Coding format independent multimedia content adaptation using XML*

Christian Timmerer[†a], Gabriel Panis[b], Harald Kosch[a], Jörg Heuer[b], Hermann Hellwagner[a], and Andreas Hutter[b]

[a]University Klagenfurt, Department of Information Technology, A-9020 Klagenfurt, Austria

[b]Siemens AG, CT IC 2, D-81730 Munich, Germany

ABSTRACT

Due to the heterogeneity of the current terminal and network infrastructures, multimedia content needs to be adapted to specific capabilities of these terminals and network devices. Furthermore, user preferences and user environment characteristics must also be taken into consideration. The problem becomes even more complex by the diversity of multimedia content types and encoding formats. In order to meet this heterogeneity and to be applicable to different coding formats, the adaptation must be performed in a generic and interoperable way.

As a response to this problem and in the context of MPEG-21, we present an approach which uses XML to describe the high-level structure of a multimedia resource in a generic way, i.e., how the multimedia content is organized, for instance in layers, frames, or scenes. For this purpose, a schema for XML-based bitstream syntax descriptions (*generic Bitstream Syntax Descriptions* or *gBSDs*) has been developed. A gBSD can describe the high-level structure of a multimedia resource in a coding format independent way. Adaptation of the resource is based on elementary transformation instructions formulated with respect to the gBSDs. These instructions have been separated from the gBSDs in order to use the same descriptions for different adaptations, e.g., temporal scaling, SNR scaling, or semantic adaptations. In the MPEG-21 framework, those adaptations can be steered for instance by the network characteristics and the user preferences. As a result, it becomes possible for coding format agnostic adaptation engines to transform media bitstreams and associated descriptions to meet the requirements imposed by the network conditions, device capabilities, and user preferences.

Keywords: Multimedia, Adaptation, Interoperability, XML, MPEG-21, Digital Item Adaptation

1. Introduction

Today, there are many technologies in place to establish an infrastructure for the delivery and consumption of multimedia content. In practice, however, several elements of such an infrastructure are often stand-alone systems and a big picture of how these elements relate to each other or even fit together is not available. Therefore, MPEG-21 aims to provide an open framework for multimedia delivery and consumption[1]. The vision of MPEG-21 is to define an *open multimedia framework* that will enable transparent and augmented use of multimedia resources across a wide range of networks and devices. The intent is that the framework will cover the entire multimedia content delivery chain encompassing content creation, production, delivery, personalization, consumption, presentation and trade.

Within MPEG-21, one objective is to define effective multimedia adaptation support. Therefore MPEG-21, within the *Digital Item Adaptation (DIA)* activities[2], specifies description tools that will facilitate the adaptation process. This is of importance, since the access to any multimedia resource from any type of terminal or network, often referred to as *Universal Multimedia Access (UMA)*, in an interoperable way, is one of today's primary goals in the multimedia area[3,4].

In this context, this paper proposes a new adaptation approach that comprises descriptions and techniques for targeting the adaptation of multimedia resources in a (distributed) multimedia system. The main idea is to introduce a *generic Bitstream Syntax Description (gBSD)* that describes the syntax of a binary media resource (called bitstream). In most cases it does not need to detail each bit of the bitstream, but rather relies on logical units of the bitstream; for a video, this is typically the frame level. In other words, a gBSD typically describes the high-level structure of a bitstream. A

* This project was funded in part by FWF (Fonds zur Förderung der wissenschaftlichen Forschung) P14788 and by KWF (Kärntner Wirtschaftsförderungsfonds).

† christian.timmerer@itec.uni-klu.ac.at; phone +43 (463) 2700 3621; fax +43 (463) 2700 3699; www.itec.uni-klu.ac.at

resource adaptation engine can then transform the bitstream and the corresponding gBSD using editing-style operations such as data truncation and simple modifications. This transformation can, for instance, be performed under the control of an XSLT[5] style sheet. It is desirable that gBSDs are independent of specific media coding formats. This enables the adaptation of binary media resources to take place at any devices, for instance network gateways that are not knowledgeable of the specific bitstream representation format.

The remainder of this paper is organized as follows. Section 2 gives a brief overview of related work. In Section 3, we detail our adaptation architecture using gBSD. Section 4 details the XML Schema (gBS Schema) for gBSDs. Section 5 describes an application of the presented technology by means of a comprehensive example including the integration of gBSD into the MPEG-21 Multimedia Framework. Section 6 describes how an adaptation application based on gBSD can be steered. Section 7 presents performance evaluation results of our implementation. Finally, Section 8 concludes and points to future work.

2. Related work

In the sequel, we give a brief overview of related technologies. The *Bitstream Syntax Description Language (BSDL)*[6,7] approach was originally inspired by XML publishing frameworks such as Cocoon (http://xml.apache.org/cocoon/) developed by the Apache Software Foundation to publish XML content adapted to the requesting client in a Web context. BSDL is very close to the approach presented in this paper. Nevertheless, there are some significant differences in the functionalities of the techniques which we attempt to highlight in the following.

BSDL is a language, also normatively specified within MPEG-21 DIA[2], based on XML Schema which makes it possible to develop specific Bitstream Syntax (BS) Schemas describing the syntax of particular media resource formats. These schemas can then be used by a generic processor/software to automatically parse a bitstream and generate its description, and vice versa. On the one hand, a BintoBSD parser is a generic process using a BS Schema to parse a bitstream and generate the corresponding BS Description. On the other hand, a BSDtoBin parser is a generic process using a BS Schema to parse a possibly transformed BS Description and generate the corresponding bitstream. The requirement that the coding format specific BS Schema must be completely available on the device when generating the description or the bitstream makes this approach inapplicable for adaptations to be performed on highly constraint devices, for instance, on network nodes such as gateways.

