Syntax Schema", Proc. of IEEE Symposium on In Computational Intelligence in Image and Signal Processing (CIISP 2007), pp. 380-385, Apr. 2007.

[48] L. Herranz, J. M. Martínez, "An integrated approach to summarization and adaptation using H264/MPEG-4 SVC", Signal processing: Image Communication, vol. 24:4, pp. 499-509, Feb 2009.

[49] F. Barreiro, J. M. Martínez, V. Valdés, "Visual Tools for ROI Montage in an Image2Video Application", IEEE Transactions on Circuits and Systems for Video Technology, vol. 19(12) pp. 1927-1932, 2009.

[50] M. Zufferey, H. Kosch, "Semantic Adaptation of Multimedia Content", Proc. of 48th International Symposium ELMAR-2006 focused on Multimedia Signal Processing and Communications, pp. 319-322. Jun. 2006.

[51] M. Hori, G. Kondoh, H. Ono, S. Hirose, S. Singhal. "Annotation-based Web Content Transcoding". Computer Networks. vol. 33(1), pp. 197-211. Jun. 2000.

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

[53] B. Köhncke, W. Balke. "Preference-driven personalization for flexible digital item adaptation", Multimedia Systems, vol. 13(2), pp. 119-130, Aug. 2007.

[54] P. Soetens, M. De Geyter. "Applying domain knowledge to multistep media adaptation based on semantic web services", Proc. of Workshop on Image Analysis for Multimedia Interactive Systems (WIAMIS 05), 4 pp. (CD-ROM Proc). Apr. 2005.

[55] D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. Payne, M. Sabou, M. Solanki, N. Srinivasan, K. Sycara. "Bringing semantics to web services: the OWL-S approach". Proc. of the first international workshop on semantic web services and web process composition (SWSWPC 2004), pp. 6–9, 2004.

[56] G. Berhe, L. Brunie, J.M. Pierson. "Planning-Based Multimedia Adaptation Services Composition for Pervasive Computing". Proc. of 2nd International Conference on Signal-Image Technology Internet based Systems (SITIS'2006) pp. 132-143. Dec 2006.

[57] C. Timmerer, "Generic Adaptation of Scalable Multimedia Resources", Ed. Verlag. 2008.

[58] A. Hutter, P. Amon, G. Panis, E. Delfosse, M. Ransburg, H. Hellwagner, "Automatic adaptation of streaming multimedia content in a dynamic and distributed environment". Proc. of the IEEE International Conference on Image Processing (ICIP 2005), pp. 716-719. 2005.

[59] D. Van Deursen, W. De Neve, D. De Schrijver, R. Van de Walle. "gBFlavor: a new tool for fast and automatic generation of generic bitstream syntax descriptions", MTAP vol 40(3), pp. 453-494. Dec. 2008.

[60] J.M. Martínez, V. Valdés, J. Bescos, L. Herranz. "Introducing CAIN: A metadata-driven content adaptation manager integrating heterogeneous content adaptation tools". Proc. of WIAMIS 2005, pp. 5-10. 2005.

[61] J. Molina, J. M. Martínez, V. Valdés, F. López. "Extensibility of adaptation capabilities in the CAIN content adaptation engine". Poster and demo Proc. of SAMT 2006, pp. 29-30. 2006.

[62] M. Ghallab, D. S. Nau, P. Traverso, “Automated Planning: Theory and Practice”. Morgan Kaufmann. 2004.

[63] A. Blum, M. Furst. "Fast planning through Planning Graph analysis". Artificial Intelligence, vol. 90, pp. 1636-1642. 1995.

[64] Li Xu, Wen-Xiang Gu, and Xin-Mei Zhang. "Backward-Chaining Flexible Planning". Lecture Notes in Computer Science, vol. 3960, pp. 1611-3349. 2006.

[65] J. V. Neumann, O. Morgenstern. “Theory of Games and Economic Behavior”. Princeton Uni-versity Press; 3rd ed. 1980.

