biblio.ugent.be

The UGent Institutional Repository is the electronic archiving and dissemination platform for all UGent research publications. Ghent University has implemented a mandate stipulating that all academic publications of UGent researchers should be deposited and archived in this repository. Except for items where current copyright restrictions apply, these papers are available in Open Access.

This item is the archived peer-reviewed author-version of:

Format-independent and metadata-driven media resource adaptation using semantic web technologies

Davy Van Deursen, Wim Van Lancker, Sarah De Bruyne, Wesley De Neve, Erik Mannens, and Rik Van de Walle

In: Multimedia Systems, Volume 16, Number 2, March, 2010

http://www.springerlink.com/content/20434747039865n5/

To refer to or to cite this work, please use the citation to the published version:

D. Van Deursen, W. Van Lancker, S. De Bruyne, W. De Neve, E. Mannens, and R. Van de Walle (2010). Format-independent and metadata-driven media resource adaptation using semantic web technologies. *Multimedia Systems* 16(2) pp. 85-104. 10.1007/s00530-009-0178-9

Noname manuscript No.
(will be inserted by the editor)

Format-independent and Metadata-driven Media Resource Adaptation using Semantic Web Technologies

Davy Van Deursen · Wim Van Lancker · Sarah De Bruyne · Wesley De Neve · Erik Mannens · Rik Van de Walle

Received: date / Accepted: date

Abstract Adaptation of media resources is an emerging field due to the growing amount of multimedia content on the one hand and an increasing diversity in usage environments on the other hand. Furthermore, to deal with a plethora of coding and metadata formats, format-independent adaptation systems are important. In this paper, we present a new format-independent adaptation system. The proposed adaptation system relies on a model that takes into account the structural metadata, semantic metadata, and scalability information of media bitstreams. The model is implemented using the Web Ontology Language (OWL). Existing coding formats are mapped to the structural part of the model, while existing metadata standards can be linked to the semantic part of the model. Our new adaptation technique, which is called RDF-driven content adaptation, is based on executing SPARQL Protocol And RDF Query Language (SPARQL) queries over instances of the model for media bitstreams. Using different criteria, RDF-driven content adaptation is compared to other adaptation techniques. Next to real-time execution times, RDF-driven content adaptation provides a high abstraction level for the definition of adaptations and allows a seamless integration with existing semantic metadata standards.

D. Van Deursen · W. Van Lancker · S. De Bruyne · W. De Neve · E. Mannens · R. Van de Walle
Multimedia Lab, Departement of Electronics and Information Systems
Ghent University – IBBT, Belgium
Gaston Crommenlaan 8, bus 201
B-9050 Ledeberg-Ghent, Belgium
Tel.: +32-9-33-14893
Fax: +32-9-33-14896
E-mail: {davy.vandeursen, wim.vanlancker, sarah.debruyne, erik.mannens, rik.vandewalle}@ugent.be, wmdeneve@icu.ac.kr

Keywords Format-independency · Media content adaptation · Multimedia model · Semantic Web technologies

1 Introduction

The efficient delivery of multimedia content in today's world of ubiquitous multimedia consumption is an important technological challenge, due to the growth in multimedia content on the one hand and an increasing diversity in usage environments on the other hand (e.g., in terms of device characteristics or network technologies). Therefore, the delivery of multimedia content needs to occur in a transparent way in order to obtain Universal Multimedia Access (UMA, [32]).

Multimedia content customization is an emerging field due to the UMA paradigm. A multimedia content customization system tries to meet the user needs by customizing the content based on the usage environment and user preferences. To take into account the restrictions of the usage environment, a wide variety of multimedia customization approaches are available, as described by Magalhães *et al.* in [18]. Basically, two major approaches exist to perform multimedia content customization:

– *Content selection*: corresponds to the identification of the most adequate media bitstream from those available to be sent to the end-user. Examples of multimedia content selection technologies are RealNetworks' SureStream, Multiple Bit Rate (MBR) profile used in the Windows Media Series, and the alternative track selection mechanism used in Quicktime. Further, the Synchronized Multimedia Integration Language (SMIL, [3]) provides support for

content selection based on the properties of the usage environment (e.g., language, device characteristics, or available bandwidth).
– *Content adaptation*: tries to meet the constraints of a usage environment by performing structural adaptations (i.e., creating tailored versions of the content) and semantic adaptations (i.e., extracting specific fragments that are of interest to the user). Three approaches can be distinguished:
  – *Transcoding*: customizes multimedia content by performing one ore more operations on the compressed media bitstream, typically using signal-processing techniques;
  – *Scalable coding*: enables the extraction of multiple (lower quality) versions of the same media resource without the need of a complete recoding process (examples of scalable coding formats are H.264/AVC SVC and JPEG2000) [22];
  – *Transmoding*: performs a modality transformation, for instance when the usage environment conditions do not allow to consume the multimedia content with its original media types.

In this paper, we introduce a novel media resource customization method which primarily relies on scalable coding formats, but which also provides support for content selection. Advantages of scalable coding are the higher coding efficiency and a higher flexibility in terms of the number of possible tailored versions of a media resource. Advantages of content selection are the lower complexity to create media resources and the possibility to switch between coding formats or even between media types (e.g., video or slideshow). Although both scalable coding and content selection have their pro's and contra's, they can be used as complementary media customization techniques. More specifically, content selection could be applied first to obtain a version of a media resource that is roughly suited for the usage environment. Next, scalability layers of the selected media resource could be removed and semantic adaptations could be applied so that the resulting tailored media resource is better suited for the targeted usage environment.

Over the last few years, the number of multimedia coding standards has grown significantly. In order to deal with current and future multimedia coding formats, format-independent adaptation systems are important. In this paper, we present a new method for the adaptation of media resources in a format-independent way, called RDF-driven content adaptation. It is inspired by the principles of XML-driven content adaptation techniques [1], while its final design is based on a model describing structural, semantic, and scalability information of media bitstreams. Existing coding and metadata formats are mapped to this model in order to be supported. Hence, adaptation operations (e.g., frame rate scaling or scene selection) can be expressed and implemented based on this model, in a way that is independent of the underlying coding format. Instead of using XML, Semantic Web technologies such as the Resource Description Framework (RDF, [16]), the Web Ontology Language (OWL, [19]), and the SPARQL Protocol And RDF Query Language (SPARQL, [24]) are used to implement our adaptation technique. This way, the interoperability between different metadata standards can be enhanced thanks to a more natural representation of objects and relationships, as reported by the W3C Multimedia Semantics Incubator Group (MMSem, [33]).

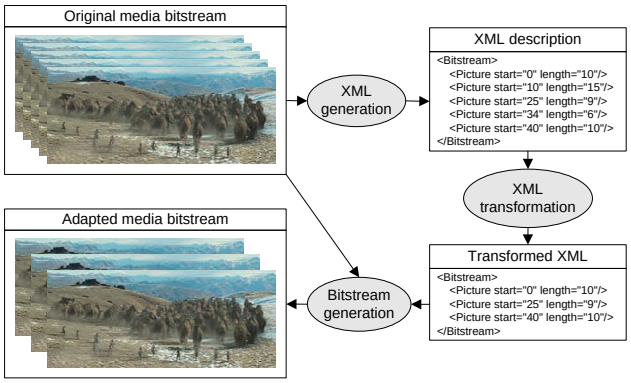

Fig. 1 Performing video frame rate reduction using XML-driven content adaptation.

2 XML-driven Content Adaptation: Concepts and Issues

In this section, the basic principles of XML-driven content adaptation are explained, together with its main weaknesses.

2.1 XML-driven Content Adaptation

A format-agnostic adaptation framework can be realized by relying on automatically generated XML descriptions, called Bitstream Syntax Descriptions (BSDs). These descriptions contain information about the high-level structure of a media bitstream. The actual adaptation occurs in the XML domain, rather than in the binary domain.

2.1.1 General Functioning

As shown in Fig. 1, an XML-driven content adaptation framework typically consists of three main processes:

the generation of an XML description, the transformation of the XML description, and the creation of an adapted bitstream using the transformed XML description. Such an XML description contains information about the high-level structure of a compressed bitstream and is used to steer the adaptation process. In particular, it describes how the bitstream is organized in terms of layers or packets of data. Note that such an XML description is not meant to replace the original binary data; it rather acts as an additional layer on top of the binary data, similar to metadata. Hence, in the remainder of this paper, information regarding the high-level structure of a compressed bitstream is referred to as *structural metadata*.

The actual adaptation takes place in the XML domain during the transformation of the XML description (e.g., by dropping descriptions of layers or packets). This transformation process takes into account the constraints of a given usage environment (e.g., available bandwidth and screen resolution). Thanks to the use of XML, many already existing XML transformation tools can be used, such as eXtensible Stylesheet Language Transformations (XSLT, [15]) or Streaming Transformations for XML (STX, [5]).

The translation from the XML domain back to the binary domain occurs in the last step, i.e., the creation of the adapted bitstream. This process takes as input the transformed XML description and the original bitstream in order to produce an adapted bitstream, which is then suited for consumption in a given usage environment.

Using high-level XML descriptions for adapting multimedia content enables the use of format-agnostic software modules within an adaptation framework. Hence, the software supports future coding formats without having to be rewritten, is suited for hardware implementations, and can be used for the adaptation of still images, audio, and video bitstreams.

2.1.2 Existing Technologies

In recent years, a number of XML-driven content adaptation technologies have been developed.

- The MPEG-21 Multimedia Framework [4] aims at realizing the 'big picture' in the multimedia production, delivery, and consumption chain. One important part of the framework, Digital Item Adaptation (DIA, [14]), provides several tools that can be used for creating an interoperable and XML-driven adaptation framework: the Bitstream Syntax Description Language (BSDL) and the generic Bitstream Syntax Schema (gBS Schema, [29]).
- The Formal Language for Audio-Visual Object Representation, extended with XML features (XFlavor, [12]) is a declarative Java-like language. This tool provides means for describing the syntax of a bitstream on a bit-per-bit basis. It enables the automatic generation of a parser that is able to generate an XML description for a given bitstream of a specific format.

2.1.3 Target Applications

The main application for XML-driven adaptation is the exploitation of scalability in media bitstreams. As discussed in Sect. 1, scalable coding is an important tool to realize a UMA environment. It enables the extraction of multiple (lower quality) versions of the same media resource without the need of a complete recoding process. The bitstream extraction process typically involves the removal of particular data blocks and the modification of the value of certain syntax elements. Such operations can easily be executed using XML-driven adaptation.

Next to the exploitation of scalability, a number of other applications can also make use of descriptions of the high-level structure of a media bitstream.

- Adaptations based on semantic information about the multimedia content: examples are the removal of violent scenes or the selection of specific segments that are of interest to the user. Hence, applications such as video summarization, video skimming, and scene selection are possible with XML-driven adaptation.
- Format-independent delivery of multimedia content: streaming servers do no longer require additional software modules for packetization purposes in order to support new coding formats [27].