Flavor, a language for media representation[8], is a technology which has similarities, in some aspects, with the gBSD approach. Flavor, which stands for *F*ormal *L*anguage for *A*udio-*V*isual *O*bject *R*epresentation, was initially designed as a declarative language with a C++-like syntax to describe the bitstream syntax on a bit-per-bit basis. Its aim is to simplify and speed up the development of software that processes audio-visual bitstreams by automatically generating the required C++ and Java code to parse the data, hence allowing the developer to concentrate on the processing part of the software. Unlike the gBSD approach, it does not generate a permanent description of the parsed data, but only an internal memory representation in the form of a collection of C++ or Java objects.

Recently, Flavor was enhanced to support XML features (*XFlavor*)[9] and tools for generating an XML description of the bitstream syntax, transforming the XML description and regenerating the adapted bitstream[9]. For the bitstream generation, unlike gBSD which relies on the XML Schema datatype definition, XFlavor uses proprietary attributes declared in the schema and indicating the binary encoding of the element content. A major drawback of this design is that two elements with the same type (e.g., encoded on two bits) will require the same verbose declaration in the schema, while gBSD will define the corresponding type once and use it for all elements. Furthermore, using XML Schema native mechanisms for datatype definition and derivation allows validating the value and optimally encoding it with tools such as MPEG-7 BiM[10]. Lastly, in XFlavor, the complete bitstream data are actually embedded in the XML document, resulting in potentially huge descriptions, while gBSD uses a specific datatype to point to a data range in the original bitstream when it is too verbose to be included in the description. However, the main differences lie in the generic, universal and codec independent format of generic Bitstream Syntax Descriptions (gBSDs), and in the availability of means to mark individual elements in the gBSD in order to enable and simplify complex and semantic-based adaptations. Such marking is not possible with XFlavor.

3. Adaptation architecture using gBSD

The motivation for multimedia content adaptation is to deliver the best possible content in terms of information and perceived quality to the end user given the current set of constraints, e.g., terminal and network capabilities as well as user preferences. A gBSD-based Digital Item Adaptation engine is depicted in **Figure 1**.

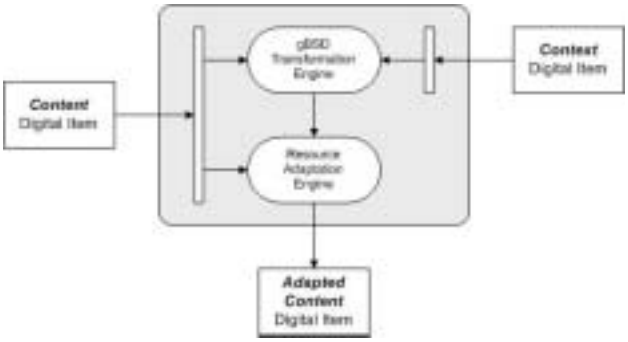

Figure 1: gBSD-based Digital Item Adaptation engine

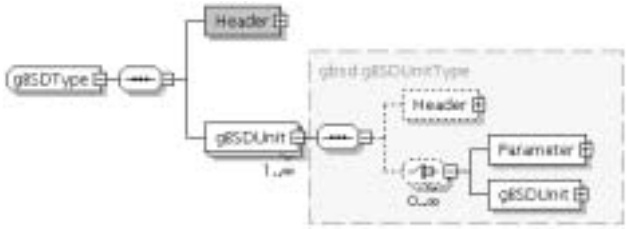

Figure 2: gBS Schema elements

The *Content Digital Item (CDI)* is referred to as a Digital Item (DI) that is used for the delivery of media resources and contains the multimedia content (bitstream) and its gBS Description among other descriptors not necessarily required for the adaptation. Assume that the multimedia content is only available in high quality and it is required to adapt this content to satisfy various constraints. On the other hand, the *Context Digital Item (XDI)* is referred to as a DI that is used for the delivery of metadata only, i.e., the constraints given by several users along the delivery path such as usage environment, user characteristics, terminal capabilities, network characteristics, or natural environment characteristics. Descriptions of the so-called constraints are also specified in MPEG-21 DIA[2]. Note that a user in this context is not restricted to the person – often called end user – consuming the content. It may include also the provider or even network nodes along the delivery path. The *gBSD Transformation Engine* transforms the gBSD governed by metadata contained within the XDI, e.g., by applying an appropriate parameterized XSLT style sheet to the gBSD. The output of this transformation process (i.e., the transformed gBSD) is used by the *Resource Adaptation Engine* which implements the gBSDtoBin process described in Section 4.3. Finally, the adapted bitstream and the updated/transformed gBSD are used to generate the *Adapted Content Digital Item*.

The adaptation process itself happens without any interaction from the end user who is only interested in consuming the content in the highest possible quality. In other words, this adaptation architecture enables interoperable and transparent access to (distributed) multimedia content by shielding users from network and terminal installation, management, and implementation issues.