[66] B. L. Slantchev. “Game Theory: Preferences and expected utility”. Technical report, Political Science Courses, University of California - San Diego, 2007.

[67] J. Nocedal. “Numerical Optimization”. Springer; 2nd ed. 2006.

[68] L. Chen, P. Pu. “Survey of Preference Elicitation Methods”, Technical report, Human Computer Interaction Group. 2004.

[69] D. Braziunas, C. Boutilier. "Minimax regret-based elicitation of generalized additive utilities". Proc. of the Twenty-third Conference on Uncertainty in Artificial Intelligence (UAI-07), pp. 25-32, 2007.

[70] E. Dellis, B. Seeger, "Efficient computation of reverse skyline queries". Proc. of the 33rd international conference on Very large databases (VLDB'07), pp. 291-302, Sep. 2007.

[71] J. Kuo, C. Fang, J. Yeh, J. Wu, “An Importance Measurement For Video And Its Application To TV News Items Distillation”. IEEE International Conference on Image Processing ICIP '04, vol. 1, pp. 621- 624, Oct. 2004.

[72] J. T. Kerr, I. Burnett, C. H. Ritz, S. Devillers, D. De Schrijever, R. Van de Walle, "Is That a Fish in Your Ear? A Universal Metalanguage for Multimedia", IEEE Multimedia, 2007, vol. 14(2) pp. 72-77, Jun. 2007.

[73] F. López, J. M. Martínez, N. García, “CAIN-21: An extensible and metadata-driven multimedia adaptation engine in the MPEG-21 framework”. Lectures Notes in Computer Science vol. 5887, pp.114-125, Dec. 2009.

[74] WWW Consortium (W3C), “XPointer xpointer() Scheme”. Dec. 2002.

[75] B. Pellan, C. Concolato, "Metadata-driven Dynamic Scene Adaptation". Proc. of the International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS 2007), pp. 67-70. 2007.

[76] W. Y. Lum, F.C.M. Lau, "A context-aware decision engine for content adaptation". IEEE Pervasive Computing, vol. 1(3) pp. 41-49, 2002.

[77] C. Timmerer, V.H. Ortega, J.M. González, A. León, “Measuring quality of experience for MPEG-21-based cross-layer multimedia content adaptation”, Proc. WISe'08, Apr. 2008.

[78] F. López, J. M. Martínez, N. García. “Towards a fully MPEG-21 compliant adaptation engine: complementary description tools and architectural models”. Lectures Notes in Computer Science, Springer Verlag, vol. 5811, pp. 155-169, 6th International Workshop on Adaptive Multimedia Retrieval (AMR 2008), Berlin, German, June 26-28, 2008.

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

[80] S. De Zutter, R. Van de Walle, "Enhanced Quality of Experience in Heterogeneous Environments from a Content Author’s Perspective". Sixth FirW PhD Symposium, Ghent University, 30th Nov. 2005.

[81] M. Einhoff, J- Casademont, F. Perdrix, Stefan. Noll. “ELIN: A MPEG Related News Framework”. Proc. of the 47th International Symposium, ELMAR-2005, pp. 139-140. Jun. 2005.

[82] V. Valdés, J. M. Martinez, “Content Adaptation Capabilities Description Tool for Supporting Extensibility in the CAIN Framework”. Lecture Notes in Computer Science 4105, pp. 395-402. Sep. 2006.

[83] Apache Java Xalan XSLT Processor. Available online at http://xml.apache.org/xalan-j/.

[84] F. López, J. M. Martinez, "Multimedia Content Adaptation Modelled as a Constraints Matching Problem with Optimisation". Proc. of the 8th International Workshop on Image Analysis for Multimedia Interactive Services, WIAMIS'2007, pp. 82-85. Jun. 2007.

[85] F. López, D. Jannach, J. M. Martínez, C. Timmerer, N. García and H. Hellwagner, "Bounded non-deterministic planning for multimedia adaptation", Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies, Springer, DOI: 10.1007/s10489-010-0242-3. Online Jul. 2010.