2.2 Problems with XML-based Format-independent Adaptation Systems

Current content adaptation systems based on XML descriptions of the high-level structure of media bitstreams enable the use of format-independent engines. However, a number of issues can still be identified:

- creators of XML filters have to implement low-level, coding-format dependent algorithms that are needed to obtain format-independent adaptation operations;
- the integration of semantic adaptation operations and semantic metadata standards is realized in an ad-hoc way.

We will illustrate these problems by means of simple examples. Suppose we have video content encoded by

making use of MPEG-2 Video and H.264/AVC. A BSD is created using BSDL for both sequences, as shown in Fig. 2. Now suppose that we want to express the adaptation operation 'remove the highest temporal layer', which will result in a decrease of the frame rate. In Fig. 2, workflow diagrams are shown that correspond to XML filters expressing the adaptation operation. Note that for the H.264/AVC example, a simplified algorithm is given; more information on the exploitation of temporal scalability in H.264/AVC can be found in [8].

The actual adaptation engines in an XML-driven content adaptation system are format-independent, while the descriptions themselves are format-specific. Hence, an XML filter implementing a particular adaptation operation (e.g., removal of temporal layers in a scalable video stream) needs to know the structure of these descriptions, and is thus dependent on the coding format. Therefore, multiple format-dependent XML filters need to be created to implement the same adaptation operation for different formats, as illustrated in Fig. 2. The functionality of these XML filters is the same: calculate the temporal level for each video frame and decide whether the frame should be dropped or not. However, the calculation of the temporal levels is format-specific and should therefore occur in the BSD generation step. This allows making the XML filter independent of the coding format and also avoids having to execute the same calculations for each adaptation request.

The same observation can be made for semantic adaptation operations, which typically express a selection of media fragments based on semantic metadata (e.g., selection of sport fragments in a news sequence). Semantic metadata are often linked to the multimedia content by means of timestamps, whereas structural metadata use bit offsets. Therefore, a mapping between the timestamp values of the semantic metadata and the bit offsets of the structural metadata needs to be implemented in the XML filter. Such a mapping is dependent on the underlying coding format and should be avoided during the BSD transformation step.

Hence, despite the use of an abstraction layer (i.e., a BSD) within XML-driven content adaptation, the adaptation operations themselves are not abstracted. More specifically, compared to format-specific adaptation software, the format-dependency is shifted from the software to the XML filters. Creators of these XML filters cannot think in terms of high-level and format-independent adaptations but have to be aware of the underlying coding formats. Hence, with the current XML-driven approach, format-independency is obtained in an ad-hoc manner.

The second problem regarding the integration of semantic adaptation operations and semantic metadata

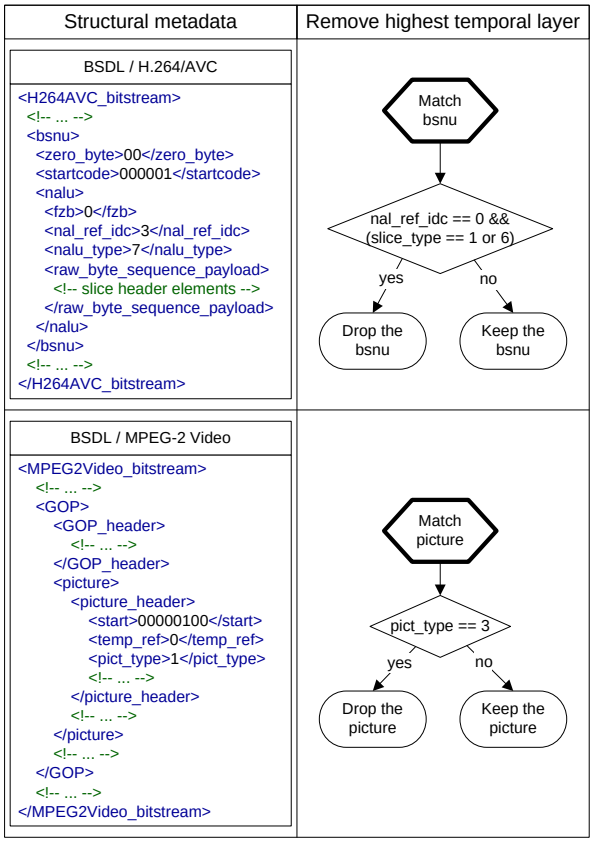

Fig. 2 Implementing temporal scalability using XML-driven content adaptation.

standards is related to the interoperability issues of XML [33,9]. Metadata formats typically consist of an XML Schema accompanied with plain text (e.g., MPEG-7). The text usually describes the semantic meaning of the XML tags specified in the XML Schema. The role of XML Schema is to define a grammar for XML documents. However, when it comes to semantic interoperability and making data understandable, XML has disadvantages. XML's major limitation is that it only describes grammars. It is impossible to recognize a semantic unit from a particular domain because XML aims at document structure and imposes no common interpretation of the data contained in the document [9]. For instance, taking into account different metadata standards, the same tags can have a different meaning while tags with the same meaning can occur in different structures. Hence, expressing the semantic adaptation operation 'select sport fragments' in a scenario where two different semantic metadata standards are used (e.g., annotations are provided in both MPEG-7 and TV-Anytime), requires the development of two different XML filters (i.e., one that can interpret MPEG-7 and one that can interpret TV-Anytime). Note that the previous observation also holds true for different struc-

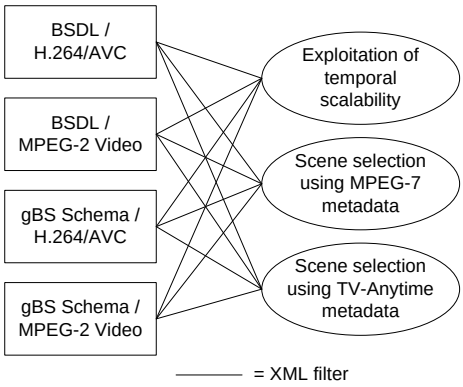

Fig. 3 Problems with XML-driven content adaptation.

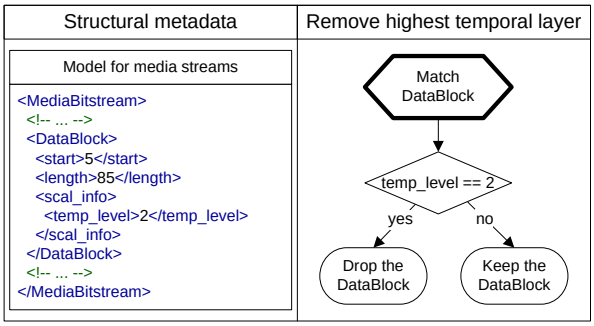

Fig. 4 Expressing adaptation operations on top of format-independent BSDs.

tural metadata standards. Examples of standardized solutions for expressing structural metadata are BSDL and gBS Schema; however, other representations such as XFlavor are also possible. It is obvious that XML filters are also dependent on the technology used for the representation of structural metadata.

Both problems are also illustrated in Fig. 3, where video content is encoded by making use of MPEG-2 Video and H.264/AVC. Furthermore, semantic metadata regarding the different scenes in the video is available in both MPEG-7 and TV-Anytime. Finally, metadata regarding the high-level structure of the encoded video bitstreams is provided by descriptions compliant with both BSDL and gBS Schema. As can be seen in this figure, XML filters performing the XML transformation step (discussed in Sect. 2.1.1) are depending on the coding format (i.e., H.264/AVC and MPEG-2 Video), the metadata format of the structural metadata (i.e., BSDL and gBS Schema), and the metadata format of the semantic metadata (i.e., MPEG-7 and TV-Anytime).

Next to the two problems mentioned above, there is also the problem of XML verbosity. XML descriptions of the high-level structure of media bitstreams tend to become verbose [10]. It is possible to efficiently compress the XML descriptions, but no generic solution exists to perform the transformation of the XML description in the compressed domain, despite a number of efforts being made in the past [28,31].

3 Modeling Media Bitstreams

In order to solve the problems discussed in Sect. 2.2 (i.e., format-specific XML filters and an ad-hoc integration of semantic adaptation operations and semantic metadata standards), we propose the following solution. As mentioned in Sect. 2.2, XML filters are dependent on the underlying coding format because the descriptions of the high-level structure of media bitstreams are format dependent. As such, despite the fact that the underlying adaptation engines are independent of the coding format, the actual transformation logic is not.

Therefore, the transformations of the structural metadata should be shifted to a higher level. This can only be realized if format-specific calculations (e.g., calculation of temporal levels or calculation of a timestamp for a particular frame) can be avoided during the BSD transformation step. Hence, we propose to shift these format-specific calculations from the BSD transformation step to the BSD generation step. As a result, we are able to obtain a fully format-independent BSD that enables the expression of high-level adaptation operations (an example can be found in Fig. 4). We have defined the structure and semantics of such a fully format-independent BSD in the form of a model for media bitstreams. That way, the adaptation operations can be formulated in terms of transformations based on the model; hence they are shifted to a higher level, independent of the coding format. Our model for media bitstreams is based on previous work regarding the definition and implementation of a model for media bitstreams [17,21,30]. However, the latter only provides support for high-level structural adaptations and uses XML as underlying technology.

To obtain a seamless integration between semantic adaptation operations and semantic metadata standards, our model for media bitstreams needs to support semantic adaptation operations. More specifically, it has to enable the extraction of specific media fragments from a media bitstream, independent of the coding format. Furthermore, the model has to provide hooks to connect to existing semantic metadata standards. That way, semantic adaptation operations can be expressed by using concepts defined in existing semantic metadata standards.

We have avoided XML Schema to describe our model because the use of XML as underlying technology causes interoperability problems between different metadata

standards, as discussed in Sect. 2.2. In contrast to XML, Semantic Web technologies such as RDF and OWL enhance the interoperability among metadata standards for multimedia content [33]. Therefore, our model for media bitstreams is implemented by using OWL. The instances of the model (i.e., the structural metadata or BSDs) are expressed in RDF. The transformation of the structural metadata is implemented by using SPARQL queries, which are independent of the coding format.

3.1 Model for Media Bitstreams

As elaborated on above, a model for media bitstreams is needed in order to abstract the transformation of the structural metadata. In this section, we present such a model covering structural, semantic, and scalability information. As discussed above, we implemented the model by using OWL. In Fig. 5, an overview is given of the model. More detailed views on this model are provided in the next subsections. Note that for figures visualizing (parts of) the multimedia model, ellipses, rectangles, and arrows represent OWL classes, literals, and properties respectively.

3.1.1 Structural Metadata

The modeling of the high-level structure of a media bitstream is shown in Fig. 6. The *MediaBitstream* class corresponds to a particular compressed, elementary media bitstream. It contains a *name* and a description of the underlying codec (e.g., H.264/AVC), i.e., the *codec* property. Furthermore, it contains a reference to the location of the media bitstream by means of the *bitstreamSource* property.

A *MediaBitstream* consists of a number of *RandomAccessUnit*s. Random access refers to the ability of the decoder to start decoding at a point in a compressed media bitstream other than at the beginning and to recover an exact representation of the decoded bitstream [6, 11]. A *RandomAccessUnit* contains a number of successive *DataBlock*s, pointing to particular segments in the media bitstream. More detailed information regarding the modeling of *DataBlock*s is provided in Sect. 3.1.3.