4. Generic Bitstream Syntax Schema (gBS Schema)

The gBS Schema is targeting enhanced binary resource adaptations. The syntax of a gBSD is generic and coding-format independent, therefore enabling the processing of binary resources without the need of coding-format specific schemas. Additionally, the gBS Schema provides the following functionalities:

- Semantically meaningful marking of syntactical elements described by use of a “marker” handle.
- Description of a bitstream in a hierarchical fashion that allows grouping of bitstream elements for efficient, hierarchical adaptations.
- Flexible addressing scheme to support various application requirements and random accessing of the bitstream.

The unique functionality that the gBS Schema provides, facilitates complex adaptations. Specifically, the “marking” of elements and the hierarchical structure of the description can greatly simplify sophisticated bitstream manipulations (e.g., remove an SNR layer from a JPEG2000 bitstream when the progression order is resolution) and semantic-related adaptations (e.g., remove violent scenes from a video sequence). Furthermore, the gBSDtoBin process which generates

the (adapted) bitstream from a (transformed) gBSD does not require an application or codec-specific schema, since all the necessary information to regenerate the bitstream is included in the gBSD and the semantics of the gBS Schema elements. This is particularly desired when the adaptation takes place on remote, constraint devices, since it saves bandwidth, memory, and computational resources.

4.1. gBS Schema elements

The complete gBS Schema is currently specified in the MPEG-21 Digital Item Adaptation Final Committee Draft (FCD)[2]. This section gives an overview of the most important elements of the gBS Schema. Example gBS Descriptions (gBSDs) illustrating the concepts described in the following are given in Section 5.

The gBS Schema defines three types of gBSD elements (**Figure 2**): *Header*, *gBSDUnit*, and *Parameter*.

Header. The *Header* element contains information which applies to the whole gBSD such as MPEG-7 classification schemes to be used and default values required for resolving address references in the gBSD. In particular, the Header element comprises the alias and URI of a classification scheme which provides the semantics of the names of the gBSDUnits and Parameters (e.g., identifying the frame type of a video bitstream). These names allow a codec aware adaptation engine to perform bitstream adaptations in an interoperable and extensible way because classification schemes are standardized and can be easily extended by simply adding new terms. In addition, the Header element defines default address values such as the URI of the bitstream location, address mode (absolute, consecutive, or offset) and address units (bit or byte) to be used.

The scope of these default values is the complete gBSD but these can be overwritten by additional Header elements within a gBSDUnit element or by address attributes within a gBSDUnit or Parameter element.

gBSDUnit. The *gBSDunit* element represents a segment of the bitstream but does not include the actual values of that segment. Therefore, it shall be used for bitstream segments which might be removed during adaptation (e.g., frame dropping) or for carrying further gBSDUnit or Parameter elements, which then results in a hierarchical structure for efficient bitstream manipulations. It can also be used to represent blocks of data in the bitstream that do not play any role in the adaptation process. Each gBSDUnit carries several possible attributes:

- *start* and *length* — attributes conveying addressing information depending on the address mode.
- *syntacticalLabel* — attribute for including coding-format specific information identified via classification scheme terms.
- *marker* — attribute which provides a handle for application-specific (e.g., semantic) information to be used for performing complex bitstream adaptations in an efficient way.

In addition, it is also possible to include attributes specifying address information normally carried in the Header element within a gBSDUnit element. This means that each gBSDUnit may also contain its own bitstream segment location, address mode and address unit, which allows locating parts of the bitstream on different devices within a distributed multimedia environment.

Parameter. The *Parameter* element describes a syntactical element of the bitstream the value of which might need to be changed during the adaptation process. Therefore, it provides the actual numerical value and datatype of the bitstream segment. The datatype is required for correctly encoding the syntactical element in the bitstream during the gBSDtoBin process. Due to the multitude of possible datatypes they are specified by using the `xsi:type` attribute in the instance instead of within the gBS Schema. Nevertheless, frequently used basic datatypes are specified within an additional XML schema. Similar to the gBSDUnit element, the Parameter element may also contain its own address information.

4.2. gBSD generation

The behavior of the BintogBSD (*Bin*ary *to g*eneric *B*itstream *S*yntax *D*escription) process is not normatively specified because there are many possibilities for generating a gBSD describing a bitstream. One possibility would be to

generate the gBSD during the encoding process of the bitstream because the encoder best knows the structure and adaptation possibilities to be "encoded" into the gBSD. However, this requires enhancing the encoder with the gBSD generation functionality. Another possibility is to use the BSDL approach to generate a coding format specific BS Description which will be subsequently transformed into a corresponding gBSD using, for instance, an XSLT style sheet.

In both cases, it is possible to include some additional information in the marker attribute or introduce some additional hierarchical levels enabling faster and more efficient bitstream adaptations. This additional processing step may be governed by other metadata, e.g., MPEG-7 descriptors describing the violence level of scenes in a video.

4.3. Bitstream generation using gBSD

The behavior of the gBSDtoBin (*g*eneric *B*itstream *S*yntax *D*escription *to Bin*ary) process is normatively specified in the MPEG-21 DIA FCD and generates the bitstream by using the information carried in a possibly transformed gBSD. The gBSDtoBin process is illustrated in **Figure 3**.

Figure 3: Bitstream generation with gBSDtoBin

Figure 4: Use case scenario

The gBSDtoBin process hierarchically parses the gBSD and constructs the corresponding bitstream. For that purpose, it uses the addressing information from the Header elements or attributes of the gBSDUnit or Parameter elements for locating the bitstream segments to be copied to the resulting bitstream. Additionally, the Parameter element contains the required information for correctly encoding the syntactical elements into the resulting bitstream.

