AN MPEG-7 TOOL FOR COMPRESSION AND STREAMING OF XML DATA

Ulrich Niedermeier[1], Jörg Heuer[2], Andreas Hutter[2], Walter Stechele[1], Andre Kaup[3]

[1]Munich University of Technology
Arcisstrasse 21,
80290 Munich, Germany

[2]Siemens AG,
Corporate Technology
81730 Munich, Germany

[3]Chair of Multimedia Communications and Signal Processing
University of Erlangen-Nuremberg
Cauerstr. 7, 91058 Erlangen

ABSTRACT

In the course of the work on the MPEG-7 standard a binary format with special features for the encoding of XML data was required. These required key features are a high data compression ratio, provision for streaming, dynamic update of the document structure and fast random access of data entities in the compressed stream. To support these features we propose a novel, schema-aware approach which exploits the knowledge of the standardized MPEG-7 syntax definition of the encoded XML document on the encoder and decoder side. The technique is part of the MPEG-7 standard. This paper gives an overview of the coding algorithm, including a comparison to standard (XML-) compression tools.

1. INTRODUCTION

The development of the MPEG-7 standard termed the „Multimedia Content Description Interface“ [1] has been initiated to simplify the management, search, retrieval, access, and filtering of audiovisual information. The need for such an interface arises as the amount of digitized multimedia data, that is offered by multiple sources, is growing quickly and makes an efficient and accurate access difficult because the meaning of the multimedia information is not immediately apparent.

In order to solve this problem, MPEG-7 defines an exchange format for multimedia descriptions with a set of standardized descriptors that are associated to the multimedia content. A broad spectrum of different properties of the multimedia material is covered, such as:
- Information about the creation and production process
- Structural information on spatio-temporal components (scene cuts, ...)
- Information about low level features of the content (colors, textures, sound timbre, ...)
- Information about how to browse the content in an efficient way (summaries, ...)

...

Beside other applications these descriptions (usually referenced as metadata) allow a fast and efficient search for, and filtering of material that is of interest to the user.

The following example application scenario illustrates the functional principle:

Search in multimedia databases

Someone is looking for a piece of music and cannot remember the title nor the artist, but a short part of the melody. It is possible to specify the search simply by humming that melody. The relevant MPEG-7 audio-descriptors are extracted and sent to the multimedia database. There the descriptor values are compared with the respective MPEG-7 descriptors of the audio files in the archive, and the best matches are returned.

XML has been chosen as the data format for the metadata descriptions. The syntax of the respective descriptors and description schemes is defined within MPEG-7 in three parts for video, audio and multimedia description schemes. In addition to the textual XML format for metadata a binary format was required which allows special functionalities to be supported:

- high data compression
- possibility of streaming the XML document including the splitting of the document in semantically meaningful fragments and a flexible transmission order of the fragments
- fast random access of data entities in the bit stream
- filtering and parsing on bit stream level
- selective updating of (parts of) a document

Previously developed binary formats for XML, such as WBXML [3] (specification of W3C) or XMill [5], do not satisfy all of the requirements identified by MPEG [2].

2. CONCEPT OF THE BINARY FORMAT

As MPEG-7 targets a wide range of applications, ranging from broadcast over E-commerce to surveillance, it is essential to provide a set of coding options, that allow the binary format to be adopted to the special needs of each application. Contrary to this coding options the structure of an XML document is fixed. In the binary

format two tools are specified to support this coding options.

- the coding-tool *context path* is applied to address one specific node in the tree
- the tool *fragment payload coding* provides the information contained in a whole sub-tree of the description, i.e. the structure as well as the payload.
- additionally *commands* allow the dynamic manipulation of the description tree at the receiver.

With these tools, the description is built up step by step in the description composer at the decoder side.

The XML document structure is hierarchical and can therefore be represented as a data-tree: the properties of each node are broken down by its children in more details. The structure of the tree symbolizes the context in which every node is instantiated in the description. The leaves of the tree contain the information payload: in the case of MPEG-7 these are the actual values of the descriptors.

2.1. Fine granularity of the description

In order to get a meaningful data entity even if the whole document has not yet been transmitted, the binary format allows to decompose the description tree in several sub-trees, where each of them is encoded in its own access unit. An *Access Unit* is defined in this context as the minimal data unit which has to be decoded at one point in time at the decoder. The Access Units can be decoded independently. The granularity, with which the description tree is decomposed into sub-trees and the order of transmission can be chosen at the encoder. Accordingly, nodes or sub-trees of the description tree can be transmitted at the point in time when the data is valid or relevant for the decoder.

2.2. Dynamic update

The binary format allows the dynamic update of description trees that have already been transmitted to the receiver. This is for example useful when transmitting live events, as it is possible to track the actual status of the described media and likewise save bandwidth on the transmission medium and memory at the terminal. The location at which the update shall take place is addressed with a *Fragment Update Context*, and the *Fragment Update Command* specifies the operation to be performed (add, replace, delete, reset).