The property *hasStructure*, which is used to connect the classes in the structural metadata, is a transitive property. This means that if a pair (x, y) is an instance of *hasStructure*, and the pair (y, z) is also an instance of *hasStructure*, then we can infer that the pair (x, z) is also an instance of *hasStructure*. Additionally, it is possible that a *MediaBitstream* has *DataBlock*s not occurring in *RandomAccessUnit*s (e.g., a Sequence Parameter Set (SPS) in H.264/AVC).

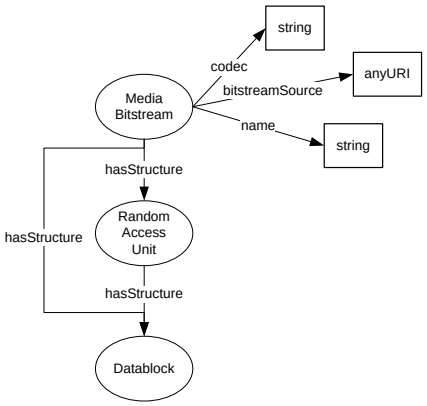

Fig. 6 Model for the structural metadata.

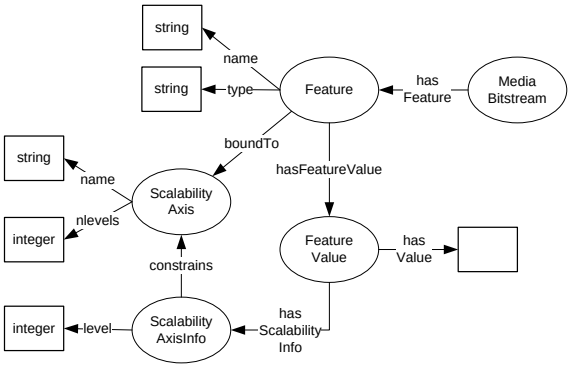

Fig. 7 Model for the scalability information.

3.1.2 Scalability Information

The scalability information part of the model provides information regarding properties of the media bitstream that are related to possible adaptation operations. It enables the declaration of what types of adaptations may or should be applied to the media bitstream in order to optimally fit a given context [20]. The modeling of the scalability information is depicted in Fig. 7. A *MediaBitstream* contains a number of *Features*, each containing a name and a type. An example of a feature is 'frame rate', having as type 'fps'. Each *Feature* contains one ore more *FeatureValue*s. Futhermore, a *Feature* can be bound to one or more *ScalabilityAxes* containing a name and an amount of levels (e.g., a temporal scalability axis containing four levels). A *FeatureValue* can be linked to a *ScalabilityAxisInfo* class, which provides information regarding the relationship between the scalability axes and the feature values. This is established by constraining a *ScalabilityAxis* by means of a level (e.g., 15 fps corresponds to the second level of the temporal scalability axis).

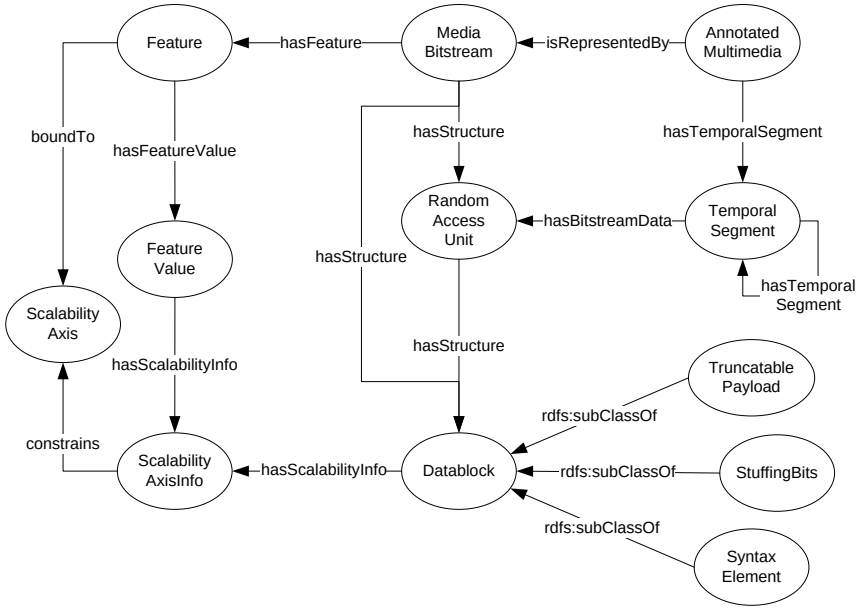

Fig. 5 Overview of the multimedia model.

3.1.3 Data Blocks

As already mentioned above, *DataBlock*s point to particular segments in the media bitstream. In Fig. 8, the modeling of a data block is shown in detail. A *DataBlock* is always characterized by two properties: the *start* and *length* of the datablock in terms of bits. Additionally, a datablock can contain *ScalabilityAxisInfo* (see Sect. 3.1.2), indicating to which scalability layers the data block belongs.

Three possible subclasses exist for a *DataBlock*.

- *TruncatablePayload*: points to a bitstream segment that can be truncated by a number of bytes, something that typically occurs within bitstreams encoded with Fine Granularity Scalability (FGS) techniques.
- *StuffingBits*: allow to write padding bits to the output bitstream until it is byte-aligned. The property *nbytes* determines on how many bytes the bitstream is aligned. E.g., if *nbytes* is equal to four, then padding bits are added until the output bitstream is aligned on four bytes.
- *SyntaxElement*: represents a specific syntax element of the media bitstream. The *value* property indicates the value of the syntax element, i.e., the decimal representation of the bits covered by the syntax element (specified by the *start* and *length* property).

3.1.4 Semantic Metadata

A *MediaBitstream* can be annotated by means of *AnnotatedMultimedia*, which contains a semantic description

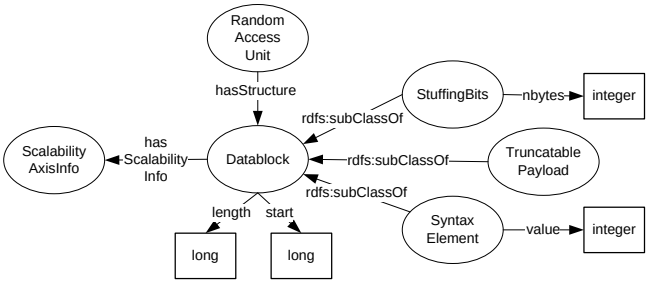

Fig. 8 Model for a data block.

of the content of the *MediaBitstream*. The semantic metadata model is depicted in Fig. 9. *AnnotatedMultimedia* consists of a number of *TemporalSegment*s, each pointing to a specific segment of the multimedia content by means of a *start* and *duration* property. A *TemporalSegment* can in its turn contain a number of other *TemporalSegment*s, allowing to model a hierarchy of *TemporalSegment*s. Furthermore, a *TemporalSegment* has a *keyword* property, allowing a simple annotation of the multimedia content by means of keywords (i.e., a form of tagging). More complicated semantic descriptions of *TemporalSegments* can be obtained by linking existing ontologies to our semantic metadata model. More information regarding the linking of other ontologies to our multimedia model is provided in Sect. 3.2.2.

The connection between the media bitstream and the *TemporalSegment*s is established by means of the *hasBitstreamData* property. Each *RandomAccessUnit* of a media bitstream can belong to one or more *TemporalSegment*s. This way, the *TemporalSegment*s are connected to the bits of particular media bitstreams. Note

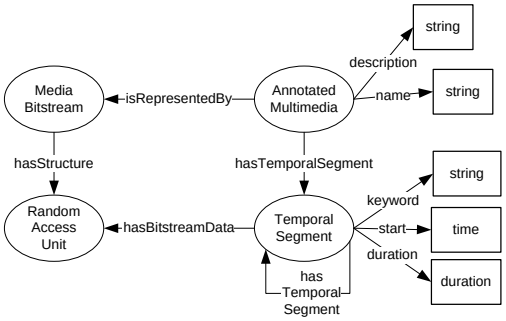

Fig. 9 Model for the semantic metadata.

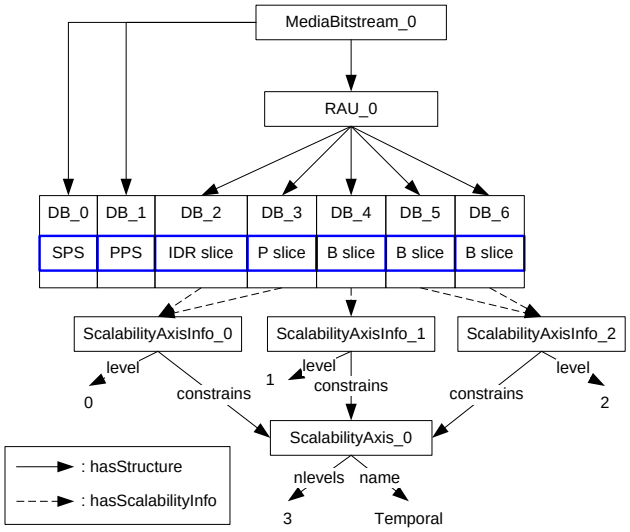

Fig. 10 Describing an H.264/AVC bitstream with the multimedia model.

that *AnnotatedMultimedia* points to the multimedia content by means of timestamps, independently of the coding format, while a *MediaBitstream* points to specific bitstream segments in terms of bits.

Furthermore, since different *MediaBitstream*s can be linked to one *AnnotatedMultimedia*, the model for media bitstreams provides support for multimedia content selection (see Sect. 1). Indeed, it is possible to relate multiple *MediaBitstream* instances (each corresponding to a different version in terms of coding format, bit rate, etc.) to one *AnnotatedMultimedia* instance.

3.2 The Multimedia Model in Practice

In this section, the relationship between the multimedia model presented in Sect. 3.1 and existing coding and metadata formats is illustrated.

3.2.1 Mapping H.264/AVC to the Multimedia Model

Existing multimedia coding formats need to be mapped to the model defined in Sect. 3.1. As an example, we

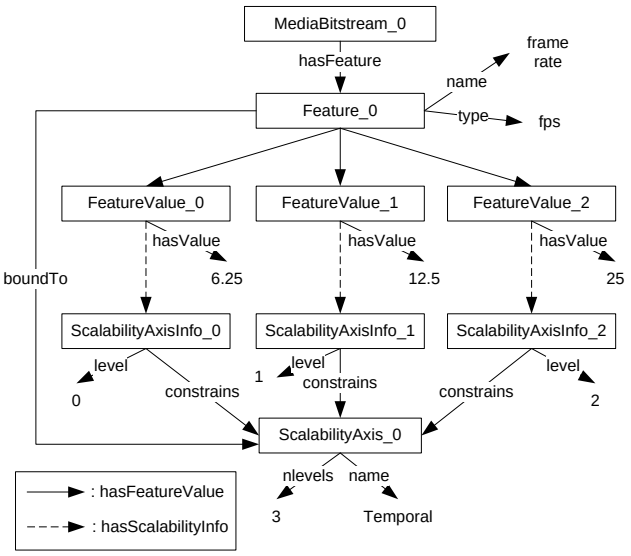

Fig. 11 Describing scalability information of an H.264/AVC bitstream with the multimedia model.

map the H.264/AVC coding format to the model. Fig. 10 provides a visual representation of the mapping of an H.264/AVC encoded bitstream to the multimedia model. Note that an excerpt of the resulting RDF triples is shown in Fig. 12.