In order to enable further adaptation steps (which may be located on different devices) it is necessary to update the transformed gBSD with respect to the adapted bitstream. This is for example required in a multi-step adaptation scenario where a server drops B-frames because the end devices are not able to decode these frame types and a home gateway removes violent scenes according to the age of the users.

5. Application example

Images and videos represent two of the most commonly used content types in multimedia systems. JPEG2000[11] and MPEG-4 Visual[12] are well known standard coding formats which offer various types of scalability, i.e. temporal, spatial, SNR, and color adaptability among others. This paper concentrates on a comprehensive multi-step example using MPEG-4 Visual Elementary Streams (ES). MPEG-4 video coding is based on the well-known motion-compensated hybrid DCT coding scheme. Therefore, I-, P- and B-frames are subject to adaptation required, e.g., for temporal scalability. Every MPEG-4 Visual ES comprises several Video Object Planes (VOPs) classified as different types such as I-, P- and B-VOPs, among others. Each ES is headed by some configuration or header information depending on the ES entry point[13]. The complete hierarchy of an MPEG-4 Visual ES is illustrated in detail in the corresponding ISO/IEC specification[12]. A scene from the famous movie "The Lord of the Rings: The Two Towers" was selected to demonstrate the gBSD approach.

The example implements the following use case scenario illustrated in **Figure 4**. A home network consists of a home gateway, which also implements a simple proxy server with a gBSD-capable adaptation engine, and several end devices. All end devices support only the MPEG-4 Simple Profile[12]. Therefore, it does not make sense to transmit the full-quality video over the Internet, i.e., the multimedia data server already drops the B-frames (first adaptation step). The second adaptation step takes place on the home gateway where the video is consumed by different family members of different ages, according to their user profiles (also stored on the home gateway). This requires for example that violent scenes above a given threshold have to be removed from the video depending on the user's age.

A fragment of the "Lord of the Rings" CDI is shown in **Document 1**. The CDI may also comprise a Digital Item Identifier (DII) which uniquely identifies the DI, a DII type, or additional MPEG-7 descriptors. The Component of the MPEG-21 Digital Item Declaration (DID)[14] comprises the bitstream within the Resource element and its gBSD.

```
<did:DIDL xmlns="urn:mpeg:mpeg21:2003:01-DIA-gBSD-NS" xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
  xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS" xmlns:did="urn:mpeg:mpeg21:2002:01-DIDL-NS"
  xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <did:Item>
      <!-- ... Identifier, Type, and MPEG-7 descriptions here ... -->
      <did:Component id="lotr2">
         <did:Descriptor id="lotr2_gBSD">
            <did:Statement mimeType="text/xml">
               <dia:DIA>
                  <dia:Description xsi:type="gBSDType">
                     <Header>
                        <ClassificationAlias alias="MV4"
                              href="urn:mpeg:mpeg4:video:cs:syntacticalLabels"/>
                        <DefaultValues addressUnit="byte" addressMode="Absolute"
               globalAddressInfo="#xpointer(//did:Component[@id='lotr2']/did:Resource/@ref)"/>
                     </Header>
                     <gBSDUnit syntacticalLabel=":MV4:VO" start="0" length="4"/>
                     <gBSDUnit syntacticalLabel=":MV4:VOL" start="4" length="14"/>
                     <gBSDUnit syntacticalLabel=":MV4:I_VOP" start="18" length="12270"/>
                     <gBSDUnit start="12288" length="47768" marker="Violence-6">
                        <gBSDUnit syntacticalLabel=":MV4:P_VOP" start="12288" length="7589"/>
                        <gBSDUnit syntacticalLabel=":MV4:B_VOP" start="19877" length="3218"/>
                        <gBSDUnit syntacticalLabel=":MV4:B_VOP" start="23095" length="3371"/>
                        <gBSDUnit syntacticalLabel=":MV4:B_VOP" start="26466" length="3511"/>
                        <gBSDUnit syntacticalLabel=":MV4:P_VOP" start="29977" length="30079"/>
                     </gBSDUnit>
                     <gBSDUnit start="60056" length="1739135" marker="Violence-4">
                        <gBSDUnit syntacticalLabel=":MV4:B_VOP" start="60056" length="12015"/>
                        <gBSDUnit syntacticalLabel=":MV4:B_VOP" start="72071" length="10353"/>
                        <gBSDUnit syntacticalLabel=":MV4:P_VOP" start="82424" length="15695"/>
                        <!--... and so on ...-->
                     </gBSDUnit>
                     <!--... and so on ...-->
                  </dia:Description>
               </dia:DIA>
            </did:Statement>
         </did:Descriptor>
         <did:Resource ref="content/lotr2.cmp" mimeType=" video/MP4V-ES"/>
      </did:Component>
   </did:Item>
</did:DIDL>
```

Document 1: The "Lord of the Rings" Content Digital Item (CDI) including gBSD information

Document 2 shows the XDI which specifies the decoding format supported by the end devices (terminals) in the home network. These terminal capabilities are collected by the home gateway and transmitted to the multimedia data server. The multimedia data server uses this information for performing the desired adaptation accordingly, i.e., in the case given below the server drops all B-frames because MPEG-4 Simple Profile does not support B-frames. The XDI may also contain an identifier and a type.

```
<did:DIDL xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS" xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
  xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS" xmlns:did="urn:mpeg:mpeg21:2002:01-DIDL-NS"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```
   <did:Item>
      <!-- ... Identifier, Type, and MPEG-7 descriptions here ... -->
      <did:Descriptor>
         <did:Statement mimeType="text/xml">
            <DIA><Description xsi:type="UsageEnvironmentType" id="videoformat">
                  <TerminalCapabilities xsi:type="CodecCapabilities">
                     <Decoding xsi:type="VideoCapabilitiesType">
                        <Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.1">
                           <mpeg7:Name xml:lang="en">MPEG-4 Simple Profile</mpeg7:Name>
                        </Format>
                     </Decoding>
                  </TerminalCapabilities>
               </Description></DIA>
         </did:Statement>
      </did:Descriptor>
   </did:Item>
</did:DIDL>
```