2.3. Filtering on bit stream level

When searching for the content of a certain node (identified by data type and context) it is possible to determine the exact bit pattern of the Fragment Update Context in the compressed stream. Thus the terminal can identify the relevant Fragment Payload units by simple bit pattern matching, without having to parse and decode the complete bit stream.

3. SCHEMA AWARE ENCODING OF XML DOCUMENTS

The Binary Format for MPEG-7 Metadata (in the following called ‘BiM’) basically consists of two parts: the Context Path and the Fragment Payload Coding. Both tools use a schema-aware approach: the XML Schema definition, which is instantiated by the XML description, is used to assign codes to the individual children of an XML element. These codes are signalled in the binary format to select nodes in the XML description tree. Hence it is necessary that the standardized schema is known by the encoder as well as the decoder. By the use of schema knowledge it is possible to exploit the structural redundancy of the usually very verbose XML document (such as element/attribute name and the nesting of the elements) to achieve a good compression of the document structure.

The Context Path is used to address nodes of the description tree, where an operation on the tree shall take place. The Context Path is essential for the support of the advanced coding options of the binary format such as the transmission of description tree nodes and sub-trees in arbitrary order or fast filtering.

The fragment payload algorithm encodes the information contained in a node or sub-tree, which has been addressed with the Context Path. It gains a very high compression for the structure, and also encodes the content of the description nodes (i.e. the descriptor values).

3.1. The Context Path

The Context Path (CP) specifies the context node in the description tree where a command has to be performed. This context node can be signalled by an *absolute path* that starts at the root element of the description tree, and descends the tree hierarchy to the respective element, or with a *relative path* that starts at the context node which has been encoded previously. The second alternative is usually more efficient in terms of bits spent for encoding the context node, whereas the first alternative has the advantage that it is error robust and fast filtering for nodes can be applied.

In XML Schema complex type definitions specify child elements and attributes. When encoding a CP in a description based on an XML schema definition, the encoder selects in a step the child that is instantiated in the description, and sends the corresponding code to the decoder. In the following, this code is referred to as Tree Branch Code (TBC code). As the complex type definition is also known at the decoder side, the same child is selected. The number of children declared in the complex type definition in the schema determines the length of the code that has to be transmitted. The concatenation of these TBCs results in an absolute or relative path. With respect to the complex type definition the children can vary by

name, type and position. Therefore the TBC is composed of:

- the *Schema Branch Code* (SBC) which selects a named child[1],
- the *Type Code* if polymorphism is applicable, and
- the *Position Code* if multiple instantiations of an element are possible.

These codes are explained in more detail in the following. Figure 1 gives an example for an XML type definition:

```
<complexType name="BookType">
  <complexContent>
    <element name="Author"
             type="PersonType"
             maxOccurs="unbounded"/>
    <attribute name="Title"
             type="string"/>
  </complexContent>
</complexType>
```

Figure 1: Example for an XML type definition

TBC Table of BookType			
SBC	Type Code	Position Code	
00	-	-	Reference to father
01	encoded	encoded	Reference to element Author
10	-	-	Reference to attribute Title
11	-	-	Termination Code

Table 1: Tree Branch Code table

3.1.1. Schema Branch Code (SBC)

A SBC is assigned to every child declared in the complex type definition. For instance in figure 1 the Book Type is defined. In table 1 the SBC assignment is listed. The rules of the assignment are specified in detail in [1]. In addition to the two children of the complex type definition, a reference to the father in the instantiation (relative paths may go upwards in the tree hierarchy) and a termination code are assigned as SBCs. The termination code is used to signal the end of a Context Path.

3.1.2. Position Code

If an element can be instantiated more than once in the description (indicated with the "maxOccurs" attribute in the type definition), it is also necessary to signal the position of the addressed element (see table 1). In the example of figure 1 it is sufficient to signal the position with respect to the other instantiations of "Author". If several groups of elements can be instantiated then a position with respect to all child elements is signalled.

[1] In case an element substitution can be applied also a substitution code is present. We do not elaborate on this due to the fact that currently in the MPEG-7 Schema definition no substitution groups are defined.

3.1.3. Type Code

In the instantiation of an element the type declared in the schema can be changed by a type cast (signalled with the xsi:type attribute in the element instantiation) if the type of the element is a base type. A type is a base type if other types are derived from this type. In case a type cast can be applied the instantiated type has to be signalled to the decoder by the *Type Code*. This code is calculated from the *Inheritance Tree* of the respective type.

3.1.4. Bit stream Syntax

The Context Path can be divided into two parts:

- the first part of the path specifies the context node in which the command is executed by the composer in the decoder
- the second part is a step which selects the operand of the command with respect to the context node.

Accordingly the nodes which are passed in the path are not leave nodes while the operand can be a leave node as well as a node specifying a sub-tree. Nodes within the description tree which contain child elements and attributes are of complex type. Therefore for the encoding of the CP the two parts are distinguished with respect to the TBC tables:

- only child elements of complex type are contained in the TBC tables addressing the context node, and
- all children are contained in the TBC tables used for the addressing of the operand.