An H.264/AVC encoded bitstream is a succession of Network Abstraction Layer Units (NALUs). Different NALU types exist: a Sequence Parameter Set (SPS), which contains information related to the whole sequence; a Picture Parameter Set (PPS), which contains information related to a set of pictures; and a slice layer, which contains the actual encoded data such as the motion vectors and the residual data.

As shown in Fig. 10, when mapping an H.264/AVC encoded bitstream to the multimedia model, each NALU is mapped to a *DataBlock*. The SPS and PPS *DataBlock*s are directly connected to the *MediaBitstream*. Furthermore, the Instantaneous Decoding Refresh (IDR) slices indicate the start of a new *RandomAccessUnit*.

Slice *DataBlock*s are provided with *ScalabilityInfo* indicating in which scalability layer the data block is located. In this case, only one *ScalabilityAxis* is present, i.e., the temporal scalability axis containing three levels. Since the example H.264/AVC bitstream is encoded using hierarchical B-pictures, the first B-picture is located in the second temporal layer, while the other two B-pictures are located in the third temporal layer. I- and P-pictures are located in the first temporal layer [8].

Fig. 11 illustrates the description of the features of the bitstream. For instance, a feature in this example is 'frame rate' with possible *FeatureValue*s 6.25, 12.5, and

```
<rdf:RDF xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:mmm="http://www.foo.be/multimedia_model.owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  >
  <mmm:MediaBitstream rdf:about="http://www.foo.be/soccer.rdf#mb">
    <mmm:bitstreamSource rdf:resource="http://www.foo.be/soccer.264"/>
    <mmm:hasFeature rdf:resource="http://www.foo.be/soccer.rdf#feat_frame-rate"/>
    <mmm:codec>H.264/AVC</mmm:codec>
    <mmm:name>soccer_avc</mmm:name>
    <mmm:codecDescription>H.264/AVC</mmm:codecDescription>
    <mmm:hasStructure rdf:resource="http://www.foo.be/soccer.rdf#db_0"/>
    <mmm:hasStructure rdf:resource="http://www.foo.be/soccer.rdf#db_1"/>
    <mmm:hasStructure rdf:resource="http://www.foo.be/soccer.rdf#rau_0"/>
    <!-- ... other RAUs ... -->
  </mmm:MediaBitstream>
  <mmm:RandomAccessUnit rdf:about="http://www.foo.be/soccer.rdf#rau_0">
    <mmm:hasStructure rdf:resource="http://www.foo.be/soccer.rdf#db_2"/>
    <mmm:hasStructure rdf:resource="http://www.foo.be/soccer.rdf#db_3"/>
    <!-- ... other DBs ... -->
  </mmm:RandomAccessUnit>
  <mmm:DataBlock rdf:about="http://www.foo.be/soccer.rdf#db_2">
    <mmm:start>3880</mmm:start>
    <mmm:length>4800</mmm:length>
    <mmm:hasScalabilityInfo rdf:resource="http://www.foo.be/soccer.rdf#si_temp_0"/>
  </mmm:DataBlock>
  <mmm:ScalabilityAxis rdf:about="http://www.foo.be/soccer.rdf#sa_temp">
    <mmm:name>temporal</mmm:name>
    <mmm:nlevels>3</mmm:nlevels>
  </mmm:ScalabilityAxis>
  <mmm:ScalabilityAxisInfo rdf:about="http://www.foo.be/soccer.rdf#si_temp_0">
    <mmm:constrains rdf:resource="http://www.foo.be/soccer.rdf#sa_temp"/>
    <mmm:level>0</mmm:level>
  </mmm:ScalabilityAxisInfo>
  <mmm:Feature rdf:about="http://www.foo.be/soccer.rdf#feat_frame-rate">
    <mmm:boundTo rdf:resource="http://www.foo.be/soccer.rdf#sa_temp"/>
    <mmm:hasFeatureValue rdf:resource="http://www.foo.be/soccer.rdf#featvalue_0"/>
    <!-- ... other featureValues ... -->
    <mmm:name>frame rate</mmm:name>
    <mmm:type>fps</mmm:type>
  </mmm:Feature>
  <mmm:FeatureValue rdf:about="http://www.foo.be/soccer.rdf#featvalue_0">
    <mmm:hasScalabilityInfo rdf:resource="http://www.foo.be/soccer.rdf#si_temp_0"/>
    <mmm:value>6.25</mmm:value>
  </mmm:FeatureValue>
  <mmm:AnnotatedMultimedia rdf:about="http://www.foo.be/soccer.rdf#soccer_annotation">
    <mmm:description>Soccer match</mmm:description>
    <mmm:isRepresentedBy rdf:resource="http://www.foo.be/soccer.rdf#mb"/>
    <mmm:hasTemporalSegment rdf:resource="http://www.foo.be/soccer.rdf#ts_0"/>
    <mmm:hasTemporalSegment rdf:resource="http://www.foo.be/soccer.rdf#ts_1"/>
    <!-- ... other temporal segments ... -->
  </mmm:AnnotatedMultimedia>
  <mmm:TemporalSegment rdf:about="http://www.foo.be/soccer.rdf#ts_0">
    <mmm:segmentStart>T00:00:00:00F25</mmm:segmentStart>
    <mmm:segmentDuration>PT10S0N25F</mmm:segmentDuration>
    <mmm:keyword>yellow card</mmm:keyword>
    <mmm:hasBitstreamData rdf:resource="http://www.foo.be/soccer.rdf#rau_0"/>
    <mmm:hasBitstreamData rdf:resource="http://www.foo.be/soccer.rdf#rau_1"/>
    <!-- ... other RAUs -->
  </mmm:TemporalSegment>
</rdf:RDF>
```

Fig. 12 Excerpt of RDF triples describing an H.264/AVC bitstream (RDF/XML notation).

25 fps. These values correspond to the first, second, and third temporal layer respectively.

3.2.2 Linking the Semantic Multimedia Model to Existing Ontologies

As discussed in Sect. 3.1.4, the multimedia model allows to create simple annotations of the multimedia content by adding keywords to *AnnotatedMultimedia* or *TemporalSegment* instances. However, more sophisticated semantic descriptions are often needed. Therefore, the semantic part of the multimedia model provides hooks for existing ontologies. An example is shown in Fig. 13, where a class *Person*, defined in the Friend-Of-A-Friend (FOAF[1]) ontology, is connected with *TemporalSegments* and *AnnotatedMultimedia*. A second example is shown in Fig. 14, where concepts described in a soccer ontol-

[1] `http://xmlns.com/foaf/spec/`

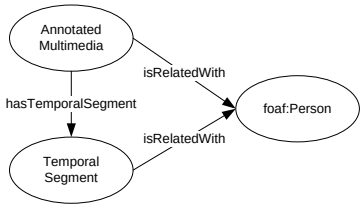

Fig. 13 Linking domain-specific ontologies with the semantic multimedia model.

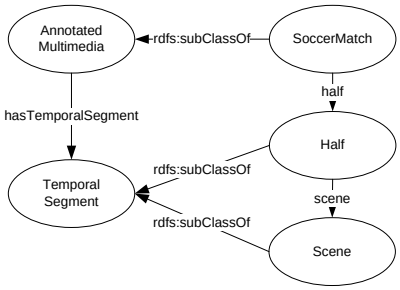

Fig. 14 Extending the semantic multimedia model with domain-specific ontologies.

ogy are subclasses of *AnnotatedMultimedia* and *TemporalSegment*. This way, media streams that are annotated as illustrated above can be adapted based on the concepts defined in the linked ontologies. For example, fragments can be selected that are related with a specific person. The latter person corresponds to a URI that is an instance of the class *Person* defined in the FOAF ontology.

The semantic multimedia model discussed in Sect. 3.1.4 can also be linked to existing ontologies specifying how to connect semantic multimedia descriptions to parts of a media asset. Examples are MPEG-7 [13], DIG35[2], Exif[3] and the Core Ontology for MultiMedia (COMM, [2]).

4 RDF-driven Content Adaptation

In order to cope with the problems of XML-driven content adaptation mentioned in Sect. 2.2, we present a new technique for multimedia content adaptation in a format-independent way, i.e., RDF-driven content adaptation. On the one hand, Semantic Web technologies are used to represent the metadata for multimedia content. On the other hand, a model for media bitstreams covering structural, semantic, and scalability information (defined in Sect. 3) is used to abstract the transformation process.

[2] `http://www.w3.org/2005/Incubator/mmsem/XGR-vocabularies/#formal-DIG35`

[3] `http://www.w3.org/2005/Incubator/mmsem/XGR-vocabularies/#formal-Exif`

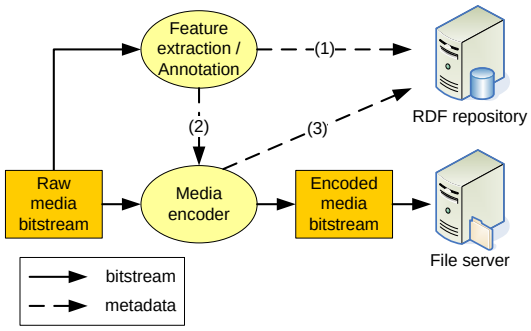

Fig. 15 Example architecture for the generation of metadata compatible with the multimedia model.

4.1 Metadata Generation

Several possibilities exist to generate metadata compatible with the multimedia model defined in Sect. 3.1. Fig. 15 shows an example of an architecture which takes as input raw media bitstreams. The first step is to extract features (e.g., shot segmentation) and/or to manually add annotations regarding the multimedia content. The resulting semantic metadata will consist of instances of *AnnotatedMultimedia* and *TemporalSegment*s and is stored in an RDF repository (arrow (1) in Fig. 15). Next, the raw media bitstream is encoded. The encoded media bitstream is send to a file server. During the encoding process, the structural metadata is generated (arrow (3) in Fig. 15). More specifically, a *MediaBitstream* instance is created accompanied by instances of *RandomAccessUnit*s, *DataBlock*s, *Feature*s, etc. The semantic metadata, obtained during the feature extraction and/or annotation, is used by the multimedia encoder (arrow (2) in Fig. 15) in order to connect the structural metadata with the semantic metadata (i.e., the bits of the encoded bitstream are linked to the timestamps available in the semantic metadata). As discussed in Sect. 3.1.4, this is realized by linking *RandomAccessUnit*s to *TemporalSegment*s by using the *hasBitstreamData* property. Finally, the structural metadata is also stored in an RDF repository.