Document 2: Usage Environment Descriptor (XDI) for B-frame dropping

Document 3 shows the XDI containing the user preferences which specifies the parental rating value of a given end user. The home gateway filters out the scenes below the given threshold and submits only the scenes to the end user which she/he is allowed to watch according to the user profile stored on the home gateway.

```
<did:DIDL xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS" xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
  xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS" xmlns:did="urn:mpeg:mpeg21:2002:01-DIDL-NS"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <did:Item>
      <!-- ... Identifier, Type, and MPEG-7 descriptions here ... -->
      <did:Descriptor id="diaUED">
         <did:Statement mimeType="text/xml">
            <DIA><Description xsi:type="UsageEnvironmentType" id="contentpreferences">
                  <UserCharacteristics xsi:type="ContentPreferencesType">
                     <UserPreferences>
                        <mpeg7:FilteringAndSearchPreferences>
                           <mpeg7:ClassificationPreferences>
                              <mpeg7:ParentalGuidance>
                                 <mpeg7:ParentalRating
                                      href="urn:mpeg:mpeg7:cs:ICRAParentalRatingViolanceCS:2001:4">
                                    <mpeg7:Name>Level 2</mpeg7:Name>
                                 </mpeg7:ParentalRating>
                                 <mpeg7:Region>at</mpeg7:Region>
                              </mpeg7:ParentalGuidance>
                           </mpeg7:ClassificationPreferences>
                        </mpeg7:FilteringAndSearchPreferences>
                     </UserPreferences>
                  </UserCharacteristics>
               </Description></DIA>
         </did:Statement>
      </did:Descriptor>
   </did:Item>
</did:DIDL>
```

Document 3: Usage Environment Descriptor (XDI) for parental filtering

Document 4 shows a generic XSLT style sheet for removing elements from a gBSD identified via the target parameter (targetParameter). The target parameter comprises attributes or elements of the gBSD which are possibly affected by the gBSD transformation process. Therefore, the target parameter is dynamically evaluated during this transformation process. The source parameter (sourceParameter) specifies the value which should be compared to the target parameter be means of the source parameter specification (sourceParameterSpec): “1” denotes “less or equal” (≤), “2” denotes “greater or equal” (≥), and “3” denotes “equal” (=). For the dynamic evaluation of the target parameter and the comparison of source and target parameters, XSLT’s escape mechanism into the procedural programming paradigm is used, i.e., simple JavaScripts are used for this purpose.

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:gbsd="urn:mpeg:mpeg21:2003:01-DIA-gBSD-NS" xmlns:dyn="urn:uni-klu:itec:myExt">
   <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
```

```
   <xsl:include href="MSXSL_extensions.xslt"/>
   <xsl:param name="sourceParameter" select="'Violence-4'"/>
   <xsl:param name="sourceParameterSpec" select="'1'"/>
   <xsl:param name="targetParameter" select="'@marker'"/>
   <xsl:template name="tpl_gBSDUnit" match="@*|node()">
      <xsl:variable name="targetParameterValue" select="dyn:evaluate(., $targetParameter)"/>
      <xsl:choose><xsl:when test="$targetParameterValue">
            <xsl:choose>
               <xsl:when test="$sourceParameterSpec=1">
                  <xsl:choose><xsl:when test="dyn:strLE(string($targetParameterValue),
                                          string($sourceParameter))"><!-- do nothing -->
                     </xsl:when><xsl:otherwise>
                        <xsl:copy><xsl:apply-templates select="@*|node()"/></xsl:copy>
                     </xsl:otherwise>
                  </xsl:choose>
               </xsl:when>
               <xsl:when test="$sourceParameterSpec=2">
                  <xsl:choose><xsl:when test="dyn:strGE(string($targetParameterValue),
                                          string($sourceParameter))"><!-- do nothing -->
                     </xsl:when><xsl:otherwise>
                        <xsl:copy><xsl:apply-templates select="@*|node()"/></xsl:copy>
                     </xsl:otherwise>
                  </xsl:choose>
               </xsl:when>
               <xsl:when test="$sourceParameterSpec=3">
                  <xsl:choose><xsl:when test="dyn:strEQ(string($targetParameterValue),
                                          string($sourceParameter))"><!-- do nothing -->
                     </xsl:when><xsl:otherwise>
                        <xsl:copy><xsl:apply-templates select="@*|node()"/></xsl:copy>
                     </xsl:otherwise>
                  </xsl:choose>
               </xsl:when>
               <xsl:otherwise><xsl:message terminate="yes">unknown source parameter specification
value</xsl:message></xsl:otherwise>
            </xsl:choose>
         </xsl:when>
         <xsl:otherwise><xsl:copy><xsl:apply-templates select="@*|node()"/></xsl:copy>
         </xsl:otherwise>
      </xsl:choose>
   </xsl:template>
</xsl:stylesheet>
```

Document 4: XSLT style sheet for removing gBSDUnit elements

The current example (**Document 4**) shows the parameters for the “parental filtering” use case. For the “B-frame dropping” use case, the source parameter (`sourceParameter=":MV4:P_VOP"`) should be changed to the value of the syntactical label (`targetParameter="@syntacticalLabel"`). Accordingly, its specification should be changed to equal (`sourceParameterSpec="3"`).

Document 5 shows the fragment of the transformed and already updated gBSD after the adaptation steps “B-frame dropping” and “parental filtering”. The updated attributes of the gBSD are shown in boldface.