[86] D. Chapman. "Planning for Conjunctive Goals". Artificial Intelligence, vol. 32, pp. 333-379. 1987.

[87] Thomas Plan, George Zorpas and Rjive Bagrodia. "An Extensible and Scalable Content Adaptation Pipeline Architecture to Support Heterogeneous Clients". Proc. of ICDCS, pp. 507-516. 2002.

[88] E. Pednault. "Synthesizing plans that contain actions with context-dependent effects". Computational Intelligence, vol. 4(3), pp. 356-372. 2007.

[89] K. Erol, D. Nau, V.S. Subrahmanian. "Complexity, Decidability and Undecidability Results for Domain-Independent Planning". Artificial Intelligence, vol. 72 (1-2), pp. 75-88. 1995.

[90] K. L. Myers. "A Continuous Planning and Execution Framework". AI Magazine vol. 20(4), pp. 63-69. 1999.

[91] M. Paolucci, T. Kawamura, T.R. Payne, K. Sycara. "Semantic Matching of Web Services Capabilities". Proc. of International Semantic Web Conference (ISWC 02), pp. 333-347. Jun. 2002.

[92] J. Kim, Y. Gil. "Towards Interactive Composition of Semantic Web Services". 2004 AAAI Spring Symposium Technical Reports. 2004.

[93] K. Sycara, M. Klusch, S. Widoff, J. Lu. "Dynamic Service Matchmaking among Agents in Open Information Environments". Journal ACM SIGMOD vol. 28(1), pp. 47-53. 1999.

[94] D. Jannach, K. Leopold, H. Hellwagner. "An Extensible Framework for Knowledge-Based Multimedia Adaptation". Lecture Notes in Computer Science. vol. 3029, pp. 144-153. 2004.

[95] ISO/IEC 15938-5:2003. "Information technology -- Multimedia content description interface -- Part 5: Multimedia description schemes". 2003.

[96] B. Pellan, C. Concolato. "Summarization of Scalable Multimedia Documents". Proc. of Workshop on Image Analysis for Multimedia Interactive Systems (WIAMIS 09), pp. 304-307. May 2009.

[97] M. Sachenbacher, B. C. Williams, "Solving Soft Constraints by Separating Optimization and Satisfiability", Proc. of the International Workshop on Preferences and Soft Constraints (SOFT-05), pp. 119-132. Oct. 2005.

[98] Y. Marton, S. Mohammad, P. Resnik, "Estimating Semantic Distance Using Soft Semantic Constraints in Knowledge-Source–Corpus Hybrid Models". Proc. of Empirical Methods in Natural Language Processing (EMNL 09). Aug. 2009.

[99] M. Prangl, I. Kofler, H. Hellwagner. "An MPEG-21-driven utility-based multimedia adaptation decision taking web service". Proc. of the 1st international conference on Ambient media and systems (AMS'08), pp. 1-8, Feb. 2008.

[100] H. Kodikara Arachchi, S. Dogan, H. Uzuner, and A.M. Kondoz, "Utilising Macroblock SKIP Mode Information to Accelerate Cropping of an H.264/AVC Encoded Video Sequence for User Centric Content Adaptation", Proc. of the 3rd International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS 2007), pp. 3-6. Nov. 2007.

[101] E. Danan. "Behavioral Foundation of Incomplete Preferences". Technical report, EUREQua. Université de Paris. 2001.

[102] B. Faltings, P. Pu, J. Zhang. "Agile Preference Models based on Soft Constraints", in "Challenges to Decision Support in a Changing World", AAAI Press. 2005.

[103] R. P. Weicker, “Dhrystone: a synthetic systems programming benchmark”, Communications of the ACM, vol. 27(10), pp. 1013-1030. 1984.

[104] R. Goularte, R. de Mello, E. Dodonov and L. Yang, “A model to estimate transcoding costs applied in live-video transmissions”, International Conference on Intelligent Pervasive Computing (IPC 2007), pp. 341-348, 2007.