It is important to remark that the scenario described above is not applicable for already encoded media bitstreams, possibly described by a specific metadata format (e.g., H.264/AVC encoded bitstreams annotated with MPEG-7 metadata and described by BSDL). In this case, software is needed to translate the available metadata into metadata compliant with the multimedia model. In case no structural metadata is present, coding-format specific parsers have to be created, taking as input an encoded media bitstream and producing structural metadata compliant with the multimedia model. Note that such coding-format specific parsers could be automatically generated by using a tech-

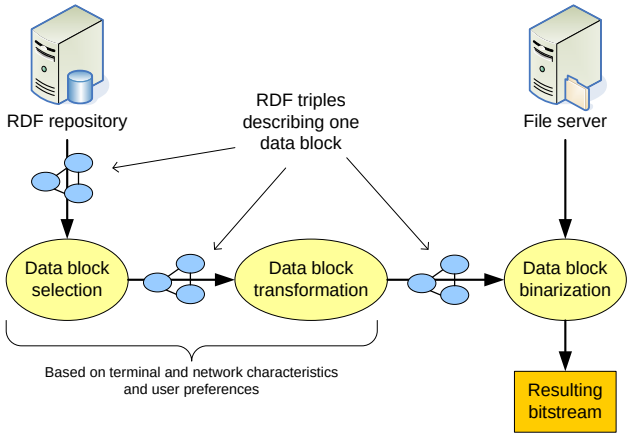

Fig. 16 The general workflow of RDF-driven content adaptation.

nique similar to XFlavor [12], which is able to automatically generate a coding-format specific parser producing XML descriptions of the high-level structure of the media bitstreams. The same approach could be followed in the context of RDF-driven content adaptation. More specifically, instead of producing XML descriptions, such parsers could generate RDF triples compliant with the multimedia model.

4.2 General Workflow

Having generated all the necessary metadata, it is now possible to retrieve and adapt a specific media bitstream. The general workflow is depicted in Fig. 16. There are three main steps during RDF-driven content adaptation: data block selection, transformation, and binarization. RDF graphs describing data blocks are queried during the data block selection step. These RDF graphs can be adapted during the data block transformation step. Finally, each selected and adapted RDF graph is used to generate the resulting media bitstream. More detailed information regarding the different steps in the workflow is provided in the next subsections.

A comparison of the workflow of XML-driven content adaptation (discussed in Sect. 2.1.1) and RDF-driven content adaptation can be made as follows. The generation of an XML description, the transformation of the XML description, and the creation of an adapted bitstream using the transformed XML description correspond to RDF metadata generation, data block selection and transformation, and data block binarization respectively.

4.2.1 Data Block Selection

Selecting data blocks is done by taking into account terminal and network characteristics together with user preferences. The preferred data blocks can be obtained by performing the following sequence of steps.

(1) Semantic selection: an instance of *AnnotatedMultimedia* is selected based on the user preferences (e.g., select a soccer match of a specific team). Furthermore, the semantic selection can be refined by selecting only specific fragments that are desired by the user (e.g., only select fragments where a specific soccer player occurs). Moreover, fragments from different instances of *AnnotatedMultimedia* can be selected. The result of the semantic selection is a list of *TemporalSegment*s corresponding with the fragments selected based on the usage environment.
(2) Bitstream selection: given an instance of *AnnotatedMultimedia*, a *MediaBitstream* representing this multimedia content is selected. Note that this selection can be based on the usage environment (e.g., the device of the end-user only supports a limited amount of coding formats).
(3) Semantic to structural mapping: the selected *TemporalSegment*s are used to obtain the *DataBlock*s contained in the selected *MediaBitstream*. This is possible by following the link between the *TemporalSegment*s and the *RandomAccessUnit*s.
(4) Structural selection: the subset of *DataBlock*s obtained by the semantic selection is further restricted by selecting only the *DataBlocks* occurring in specific scalability layers. For instance, if a video stream with a frame rate of 15 fps is requested, only the *DataBlocks* having *ScalabilityInfo* meeting this condition are selected.

Note that it is not necessary to execute each step every time. One could for instance avoid the structural selection if there is no scalability information present.

The selection of data blocks can be performed by using the SPARQL Protocol and RDF Query Language (SPARQL, [24]). This is a query language and data access protocol for the Semantic Web, standardized by the RDF Data Access Working Group (DAWG) of the World Wide Web Consortium (W3C). SPARQL offers querying based on triple patterns, conjunctions, disjunctions, and optional patterns; results of SPARQL queries can be results sets or RDF graphs.

In order to demonstrate the data block selection process, a number of SPARQL queries are discussed. Since we want to obtain RDF graphs describing data blocks, we need SPARQL CONSTRUCT queries, i.e., queries resulting in RDF graphs. Listing 1 shows a query which selects data blocks occurring in the temporal base

Listing 1 Format-independent SPARQL query which selects data blocks occurring in the temporal base layer.

```
1  PREFIX mmo: <multimedia_model.owl#>
   CONSTRUCT {
     ?db rdf:type ?types.
     ?db mmo:start ?start.
5    ?db mmo:length ?length.
     ?db mmo:nBytes ?sb.
     ?db mmo:value ?value.
   }
   WHERE {
10   ?bitstream rdf:type mmo:MultimediaBitstream.
     ?bitstream mmo:name 'foo'.
     ?bitstream mmo:hasStructure ?db.
     ?db rdf:type ?types.
     ?db mmo:start ?start.
15   ?db mmo:length ?length.
     OPTIONAL {
       ?db mmo:syntaxElementValue ?value.
     }
     OPTIONAL {
20     ?db mmo:nStuffingBytes ?sb.
     }
     OPTIONAL {
       ?db mmo:hasScalabilityInfo ?si_temp.
       ?si_temp mmo:hasScalabilityAxis ?sa_temp.
25     ?sa_temp mmo:name 'temporal'.
       ?si_temp mmo:level ?temp_level.
     }
     FILTER(!bound(?temp_level) || ?temp_level = 0)
   }
```

Listing 2 Format-independent SPARQL query to obtain data blocks belonging to the first half of a specific soccer match.

```
1  PREFIX mmo: <multimedia_model.owl#>
   PREFIX so: <soccer.owl#>
   CONSTRUCT {
     ?db rdf:type ?types.
5    ?db mmo:start ?start.
     ?db mmo:length ?length.
     # ...
   }
   WHERE {
10   ?bitstream rdf:type mmo:MultimediaBitstream.
     ?bitstream mmo:name 'foo'.
     ?annoMM mmo:isRepresentedBy ?bitstream.
     ?annoMM rdf:type mmo:AnnotatedMultimedia.
     ?annoMM mmo:hasTemporalSegment ?segment.
15   ?segment rdf:type so:Half.
     ?segment so:number 1^^xsd:integer.
     ?segment mmo:hasBitstreamData ?rau.
     ?bitstream mmo:hasStructure ?rau.
     ?rau mmo:hasStructure ?db.
20   ?db rdf:type ?types.
     ?db mmo:start ?start.
     ?db mmo:length ?length.
     # ...
   }
```

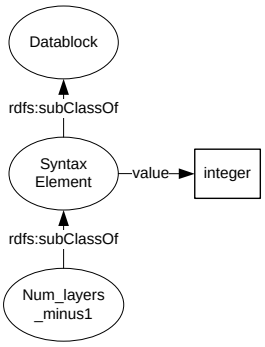

Fig. 17 Extending the multimedia model for the scalability information located in an SVC bitstream.

layer. Lines 2–8 contain the triples needed to describe one data block. The WHERE clause (lines 9–29) determines which data blocks are selected. In this example, no semantic information is used to select the *MediaBitstream*, i.e., the bitstream is selected based on its name (line 11). Lines 12-21 bind the variables defined in the CONSTRUCT clause. Lines 22–28 specify that only data blocks which occur in the temporal base layer or which do not have scalability information (e.g., SPS or PPS in an H.264/AVC bitstream) are selected.

An example of selecting datablocks based on semantic information is provided in Listing 2. In this example, data blocks belonging to the first half of a specific soccer match are selected. Lines 3–8 contain the triples needed to describe one data block, analogous to the previous example. In the WHERE clause, the *TemporalSegment* is selected which corresponds to the first half of the soccer match (lines 13–16). Next, the data blocks are determined by the *RandomAccessUnit*s related to this temporal segment (lines 17–19).

4.2.2 Data Block Transformation

The data block transformation step is in the first place meant to make changes inside the selected RDF graphs describing a data block. For instance, the value of a *SyntaxElement* can be changed or the length of a *TruncatablePayload* can be shortened. In practice, data block transformation is needed when a particular coding format requires, next to the selection of datablocks, certain syntax element modifications in order to deliver a compliant adapted bitstream. An example of such a coding format is Scalable Video Coding (SVC). For instance, the following approach needs to be followed to adapt the SVC syntax element *num_layers_minus1*, which denotes the amount of scalability layers available in an SVC bitstream. The structural part of the multimedia model needs to be extended with a coding-format specific syntax element as illustrated in Figure 17. In this figure, the class *Num_layers_minus1* is created as a subclass of *SyntaxElement*. When a data block of type *Num_layers_minus1* is detected during the data block transformation step, the value of this syntax element is changed according to the requested scalability properties.

A second use case for data block transformation is the support for dynamic adaptations, i.e., when the multimedia content is delivered during varying usage

environment conditions. In order to avoid the initialization and evaluation of a new query each time a structural adaptation property (e.g., amount of temporal layers) changes, the structural selection is omitted during the data block selection step. More specifically, the semantic selection is established during the data block selection step and the structural selection is performed during the data block transformation step. For example, during data block selection, all data blocks can be selected belonging to the first half of a soccer match; during data block transformation, the frame rate of the video fragment can be scaled to 15 fps by dropping the necessary temporal scalability layers. Hence, when structural adaptation parameters change during the adaptation process, the query does not need to be re-executed.

Currently, no standardized solution exists to transform RDF graphs. Next to ontology-specific software (i.e., write an own RDF transformer based on a specific ontology), the following solutions are available.

- *XML transformation technologies*: XPath and XSLT can be used to access and transform an XML serialization of RDF data. However, the main problem with this approach is that the standard RDF/XML serialization is non-deterministic (i.e., there are many possible serializations for a given RDF model). Furthermore, XPath expressions are not aware of the semantics of the RDF model. Several approaches such as Twig [34] and RxPath [26] define a set of XPath/XSLT extension functions and/or provide a mapping of RDF to the XPath data model, in order to cope with the RDF/XML serialization problem.
- *RDF transformation technologies*: the non-deterministic character of RDF/XML serialization is caused by the fact that XML is tree-based while RDF is graph-based. Converting a graph-based model such as RDF to a tree-based model such as XML is not trivial. Therefore, a graph-oriented RDF access mechanism is needed. RDF Path [23] is one example of an attempt to come to an RDF Path language. It is triple oriented, tries as far as possible to mimic XPath, and treats a graph as an extended tree with no root.
- *SPARQL/Update*: in addition to SPARQL, which provides a retrieval language for RDF, SPARQL/Update (a.k.a. SPARUL, [25]) is proposed as an update language for RDF graphs. It is a language to express updates to an RDF store and is intended to be a standard mechanism by which updates to a (remote) RDF Store can be described, communicated, and stored.