```
<did:DIDL xmlns="urn:mpeg:mpeg21:2003:01-DIA-gBSD-NS" xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
  xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS" xmlns:did="urn:mpeg:mpeg21:2002:01-DIDL-NS"
  xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS">
   <did:Item>
      <!-- ... Identifier, Type, and MPEG-7 descriptions here ... -->
      <did:Component id="lotr2">
         <did:Descriptor id="lotr2_gBSD">
            <did:Statement mimeType="text/xml">
               <dia:DIA>
                  <dia:Description xsi:type="gBSDType">
                     <Header>
                        <ClassificationAlias alias="MV4"
                              href="urn:mpeg:mpeg4:video:cs:syntacticalLabels"/>
                        <DefaultValues addressUnit="byte" addressMode="Absolute"
               globalAddressInfo="#xpointer(//did:Component[@id='lotr2']/did:Resource/@ref)"/>
```

```
                    </Header>
                    <gBSDUnit syntacticalLabel=":MV4:VO" start="0" length="4"/>
                    <gBSDUnit syntacticalLabel=":MV4:VOL" start="4" length="14"/>
                    <gBSDUnit syntacticalLabel=":MV4:I_VOP" start="18" length="12270"/>
                    <gBSDUnit start="12288" length="37668" marker="Violence-6">
                       <gBSDUnit syntacticalLabel=":MV4:P_VOP" start="12288" length="7589"/>
                       <gBSDUnit syntacticalLabel=":MV4:P_VOP" start="19877" length="30079"/>
                    </gBSDUnit>
                    <!--... and so on ...-->
                 </dia:Description>
              </dia:DIA>
           </did:Statement>
        </did:Descriptor>
        <did:Resource ref="content/lotr2.cmp" mimeType=" video/MP4V-ES"/>
     </did:Component>
  </did:Item>
</did:DIDL>
```

Document 5: Transformed and updated gBSD

6. Steering gBSD-based adaptations

In the previous sections, the required components to perform generic bitstream adaptations were explained and illustrated. In the following, the initiation of the adaptation shall be emphasized. Requests to adapt bitstreams are usually triggered by the information receiver. The request contains information (in the form of Digital Items) to generate the modified media resource on demand, namely: the original resource, the gBSD of the original resource, and the Transformation Instructions, formulated in XSLT, for instance.

All three information assets have specifically to be compiled to specify the generation of the described and requested Digital Item. To ease the referencing of these information assets, references to these assets can be stored in a *BSDLink*. This description contains a reference to the gBSD of the media resource (which in turn contains a reference to the resource proper) and a reference to the Transformation Instructions.

With the current approach of the BSDLink, this requires to specify for each possible adaptation the information assets and a BSDLink. It is obvious that most of the information contained in the separate information assets needed for Digital Item Adaptation and the BSDLink is highly redundant.

In practice, however, Transformation Instructions can be categorized into the same *remove* and *update* transformations for a specific gBSD. Most often for the same resource type, the Transformation Instructions only differ with respect to the units of the gBSD they are applied to or the values they are manipulating. Accordingly, in order to reduce the mentioned redundancy, the gBSD, the Transformation Instructions and the BSDLink are parameterized with respect to the applicability of the Transformation Instructions. Therefore, the elements (gBSDUnit, Parameter) within a gBSD are labeled with respect to the possible adaptations. Those labels can be used to signal – in the form of parameters – which gBSD elements the Transformation Instructions apply to or how values represented in those gBSD elements have to be manipulated.

An example of such a BSDLink is given in **Document 6**. This example includes already improvements which are currently not specified within MPEG-21 DIA[2], namely the qualified names for the *Parameter* elements and the additional *Specification* element. These improvements are currently subject to several validation experiments within MPEG.

```
<DIA xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <Description xsi:type="BSDLinkType">
      <SteeringDescriptionRef uri="AQoS.xml"/><BSDRef uri="gBSD.xml"/>
      <BSDTransformationRef uri="transform.xslt"/>
      <Parameter xsi:type="IOPinRefType" name="urn:mpeg:mpeg21:2003:01-DIA-BSDLink-NS:SourceParam">
         <Value>VIOLENCE_LEVELS</Value><Specification>1</Specification>
      </Parameter>
      <Parameter xsi:type="ConstantType" name="urn:mpeg:mpeg21:2003:01-DIA-BSDLink-NS:TargetParam">
         <Value>@marker</Value></Parameter>
   </Description>
```