[105] M. Prangl, T. Szkaliczki, H. Hellwagner. "A framework for Utility-Based Multimedia Adaptation". IEEE transactions on circuits and systems for video technologies, vol. 17(6), pp. 719--728. Jun. 2007.

[106] S. Kim, Y. Yoon, "Intelligent Adaptation Model based on MPEG-21 Framework", Proc. of 4th International Conference on Embedded and Multimedia Computing, 2009. EM-Com 2009. Dec. 2009.

[107] I. Kofler, C. Timmerer, H. Hellwagner, A. Hutter, and F. Sanahuja. "Efficient MPEG-21-based Adaptation Decision-Taking for Scalable Multimedia Content". Proc. of the 14th Multimedia Computing and Networking Conference (MMCN'07), pp. 65040J-1, Feb. 2007.

[108] R. Izbal, S. Shirmohammadi, “MPEG-21 based temporal video adaptation for heterogeneous devices and mobile environments”, Proc. of IEEE International Conference on Multimedia and Expo. ICME 2009. Jul. 2009.

[109] I. Klofer, J. Seidl, C. Timmerer, H. Hellwagner, I. Djama, T. Ahmed, "Using MPEG-21 for cross-layer multimedia content adaptation", Signal, Image and Video Processing Journal, vol. 2:4, pp. 355-37, Dec. 2008.

[110] H. Kodikara Arachchi, C. Hewage, S. Dogan, M. Mrak,V. Barbosa, M.T. Andrade, and A.M. Kondoz. "Context-aware adaptation of SVC scalability structure for improved coding efficiency", Proc. of 50th International Symposium ELMAR-2008, Sep. 2008.

[111] D. Renzi, P. Amon and S. Battista, "Video Content Adaptation Based on SVC and Associated RTP Packet Loss Detection and Signaling", Proc. of Ninth International Workshop on Image Analysis for Multimedia Interactive Services WIAMIS 2008, pp. 97-100, May 2008.

[112] A. García, J. Molina, F. López, V. Valdés, F. Tiburzi, J. M. Martínez, J. Bescós, "Instant Customized Summaries Streaming: a service for immediate awareness of new video content", Lectures Notes in Computer Science, Springer Verlag, under preparation, 7th International Workshop on Adaptive Multimedia Retrieval (AMR 2009), Sep. 2009.

[113] P. Bertoli, R. Kazhamiakin, M. Paolucci, M. Pistore, H. Raik, M. Wagner, "Continuous Orchestration of Web Services via Planning", Proc. of 19th International Conference on Automated Planning and Scheduling (ICAPS'09), pp. 18-25, 2009.

[114] D. Weld, "An Introduction to Least Commitment Planning", AI Magazine, vol. 15:4, pp. 27-61, Winter 1994.

[115] D. Smith and M. Peot, " Postponing Conflicts in Partial-Order Planning", Proc. of the Eleventh National Conference on Artificial Intelligence, Washington D.C. 1993.

[116] M. Paolucci, T. Kawamura, T.R. Payne, K. Sycara. "Semantic Matching of Web Services Capabilities". Proc. of International Semantic Web Conference (ISWC 02), pp. 333-347. Jun. 2002.

[117] V. Rodriguez-Doncel, J. Delgado. "A Media Value Chain Ontology for MPEG-21". IEEE Multimedia. vol. 16(4), pp. 44-51. Dec. 2009.

Universidad Autónoma de Madrid
Departamento de Ingeniería Informática

En Madrid a .... de .... de 2010.

Constituido el tribunal compuesto por:

• Presidente

– D...

• Vocales:

– D...

– D...

– D...

• Secretario:

– D...

Por cuanto Fernando López Hernández ha defendido su tesis titulada “Contributions to multimedia adaptation within the MPEG-21 framework”, este tribunal le otorga la calificación de:

....

Fdo:.. Fdo:..
(Presidente) (Secretario)

Fdo:.. Fdo:.. Fdo:..
(Vocal) (Vocal) (Vocal)