4.2.3 Data Block Binarization

The final step in the workflow of RDF-driven content adaptation is the binarization of the data blocks. More specifically, based on the original media bitstream (present in the file server as shown in Fig. 16) and the selected and adapted data block graphs, an adapted media bitstream is created. The *start* and *length* properties of the data block are used to copy a part of the original bitstream into the adapted bitstream. If the data block is a *Syntax Element*, the syntax element value is written to the adapted bitstream. In case the data block has as type *StuffingBits*, stuffing bits are added to the adapted bitstream until it is byte-aligned.

5 RDF-driven Content Adaptation versus Other Techniques

In this section, a comparison is made between the proposed RDF-driven content adaptation technique and other techniques for multimedia content adaptation, i.e., XML-driven content adaptation and dedicated software approaches. The comparison is based on a number of criteria that are listed below.

Criterion 1: *Format-independency of the software.* The software modules used between the original compressed bitstream and the adapted compressed bitstream are investigated in terms of their independency of underlying coding and/or metadata formats. The more software modules are format-agnostic in the adaptation chain, the more extensible the adaptation framework and the better the support for new formats will be.
Criterion 2: *Knowledge needed for adaptation operations.* This criterion examines the specific knowledge needed to define adaptations. More specifically, to what extent is knowledge needed regarding a specific coding format in order to define a specific adaptation. Within format-independent content adaptation systems, it is important to avoid coding-format specific calculations when defining adaptation operations. Otherwise, format-independency is obtained in an ad-hoc way, as discussed in Section 2.2.
Criterion 3: *Semantic metadata integration.* In order to perform adaptations based on semantic information (e.g., indication of violent video scenes), a straightforward integration between the adaptation logic and the semantic metadata is desired.
Criterion 4: *Adaptation possibilities.* The different kinds of adaptations that are possible with a specific content adaptation technique are examined in this criterion. Furthermore, prerequisites of the

compressed bitstreams are investigated in order to enable certain kinds of adaptations.

Criterion 5: *Performance.* The performance of adaptation techniques is measured in terms of execution speed, memory consumption, and overhead in terms of disk usage.

The first four criteria are discussed below, while the performance measurements are provided in Sect. 6. A summarizing table is provided in Sect. 7. The first four criteria are applied to RDF-driven content adaptation, XML-driven content adaptation, and dedicated software approaches.

Criterion 1: Format-independency of the software

It is clear that dedicated software approaches do not provide format-agnostic software modules. Hence, support for new coding and/or metadata formats requires the development of new software modules.

Both XML- and RDF-driven content adaptation are able to deliver format-agnostic software modules for the content adaptation chain. The transformation and binarization steps are driven by coding-format and metadata agnostic software modules (i.e., modules for transformation and binarization). The generation of structural metadata can also be performed by using format-agnostic software modules; BSDL even provides a standardized solution taking the form of the BintoBSD parser.

Criterion 2: Knowledge needed for adaptation operations

Developing dedicated software for particular multimedia content adaptation operations requires knowledge of the high-level structure of the coding formats that will be supported by the dedicated software (to be able to parse the compressed bitstream), knowledge of the supported semantic metadata formats, and the coupling (i.e., the mapping) of the structure of the compressed bitstream and the semantic metadata.

XML-driven content adaptation requires the same knowledge as the dedicated software approach. The technique claims to rely on high-level descriptions of the media bitstreams. However, as discussed in Sect. 2.2, these descriptions are just a textual representation of the coding-format specific structures and syntax elements. Coding-format specific algorithms to implement coding-format independent adaptation operations such as 'lower the frame rate' have to be used within the BSD transformation step. Therefore, an XML filter is dependent on the coding format and the metadata formats used. Furthermore, the mapping between the structure of the compressed bitstream and the semantic metadata is also defined inside the XML filter. In comparison to the dedicated software approach, XML-driven content adaptation requires less implementation effort since I/O operations are abstracted by the format-agnostic software modules.

RDF-driven content adaptation also requires knowledge regarding the high-level structure of the coding formats, metadata formats, and mapping between structural and semantic metadata; however, this knowledge is obtained during the metadata generation and is separated from the actual adaptation operations. Indeed, the high-level structure of coding formats is mapped to the structural multimedia model (see Sect. 3.1.1) and is independent of possible adaptations. The same holds true for the semantic multimedia model (see Sect. 3.1.4), which contains a description of the semantic information of the multimedia content. Hence, defining adaptations within an RDF-driven content adaptation system is purely based on the multimedia model.

Criterion 3: Semantic metadata integration

As discussed in Sect. 2.2, integrating semantic and structural metadata is obtained in an ad-hoc way in case of XML-driven content adaptation. This is due to the lack of semantic interoperability of XML, resulting in a metadata-format dependency of the XML filters. Furthermore, coding-format specific mappings between structural and semantic metadata need to be created during the adaptation process. The same holds true for the dedicated software approach, since the semantic metadata needs to be mapped to the coding-format specific bitstream structures.

By the definition of a multimedia model, RDF-driven content adaptation provides a seamless integration of structural and semantic metadata. More specifically, the structural part of the model can be seen as a layer on top of media bitstreams, providing support for easy access to the high-level structures and syntax elements of the media resource. Based on a subset of the structural metadata (i.e., a subset of the data blocks), an adapted media resource is generated. Because we are working with an additional layer on top of the media resource (i.e., the structural metadata), it is possible to link these metadata to semantic descriptions of the media resource (i.e., the semantic metadata). Moreover, Semantic Web technologies, which solve the semantic interoperability problem of XML, are inherently present since the multimedia model is implemented by making use of Semantic Web technologies.

Criterion 4: Adaptation possibilities

All adaptation operations are possible by using dedicated software. Both semantic and structural adaptations are possible and do not require any prerequisites of the compressed bitstream, except that the bitstream needs to be encoded in a coding format that is supported by the software. In particular cases, the compressed bitstream can be completely decoded and reencoded in order to perform the necessary adaptations.

XML-driven content adaptation is mainly focussed on the adaptation of scalable bitstreams as discussed in Sect. 2.1.3. Hence, when structural adaptations need to be established, the compressed bitstreams need to be encoded in such a way that it is possible to perform the adaptations without the need of a complete recode process. For example, efficiently exploiting temporal scalability using the H.264/AVC coding format requires the presence of a hierarchical coding structure [8]. The same observation can be made for semantic adaptations: the encoded bitstreams need to provide several random access points in order to extract specific segments of the bitstream.

The adaptation possibilities of RDF-driven content adaptations are equal to the ones of XML-driven content adaptation. They both operate on compressed bitstreams and do not perform any decoding and/or reencoding of the bitstream.

6 Performance Measurements

In this section, the fifth criterion is discussed in the context of XML- and RDF-driven content adaptation. We do not discuss the dedicated software approach since it is clear that the performance of dedicated software in terms of execution speed and memory consumption will generally be better or equal to the performance of format-independent approaches such as XML- and RDF-driven content adaptation. Also, no overhead in terms of disk usage is present in case of dedicated software solutions since no structural metadata is present.

We assume that the metadata generation step is done in advance, i.e., the XML descriptions and RDF triples are present. Previous work has shown that the generation of XML descriptions of the high-level structure of a media bitstream can be performed in real time by making use of BFlavor [7], which is able to automatically generate a coding-format specific parser producing XML descriptions compliant to BSDL. In our research, the same approach was followed in the context of RDF-driven content adaptation, i.e., automatically generated parsers can produce RDF triples compliant with the multimedia model presented in Sect. 3.1. The semantic metadata was obtained by manually annotating the multimedia content.

6.1 Application Scenario

In order to evaluate and compare XML- and RDF-driven content adaptation, the following application scenario is used. A video fragment of a part of a soccer game is present as test sequence, together with the appropriate audio fragment. Furthermore, information regarding the level of importance of specific scenes is present. Each scene was annotated by a number equal to 0 (everything except game interruptions), 1 (goals, chances, and faults), or 2 (only goals).

The video fragment is encoded using four different video codecs (i.e., H.264/AVC, the scalable extension of H.264/AVC (SVC), H.263+, and MPEG-2 Video); the appropriate audio fragment is encoded with two audio codecs (i.e., Advanced Audio Coding (AAC) and MPEG-1 Audio Layer 3 (MP3)). The user selects specific parts of the test sequence based on the level of importance (i.e., semantic adaptation). Furthermore, exploitation of scalability properties of the encoded bitstreams is performed (i.e., structural adaptation). The latter can vary dynamically during the adaptation, implying that the adaptation parameters regarding the scalability properties need to be adjustable in an on-the-fly fashion.

6.2 Experimental Results

6.2.1 Bitstream Characteristics

The video fragment contains a length of 100 s, a frame rate of 25 fps, and a resolution of 720x576 pixels. Selecting scenes with a level of importance of 0, 1, or 2 results in an adapted bitstream of 2484, 1503, or 855 frames respectivily. An overview of the properties of the encoded bitstreams can be found in Table 1. The sizes of the XML descriptions and the amount of RDF triples corresponding to the structural metadata[4] are also shown in this table. Note that the overhead of format-agnostic content adaptation (i.e., the XML descriptions and RDF triples) is dependent on the number of parse units present in the bitstream. The number of parse units will also have an impact on the execution times regarding the adaptation of the bitstreams (see

[4] The amount of RDF triples to represent the scalability information and semantic metadata is in this case negligible in comparison with the amount of triples needed for the structural metadata. Note that in our configuration, one triple took approximately 90 bytes of disk usage.

Table 1 Overview of the bitstream characteristics.

Coding format	Size (MB)	Bit rate (kbit/s)	Parse unit	# Parse units	# Temp. levels	# Spat. levels	# Qual. levels	XML descr. size (MB)	# RDF triples	RDF gen. (s)
H.264/AVC	35.9	2942	NALU	5004	3	1	1	4.4	47330	54.6
SVC	40.8	3343	NALU	15008	3	2	4	18.5	147506	282.5
H.263+	45.2	3705	Picture	5000	3	2	1	7.7	54492	121.6
MPEG-2 Video	77.3	6332	Picture	2500	3	1	1	9.6	29830	67.6
AAC	1.5	128	Frame	4691	-	-	-	3.0	60729	54.8
MP3	2.3	192	Frame	4170	-	-	-	1.9	54136	50.9

Sect. 6.2.3). However, it is clear that both XML- and RDF-driven content adaptation introduce a significant amount of overhead in terms of disk usage due to the occurrence of the structural metadata.

The last column of Table 1 shows the execution times for the generation of RDF triples describing structural metadata and scalability information. These RDF triples can be generated in the same way as XML descriptions suitable for XML-driven content adaptation. More specifically, a modified version of BFlavor [7] was used to automatically create format-specific parsers that are capable of outputting RDF triples (in its original form, BFlavor was only able to automatically create format-specific parsers producing XML descriptions compliant to the BSDL standard). As can be seen in Table 1, the execution times depend on the number of parse units and the bit rate. Note that each parse unit corresponds to an RDF data block graph. SVC and H.263+ perform less than real-time as the length of the test sequences used is 100 seconds. This is due to the use of multiple scalability layers (resulting in a higher amount of parse units) and a higher bit rate.