```
</DIA>
```

Document 6: BSDLink example

The described parameterization of BSDLink, gBSD and Transformation Instructions supports a highly flexible configuration of adaptations. For instance, it becomes possible to describe the results of possible adaptations based on the parameter sets with respect to the quality of service the delivery chain has to provide. These descriptions are called *Adaptation Quality of Service (AQoS) Descriptions*. The AQoS Descriptions can be generated when the parameterized BSDLink, gBSD and Transformation Instructions are specified. Based on the AQoS Description and constraints described in the XDI, a decision taking engine can steer an adaptation which results in a transmitted resource meeting the specified constrains. Thus, a decision taking process can decide before the actual adaptation which one fits best the needs signaled by the XDI.

The current approach within MPEG-21 DIA is focused on steering adaptations at request time. However, in streaming applications also dynamic changes of constraints have to be considered. In these cases, especially the AQoS Description and the specification of Transformation Instructions have to be investigated further. To that end, efficient representations of streamed AQoS descriptions as well as the proposal of Streaming Transformation for XML (STX) [15,16] will be studied.

7. Performance measurements

In order to evaluate the feasibility and applicability of the presented gBSD approach in real-world applications, the performance of the approach was measured. Two criteria were used that are regarded as important for many applications. The criteria are:

- The *file size* of the DID containing the gBSD. The file size is especially critical in distributed adaptation scenarios, where the gBSD is transmitted along with the media resource and therefore represents an overhead. It is desired that this overhead is as low as possible, relative to the multimedia resource's file size. The file size of the original and adapted DIDs was measured both in textual form and in compressed form. For the encoding and compression of the XML files the Binary Model (BiM) encoding scheme defined in the MPEG-7 Systems standard[10] was used. There is currently an activity within MPEG to enhance BiM towards a version 2, with functionality that improves encoding efficiency; hence the latest version of the scheme was used for the experiments presented here.
- The *processing requirements* for a gBSD-based adaptation process. The processing time for each step of the adaptation process was selected as an indicative measure of the requirements imposed by the process in terms of processing power. The characteristics of the machine on which the experiments were performed are provided.

Two media resources have been selected for performing the experiments. The first is a fragment extracted from the movie "The Lord of the Rings: The Two Towers" (LOTR), encoded using the MPEG-4 Visual Advanced Simple Profile scheme, with resolution 480x480 pixel, at 30 frames/sec. This bitstream was used in the application example presented in Section 5. The second is an image portraying the skyline of Shanghai, China (SHANGHAI), encoded using the JPEG2000 encoding scheme[11], at 1024x768 resolution, 3.34 bpp bitrate, progression type 1 (resolution scalability first).

7.1. Description size

Table 1 presents the measurements of the file sizes for two bitstreams and their associated DIDs. The DIDs were kept simple, mainly including the gBSDs, as shown in the example application example in Section 5 (see **Document 1**). For the video bitstream, two adaptations (B-frame dropping and violence removal) were performed in sequence. For the image bitstream, three adaptations (spatial resolution, SNR reduction, and luminance removal) were independently performed. In the table, the sizes of the original bitstream and corresponding DID as well as the adapted resource and DID after each adaptation step are listed. The sizes of the DID files were measured in both the textual and the BiM-encoded forms. Due to the fact that certain XML Schema elements used in MPEG-21 DID[14] are not supported by version 1 of the BiM, the schema need to be slightly changed. MPEG-21 DID allows including any XML data within

several elements. Therefore, the affected elements have been changed to allow only carrying XML data needed in the examples. (It is expected that in version 2 which is currently being standardized all XML Schema elements will be supported.) In order to obtain a measure of the overhead that the DIDs impose on the bitstreams, the ratio of the description (textual and encoded) to the bitstream size was also calculated.

Digital Item	Bitstream size [bytes]	DID size [bytes]	BiMed size [bytes]	DID/ Bitstream	BiM/ Bitstream
original					
LOTR	12,858,782	246,589	29,309	1.92%	0.23%
SHANGHAI	328,413	245,751	19,663	74.83%	5.99%
adapted					
LOTR_1	9,255,890	140,541	15,236	1.52%	0.16%
LOTR_2	9,761,780	181,887	21,921	1.86%	0.22%
LOTR_3	6,889,244	97,774	10,401	1.42%	0.15%
SHANGHAI_1	106,772	88,694	7,097	83.07%	6.65%
SHANGHAI_2	68,206	163,526	13,300	239.75%	19.50%
SHANGHAI_3	23,317	63,018	4,967	270.27%	21.26%

Table 1: File sizes of original and adapted DIDs (including the gBSDs) for the "The Lord of the Rings" MPEG-4 Visual ES and the "Shanghai" JPEG2000 image

For the LOTR_1 DI all B-frames have been removed from the video bitstream fragment in order to satisfy the terminal's constraints. The LOTR_2 DI includes an adapted version with scenes only that have an ICRA (http://www.icra.org/) parental rating above three. For the LOTR_3 DI both adaptation techniques described before have been applied, i.e., B-frame dropping and parental filtering.

The SHANGHAI_1 DI contains a grey scale JPEG2000 image instead of a colored. For the SHANGHAI_2 DI the resolution of the JPEG2000 image has been reduced from 1024x768 to 256x192 pixels. The SHANGHAI_3 DI comprises the JPEG2000 image with reduced resolution and removed color components, i.e., a grey scale image with resolution 256x192 pixels.