According to this coding of the CP the TBC codes are inserted into the bit stream. First the SBCs and Type Codes of the path to the context node, then after a termination code the SBC and Type Code of the operand are signalled and finally all Position Codes are inserted. The data partitioning between SBC/Type Codes and Position Codes has been applied to allow a fast filtering of Fragments.

3.2. Fragment Payload Coding

Like the CP, the Fragment Payload Coding (FPC) uses the syntax definitions in the XML schema to generate the code words, which identify the coded elements [1]. In contrast to the CP finite state automatons are used to represent the content of the complex types. The FPC achieves an even higher compression of the document structure, but at the price of less flexibility, since it does e.g. not allow to select the order of transmission of the elements, which is fixed by the XML schema.

4. RESULTS

In order to show the compression efficiency and to compare it to other algorithms some examples from the MPEG-7 test material have been processed. As reference we use the standard compressor ZIP and the XML-optimized compression tool XMill [5]. Each file is

File	Original size		Fragment payload	Path absolute	Path relative	XMill	ZIP
MonsterJr4Ver3	Complete:	87,466	8,217	-	-	9,582	10,444
	Structure:	**65,825**	**1,526**	**10,948**	**3,495**	**3,179**	**4,392**
	Content:	21,621	6,691	-	-	-	5,501
MdsExamples-Clause4_7	Complete:	27,711	3,757	-	-	5,513	5,758
	Structure:	**20,758**	**488**	**3,090**	**1,272**	**2,008**	**2,513**
	Content:	6,953	3,269	-	-	-	3,038
MdsExamples-Clause11_12	Complete:	136,501	10,488	-	-	12,070	13,000
	Structure:	**104,094**	**2,047**	**17,429**	**5,711**	**2,812**	**4,168**
	Content:	32,407	8,441	-	-	-	7,427
MdsExamples-Clause13_15	Complete:	65,209	7,986	-	-	7,101	7,616
	Structure:	**52,878**	**1,052**	**5,511**	**2,411**	**3,058**	**3,889**
	Content:	12,331	6,934	-	-	-	3,431

Table 2: Comparison of file size (all numbers in bytes)

decomposed in a structure part and a content part in order to evaluate the compression efficiency of the CP. The structure part contains the opening and closing tags of the elements, the content part consists of the values of attributes, and the data within the tags. Structure compression with the CP was done by parsing the document depth-first and encoding the path to every element and attribute in absolute and relative mode.

The structure compression of the FPC as well as the relative CP is usually better than a compression with XMill or ZIP. The CP is used to specify the root node for FPC. Therefore, in most cases only a small part of the BiM bitstream consists of encoded CPs, and thus the values in the columns for the CP of table 2 are an upper limit for the theoretical case, that each node is encoded with a path. The CP provides the essential support for the enhanced functionalities of the binary format. In table 2 the FPC algorithm employs a ZIP compression for string content and represents numbers e.g. as IEEE floating point instead of a character string. Other specific datatype codecs (e.g. for time descriptors) can be plugged in to improve the compression of content. Neither ZIP nor XMill are capable of supporting the advanced coding options of the BiM-algorithm (cf. the requirements in Section 1), as the compression of both tools results in a single non-streamable data entity.

5. CONCLUSIONS

The binary format for XML data that is developed in course of the MPEG standardization as a representation for MPEG-7 descriptions of audiovisual material has been introduced. The wide range of targeted applications requires a highly flexible and scalable data format that also reaches a high compression ratio. The key features supported are:

- flexible transmission and update of MPEG-7 descriptions
- fine granularity of the transmission
- fast random access and filtering capability
- random order of transmission at the encoder side
- high compression efficiency
- direct parsing and filtering on bit stream level without prior decoding into XML

To fulfill these requirements, a context sensitive approach has been chosen that builds on the knowledge of standardised schema definition at the encoder and decoder. Beside the required functionalities this approach shows a good compression ratio (up to 98% for the document structure). As the schema knowledge on the encoder and decoder side is the only prerequisite for the usage of the binary format the technique works not only for MPEG-7, but for all XML documents, that are valid according to a known schema (that need not be the MPEG-7 schema).

6. ACKNOWLEDGEMENTS

The MPEG-7 binary format presented in this paper is the result of the contribution and collaborative effort of many MPEG-7 participants. Special thanks go to Claude Seyrat, Cedric Thienot and Gregoire Pau for their work on the Fragment Payload Coding and Michael Wollborn for the leadership of the “BiM working group”.

7. REFERENCES

[1] ISO/IEC JTC1/SC29/WG11 N4285 “Text of ISO/IEC FDIS 15938-1 Information Technology – Multimedia Content Description Interface – Part 1: Systems”, Sydney, July 2001

[2] ISO/IEC JTC1/SC29/WG11 N4320 “MPEG-7 requirements”, Sydney, July 2001

[3] WBXML: http://www.w3.org/TR/wbxml

[4] MPEG home page: http://mpeg.telecomitalialab.com

[5] http://www.research.att.com/sw/tools/xmill/technique.html