6.2.2 Implementation Details

In our experiments, BSDL and STX[5] were used to perform XML-driven content adaptation. Bitstream Syntax Schemas[6] (BS Schemas) for the six codecs were designed, together with ten STX stylesheets (six stylesheets to implement the scene extraction (one for each codec) and four stylesheets to implement the exploitation of scalability (one for each video codec)). The STX engine transforms the XML descriptions based on a STX stylesheet. The BSDtoBin parser is a standardized parser from BSDL taking as input the transformed XML description and the original bitstream and which produces the adapted bitstream. The STX engine and the BSDtoBin parser are connected through SAX events. More specifically, no intermediate (transformed) XML description is generated, since the transformed SAX events are immediately transferred to the BSDtoBin parser. This will significantly speed up the XML-driven adaptation process because I/O operations regarding intermediate XML descriptions are avoided.

The data block selection, transformation, and binarization modules of our RDF-driven content adaptation framework are built on top of Sesame[7], which is an open source RDF database with support for RDF Schema inferencing and querying. The built-in SPARQL engine of Sesame is used for the evaluation of queries during the data block selection step. The native store facilities of Sesame were used for our RDF repository, implying that the RDF data is retrieved directly from disk. Using a native store provides better scalability since it is independent of the available system memory. However, when generic RDF storage solutions such as the native store facilities of Sesame become insufficient in terms of scalability due to a high amount of RDF triples (i.e., the structural metadata), other solutions should be considered to store the structural metadata.

For example, one solution for this problem is to store the structural metadata and scalability information in an RDF store which is specifically designed for the model for media bitstreams. More specifically, the structural metadata and scalability information can be stored in a highly scalable Relational Database Management System (RDBMS), using a database scheme based on the structural and scalability part of the model for media bitstreams. Hence, such a RDBMS can be seen as an efficient RDF store specifically designed for our model for media bitstreams. This way, SPARQL queries can be translated into SQL queries. The results of these SQL queries can then be converted back to RDF graphs corresponding to the selected data blocks. RDBMSs should be capable of dealing with a large amount of structural metadata since they are mature, stable, and scalable, while also providing a high performance in terms of query execution speed.

In contrast to the combination of BSDL and STX where a lot of BS Schemas and STX stylesheets need to be developed, only one SPARQL statement and a

[5] Version 1.3.1 of the BSDL reference software and version 2008-03-09 of the STX engine Joost were used.

[6] A Bitstream Syntax Schema describes the high-level structure of a specific coding format.

[7] Version 2.0.1 of Sesame was used.

domain-specific ontology[8] was needed for RDF-driven content adaptation. In order to build this SPARQL statement, knowledge of the multimedia model is needed together with the concepts of the domain-specific ontology which is coupled with the semantic metadata model. Note that the exploitation of scalability was included during the data block selection step (i.e., scalability options are static during the adaptation). Additionally, an implementation where the exploitation of scalability was performed during the data block transformation step (i.e. scalability options can dynamically vary during the adaptation) was used to compare the static and dynamic exploitation of scalability.

Performance measurements were done on a PC having an Intel Pentium D 2,8 GHz CPU and 1 GB of system memory at its disposal. The operating system used was Windows XP Pro SP2, running Java 2 Runtime Environment (SE version 1.5.0_09). The memory consumption of the Java programs was measured by relying on JProfiler 4.2.1. All time measurements were executed six times, whereupon an average was calculated over the last five runs to avoid startup effects (the standard deviation was 0.05 s).

6.2.3 Results

An overview of the execution times and resulting file sizes of the adapted bitstreams is given in Table 2. The names of the resulting sequences correspond to `CCC_iI_tT_sS_qQ`, with `CCC` equal to the coding format; `I` corresponds to the level of importance; and `T`, `S`, and `Q` respectively correspond to the number of temporal, spatial, and quality scalability layers present in the adapted bitstream.

In general, both XML- and RDF-driven content adaptation perform well and are able to adapt bitstreams of various coding formats in real time. As can be seen from Table 2, the coding format (more specifically the number of parse units per frame and the bit rate) has a significant impact on the execution times. For instance, in case of RDF-driven content adaptation, the higher the number of parse units per frame, the higher the number of data blocks that need to be selected, optionally transformed, and serialized. Furthermore, a high bit rate of the adapted bitstream results in a high amount of I/O operations during the binarization of the data blocks. The same observations can be made for XML-driven content adaptation. In Fig. 18, this is illustrated by plotting the execution times together with the adapted file sizes for the video coding formats.

[8] In our case, a simple soccer ontology was used, including concepts such as *SoccerMatch* and *SoccerScene*, as well as properties such as *levelOfImportance*.

Table 2 Overview of the execution times and resulting file sizes.

Sequence	STX + BSDtoBin (s)	RDF-driven adaptation (s)	Adapted file size (MB)
avc_i0_t2	18.0	9.3	35.5
avc_i0_t0	12.8	7.7	20.3
avc_i1_t2	13.2	5.9	22.8
avc_i1_t0	9.9	4.9	12.6
avc_i2_t2	10.0	3.7	14.4
avc_i2_t0	8.1	3.1	7.8
svc_i0_t2_s1_q3	47.1	42.3	40.4
svc_i0_t2_s1_q0	36.7	31.1	38.7
svc_i0_t2_s0_q0	28.9	27.1	3.7
svc_i0_t0_s1_q3	31.7	39.0	23.3
svc_i0_t0_s1_q0	28.9	28.9	22.8
svc_i0_t0_s0_q0	25.5	26.2	2.6
svc_i1_t2_s1_q3	32.7	25.7	25.8
svc_i1_t2_s1_q0	26.5	18.9	24.8
svc_i1_t2_s0_q0	21.6	16.7	2.3
svc_i1_t0_s1_q3	23.4	23.5	14.5
svc_i1_t0_s1_q0	21.7	17.6	14.1
svc_i1_t0_s0_q0	19.6	16.1	1.6
svc_i2_t2_s1_q3	23.3	14.8	16.2
svc_i2_t2_s1_q0	19.8	11.0	15.6
svc_i2_t2_s0_q0	16.8	9.7	1.4
svc_i2_t0_s1_q3	17.9	13.6	8.9
svc_i2_t0_s1_q0	16.9	10.9	8.7
svc_i2_t0_s0_q0	15.6	9.3	1.0
m2v_i0_t2	23.1	7.8	76.6
m2v_i0_t0	13.5	5.3	25.0
m2v_i1_t2	14.7	5.3	48.4
m2v_i1_t0	8.8	3.6	15.6
m2v_i2_t2	9.6	3.3	29.2
m2v_i2_t0	6.1	2.4	9.4
263_i0_t2_s1	20.2	12.4	44.8
263_i0_t2_s0	13.9	8.6	9.6
263_i0_t0_s1	13.6	10.7	22.8
263_i0_t0_s0	11.9	8.1	4.9
263_i1_t2_s1	14.1	7.6	25.1
263_i1_t2_s0	10.5	5.5	5.4
263_i1_t0_s1	10.3	6.8	12.6
263_i1_t0_s0	9.4	5.2	2.7
263_i2_t2_s1	10.2	4.6	13.5
263_i2_t2_s0	8.2	3.4	2.9
263_i2_t0_s1	8.1	4.1	6.7
263_i2_t0_s0	7.6	3.3	1.4
aac_i0	4.5	3.9	1.5
aac_i1	3.4	2.6	0.9
aac_i2	2.7	1.8	0.5
mp3_i0	3.9	3.6	2.3
mp3_i1	3.0	2.5	1.4
mp3_i2	2.4	1.4	0.6

In most of the cases, RDF-driven content adaptation has lower execution times than XML-driven content adaptation, but both techniques have a comparable performance. This is for instance illustrated in Fig. 19, where the execution times for the H.264/AVC coding format are plotted. There are two reasons why RDF-driven content adaptation performs slightly better than XML-driven content adaptation. First, coding-format specific algorithms need to be executed during

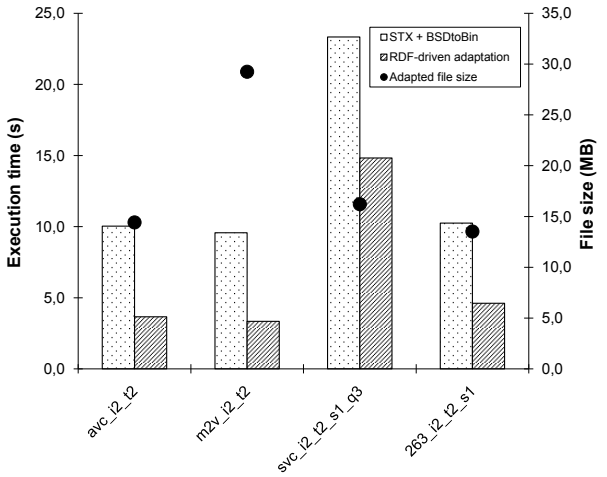

Fig. 18 Execution times and file sizes for the video coding formats.

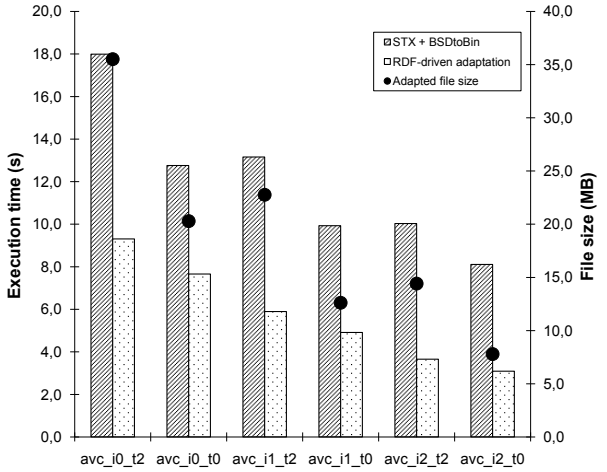

Fig. 19 Execution times for the H.264/AVC coding format.

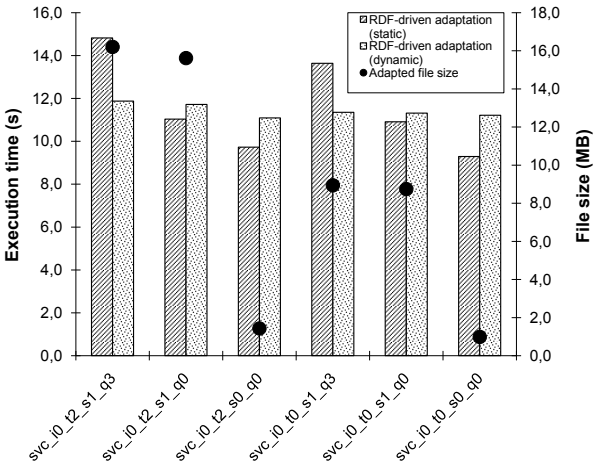

Fig. 20 Execution times for the SVC coding format, comparing static versus dynamic exploitation of scalability.