For the video resource the gBSD overhead is very small, even in the textual form of the description. The reason that the gBSDs of the image are large is because the level of detail required for the adaptation of such resources is high, making the descriptions very verbose. The overhead is significantly reduced through binarization of the descriptions. For size-critical applications, the optional `syntacticalLabels` and names can be omitted, further reducing the size of the descriptions. The gBSDs used in these experiments included `syntacticalLabels` and names for each element.

7.2. Processing time

In **Table 2** below the measurements of the time required by the adaptation process are presented. The adaptation process is broken down into each step, namely the parsing/validating of the DID and extraction of the gBSD, the transformation of the gBSD using an XSLT style sheet, the generation of the adapted bitstream from the transformed gBSD, and the aggregation of the adapted DID containing the transformed gBSD. The experiments were conducted on a computer with the following characteristics: Intel Pentium M at 1.6 GHz with 512 MB of RAM

Processing step	Processing time [ms] LOTR_2	Processing time [ms] SHANGHAI_3
DID parsing/validating and description extraction	2,724	2,454
gBSD Transformation	451	711
gBSDtoBin	3,796	351
DID generation	570	280
Sum	6,370	2,885

Table 2: Processing times for each step of the adaptation process

The important measurements with respect to the performance of the technology presented in this paper are the *gBSD Transformation* and the *gBSDtoBin* process. The time for transforming the gBSD is low whereas the time for the bitstream generation (gBSDtoBin) increases with the size of the bitstream. Please note that in this example the gBSDtoBin is performed on the entire bitstream. This can be neglected in a streaming scenario where gBSD transformation and bitstream generation goes hand in hand.

8. Conclusion

Adaptation of multimedia resources is expected to play a key role in the realization of the Universal Multimedia Access vision. Furthermore, adaptation significantly enhances the quality of service for the delivery of multimedia content and will therefore provide a competitive edge for service providers. An elegant and simple solution, within the MPEG-21 DIA framework, for the application of multimedia adaptation was presented in this paper. This solution is based on the concept of using an XML description of a media bitstream's syntax (gBSD) to abstract the adaptation process from the encoding format specifics of the media resource. The presented gBS Schema is generic in terms of the resource type and encoding format; therefore the gBSD and the related adaptation process is applicable across all multimedia resource types and codecs. This significantly simplifies the design and maintenance of the adaptation engine. Furthermore, the gBS Schema provides functionality (flexible addressing scheme, "marker" handles) that increases the efficiency of the adaptation process, particularly in the case of complex and semantic adaptations.

Several enhancements of the presented technology as well as the MPEG-21 adaptation framework are envisioned and will be pursued in the future. The future work will include the testing of the technology in a distributed and dynamically changing streaming environment as well as the design of a complete decision taking engine that will utilize the MPEG-21 DIA defined descriptors to make a decision on the optimal adaptation for a given user.

REFERENCES

1. ISO/IEC JTC1/SC29/WG11/N5231 (editors: J. Bormans and K. Hill), *MPEG-21 Overview v.5*, http://www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm, Shanghai, China, October 2002.
2. ISO/IEC JTC1/SC29/WG11/N5845 (editors: A. Vetro and C. Timmerer), *Text of ISO/IEC 21000-7 FCD – Digital Item Adaptation*, Trondheim, Norway, July 2003, http://www.chiariglione.org/mpeg/working_documents.htm.
3. G. Panis, A. Hutter, J. Heuer, H. Hellwagner, H. Kosch, C. Timmerer, S. Devillers and M. Amielh, *Binary Multimedia Resource Adaptation Using XML Bitstream Syntax Description*, to appear in: Signal Processing: Image Communication Special Issue on Multimedia Adaptation, 2003.
4. A. Perkis, Y. Abeljaoued, C. Cristopoulos, T. Ebrahimi, and J.F. Chicaro, *Universal Multimedia Access from Wired and Wireless Systems*, *Transactions on Circuits*, Systems and Signal Processing. Special Issue on Multimedia Communications, Vol. 20, No. 3, 2001, pp. 387–402, Birkhauser, Boston.
5. W3C, *XSL Transformations (XSLT), Version 1.0*, W3C Recommendation, November 16, 1999, http://www.w3.org/TR/xslt.
6. M. Amielh and S. Devillers, *Multimedia Content Adaptation with XML*, 8th International Conference on Multimedia Modeling (MMM'2001), pp. 127-145, Amsterdam, Netherlands, November 5-7, 2001.
7. M. Amielh and S. Devillers, *Bitstream Syntax Description Language: Application of XML-Schema to Multimedia Content*, 11th International World Wide Web Conference (WWW 2002), Honolulu, May 6-11, 2002
8. A. Eleftheriadis, *Flavor: A Language for Media Representation*, ACM Multimedia Conference, pp. 1-9, Seattle, WA, November 1997.
9. D. Hong and A. Eleftheriadis, *XFlavor: Bridging Bits and Objects in Media Representation*, IEEE International Conference on Multimedia and Expo (ICME2002), Lausanne, Switzerland, August 2002.
10. ISO/IEC 15938-1:2002: *Information Technology – Multimedia content description interface – Part 1: Systems.*
11. ISO/IEC 15444-1:2000: *Information technology – JPEG 2000 image coding system – Part 1: Core coding system.*
12. ISO/IEC 14496-2:2001: *Information Technology - Generic coding of audio-visual objects – Part 2: Visual.*
13. F. Pereira and T. Ebrahimi (eds.), *The MPEG-4 Book*, Prentice Hall PTR, 2002.
14. ISO/IEC 21000-3:2002: *Information Technology — Multimedia Framework — Part 2: Digital Item Declaration.*
15. Oliver Becker, *Transforming XML on the Fly*, Proceedings XML Europe 2003 Conference, London, May 2003.
16. *Streaming Transformations for XML (STX) Version 1.0*, Working Draft, May 2003, http://stx.sourceforge.net/documents/spec-stx-20030505.html.