Table 3 Overview of the discussed criteria.

Criterion	Dedicated software	XML-driven adaptation	RDF-driven adaptation
Format-independency	No	Yes	Yes
Knowledge needed for adaptation	High	High	Low
Semantic metadata integration	Low	Low	High
Adaptation possibilities	All	Coding format dependent	Coding-format dependent
Execution speed	Real time	Real time	Real time
Memory consumption	Low	Low	Low
Metadata overhead	None	Structural metadata	Structural metadata

the transformation step in case of XML-driven content adaptation. As discussed in Section 3, these coding-format specific algorithms are already executed during the structural metadata generation step in case of RDF-driven content adaptation. Second, descriptions compliant to the model for media bitstreams only contain information that is really necessary for the adaptation operation. On the contrary, XML-driven content adaptation needs to process BSDs containing coding-format specific structures and syntax elements necessary to execute these coding-format specific algorithms. Hence, the processing of these BSDs will take longer than the processing of a description compliant with the model for media bitstreams. Further, both XML- and RDF-driven content adaptation have a low and constant memory consumption (approximately 2 MB).

A comparison between static and dynamic exploitation of scalability is shown in Fig. 20. In case of static adaptation, the proper data blocks are already present after the data block selection step. On the contrary, dynamic adaptation implies that during the data block selection step, only a selection is made based on semantic preferences (and not regarding scalability options). Hence, more data blocks are selected during the data block selection step. The actual structural adaptation is then performed during the data block transformation step. This declares why the execution times for the dynamic adaptation process are almost constant (between 11 and 12 s in Fig. 20), since the same amount of data blocks are selected, regardless of the scalability preferences. The little variations in execution times are due to the varying file sizes of the adapted bitstreams.

7 Conclusions and Future Work

In this paper, we have introduced the concept of RDF-driven content adaptation, a multimedia content adaptation technique that operates independent of coding and metadata formats. It is based on the definition of a multimedia model describing and coupling structural metadata, semantic metadata, and scalability information. Existing coding and metadata formats are mapped to our multimedia model. An implementation of the model is realized using Semantic Web technologies such as RDF, OWL, and SPARQL. The adaptation of media bitstreams is performed by selection, transformation, and serialization of the structural metadata, based on the semantic metadata and scalability information.

We have compared RDF-driven content adaptation with dedicted multimedia content adaptation software and XML-driven content adaptation. The latter is another coding-format independent adaptation technique. A summarization of the evaluation of different criteria is provided in Table 3. The most significant advantages of RDF-driven content adaptation are the low amount of knowledge needed to define adaptation operations and the seamless integration with semantic metadata. On the other hand, disadvantages are the coding-format dependency of the adaptation possibilities and the overhead in terms of file size.

Future work consists of solving a number of remaining challenges for RDF-driven content adaptation. First of all, the overhead in terms of file sizes should be reduced; for instance, by performing queries over compressed RDF triples. Secondly, the semantic multimedia model could be extended with support for spatial segments. This will enable the extraction of Region of Interests (ROIs), on condition that the coding format does have support for this. Finally, the relation between RDF-driven content adaptation, which focusses on high-level adaptations, and low-level adaptations (e.g., adaptations where partial decoding operations are needed) should be investigated.

Acknowledgements The research activities that have been described in this paper were funded by Ghent University, the Interdisciplinary Institute for Broadband Technology (IBBT), the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT-Flanders), the Fund for Scientific Research-Flanders (FWO-Flanders), and the European Union.

References

1. Amielh, M., Devillers, S.: Multimedia Content Adaptation with XML. In: Proceedings of 8th International Conference on Multimedia Modeling, pp. 127–145. Amsterdam, The Netherlands (2001)
2. Arndt, R., Troncy, R., Staab, S., Hardman, L., Vacura, M.: COMM: Designing a Well-Founded Multimedia Ontology for the Web. In: 6th International Semantic Web Conference (ISWC 2007). Busan, Korea (2007)
3. Bulterman, D., Grassel, G., Jansen, J., Koivisto, A., Layaïda, N., Michel, T., Mullender, S., Zucker, D. (eds.): Synchronized Multimedia Integration Language (SMIL 2.1). W3C Recommendation. World Wide Web Consortium (2005). URL `http://www.w3.org/TR/SMIL2/`
4. Burnett, I., Pereira, F., Van de Walle, R., Koenen, R. (eds.): The MPEG-21 book. John Wiley & Sons (2006)
5. Cimprich, P., et al.: Streaming Transformations for XML (2004). Available on `http://stx.sourceforge.net/documents/spec-stx-20040701.html`
6. De Bruyne, S., De Schrijver, D., De Neve, W., Van Deursen, D., Van de Walle, R.: Enhanced Shot-Based Video Adaptation using MPEG-21 generic Bitstream Syntax Schema. In: Proceedings of the 2007 IEEE Symposium Series on Computational Intelligence, pp. 6 pp on CD–rom. Honolulu, Hawai (2007)
7. De Neve, W., Van Deursen, D., De Schrijver, D., De Wolf, K., Lerouge, S., Van de Walle, R.: BFlavor: a Harmonized Approach to Media Resource Adaptation, inspired by MPEG-21 BSDL and XFlavor. Signal Processing: Image Communication **21**(10), 862–889 (2006)
8. De Neve, W., Van Deursen, D., De Schrijver, D., De Wolf, K., Van de Walle, R.: Using Bitstream Structure Descriptions for the Exploitation of Multi-layered Temporal Scalability in H.264/MPEG-4 AVC's Base Specification. Lecture Notes in Computer Science, Advances in Multimedia Information Processing **3767**, 641–652 (2005)
9. Decker, S., Melnik, S., van Harmelen, F., Fensel, D., Klein, M.C.A., Broekstra, J., Erdmann, M., Horrocks, I.: The Semantic Web: The Roles of XML and RDF. IEEE Internet Computing **4**(5), 63–74 (2000)
10. Devillers, S., Timmerer, C., Heuer, J., Hellwagner, H.: Bitstream Syntax Description-Based Adaptation in Streaming and Constrained Environments. IEEE Trans. Multimedia **7**(3), 463–470 (2005)
11. Hannuksela, M.M., Wang, Y.K., Gabbouj, M.: Isolated Regions in Video Coding. IEEE Trans. Multimedia **6**, 259–267 (2004)
12. Hong, D., Eleftheriadis, A.: XFlavor: Bridging Bits and Objects in Media Representation. In: Proceedings of IEEE International Conference on Multimedia and Expo, pp. 4 on CD–rom. Lausanne, Switzerland (2002)
13. Hunter, J.: Adding Multimedia to the Semantic Web - Building an MPEG-7 Ontology. In: First Semantic Web Working Symposium (SWWS), Proceedings, pp. 261–281. Stanford, USA (2001)
14. ISO/IEC: 21000-7:2004 Information technology – Multimedia framework (MPEG-21) – Part 7: Digital Item Adaptation (2004)
15. Kay, M.: XSLT Programmers's Reference, 2nd Edition. Wrox Press Ltd., Birmingham, UK (2001)
16. Klyne, G., Carroll, J.J. (eds.): Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation. World Wide Web Consortium (2004). URL `http://www.w3.org/TR/rdf-concepts/`
17. Lerouge, S., Lambert, P., Van de Walle, R.: Multi-criteria Optimization for Scalable Bitstreams. Lecture Notes in Computer Science, Visual Content Processing and Representation **2849**, 122–130 (2003)
18. Magalhães, J., Pereira, F.: Using MPEG standards for multimedia customization. Signal Processing: Image Communication **19**(5), 437–456 (2004)

19. McGuinness, D., van Harmelen, F. (eds.): OWL Web Ontology Language: Overview. W3C Recommendation. World Wide Web Consortium (2004). URL `http://www.w3.org/TR/owl-features/`
20. Mukherjee, D., Delfosse, E., Kim, J.G., Wang, Y.: Optimal Adaptation Decision-Taking for Terminal and Network Quality of Service. IEEE Trans. Multimedia **7**(3), 454–462 (2005)
21. Mukherjee, D., Said, A.: Structured Scalable Meta-formats (SSM) for Digital Item Adaptation. In: Internet Imaging IV, *Proceedings of SPIE*, vol. 5018 (2003)
22. Ohm, J.R.: Advances in Scalable Video Coding. Proceedings of the IEEE **93**(1), 42–56 (2005)
23. Palmer, S.B.: Pondering RDF Path (2003). URL `http://infomesh.net/2003/rdfpath`
24. Prud'hommeaux, E., Seaborne, A. (eds.): SPARQL Query Language for RDF. W3C Recommendation. World Wide Web Consortium (2007). URL `http://www.w3.org/TR/rdf-sparql-query/`
25. Seaborne, A., Manjunath, G.: SPARQL/Update: a language for updating RDF graphs (2008). URL `http://jena.hpl.hp.com/~afs/SPARQL-Update.html`
26. Souzis, A.: RxPath: a mapping of RDF to the XPath Data Model. In: Extreme Markup Language 2006. Montreal, Canada (2006)
27. Thomas-Kerr, J., Burnett, I., Ritz, C.: Format-Independent Multimedia Streaming. In: Proceedings of 2006 IEEE International Conference on Multimedia and Expo (ICME), pp. 1509–1512 (2006)
28. Timmerer, C., Frank, T., Hellwagner, H.: Efficient processing of MPEG-21 metadata in the binary domain. In: Proceedings of SPIE International Symposium ITCom 2005 on Multimedia Systems and Applications VIII. Boston, Massachusetts, USA (2005)
29. Timmerer, C., Panis, G., Kosch, H., Heuer, J., Hellwagner, H., Hutter, A.: Coding Format Independent Multimedia Content Adaptation using XML. In: Proceedings of SPIE International Symposium ITCom 2003 on Internet Multimedia Management Systems IV, vol. 5242, pp. 92–103. Orlando (2003)
30. Van Deursen, D., De Schrijver, D., De Bruyne, S., Van de Walle, R.: Fully Format Agnostic Media Resource Adaptation Using an Abstract Model for Scalable Bitstreams. In: Proceedings of the 2007 IEEE International conference on Multimedia and Expo, pp. 240–243. Beijing, China (2007)
31. Van Lancker, W., De Sutter, R., De Schrijver, D., Van de Walle, R.: A Framework for Transformations of XML within the Binary Domain. In: Proceedings of the IASTED international conference on Internet and Multimedia Systems and Applications, pp. 29–34 on CD–rom. Innsbruck, Austria (2006)
32. Vetro, A., Christopoulos, C., Ebrahimi, T.: Universal Multimedia Access. IEEE Signal Processing Mag. **20**(2), 16 (2003)
33. W3C Multimedia Semantics Incubator Group: Available on `http://www.w3.org/2005/Incubator/mmsem/`
34. Walsh, N.: RDF Twig: accessing RDF graphs in XSLT. In: Extreme Markup Language 2003. Montreal, Canada (2003)