MULTIMEDIA METADATA PROCESSING: A FORMAT INDEPENDENT APPROACH

Robbie De Sutter[*], *Christian Timmerer*[‡], *Hermann Hellwagner*[‡], *and Rik Van de Walle*[*]

* Multimedia Lab, Ghent University, Belgium
{robbie.desutter, rik.vandewalle}@ugent.be
[‡] Department of Information Technology (ITEC), Klagenfurt University, Austria
{christian.timmerer, hermann.hellwagner}@itec.uni-klu.ac.at

Department of Information Technology (ITEC)
Klagenfurt University
Technical Report No. TR/ITEC/05/1.02
February 2005

MULTIMEDIA METADATA PROCESSING: A FORMAT INDEPENDENT APPROACH

Robbie De Sutter[1], Christian Timmerer[2], Hermann Hellwagner[2], and Rik Van de Walle[1]
[1] Multimedia Lab
Ghent University – IBBT, Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium
E-mail: {robbie.desutter;rik.vandewalle}@ugent.be
[2] Dept. of Information Technology (ITEC)
Alpen-Adria-Universität Klagenfurt, Universitätsstraße 65-67, A-9020 Klagenfurt, Austria
E-mail: {christian.timmerer;hermann.hellwagner}@itec.uni-klu.ac.at

ABSTRACT
In multimedia applications, XML is being increasingly used to represent metadata; examples are MPEG-7 multimedia description schemes and MPEG-21 usage environment descriptions. As with the media data, the size of, or the overhead induced by, the XML metadata is important, particularly when used on constrained mobile devices. Therefore, compression (binary encoding) of the XML data becomes relevant to reduce this overhead. Within the MPEG-7 standardization effort, a Binary Format for Metadata (BiM) was developed, providing good compression efficiency and facilitating random access into, and manipulation of, the binary encoded bit stream. However, using binary encoded XML should not introduce interoperability issues with existing applications, nor add additional complexity to new applications. In this paper we investigate a solution for this issue by handling the binary encoded XML data by the XML parser. As such, applications do not need to be aware of the type of encoding of the XML data. In this paper, we introduce such an XML parser and evaluate its usability in different scenarios. We measure the memory requirements and compare the processing speed of parsing binary encoded XML to plain text XML.

KEY WORDS
Multimedia Information Systems, Multimedia Communication Systems, Multimedia Metadata, Binary Encoded XML, MPEG-7 BiM

1 Introduction

As more and more data is structured, stored, and sent over a network using the XML language, the main disadvantage of XML is becoming an issue that no longer can be ignored. XML encodes its data in plain text, thus guaranteeing – to a certain level – platform independent processing thereof. However, this also introduces a lot of overhead, also known as the verbosity of the XML language. It is this overhead that is the main disadvantage of XML. This is especially true when using XML in constrained environments, e.g., mobile devices, where memory, processing power and network bandwidth are limited. On the other hand, these devices become more and more powerful and such constraints seem to be negligible. In practice, however, such network-enabled devices are becoming smaller and smaller and usage of XML-based data is increasing, i.e., similar constraints will apply to future devices as apply to the devices we are using today.

While the World Wide Web Consortium (W3C) has recognized this problem and has created a task force to address the issue[1], the MPEG group has already standardized a solution for the issue within MPEG-7 Part 1, known as Binary Format for Metadata (BiM) [1][2]. BiM was intended to binary encode multimedia descriptions created by other parts of MPEG-7. However, this solution turned out to be very generic as it is able to handle most XML structured data, provided the data is valid to a given XML Schema or Document Type Definition (DTD). Furthermore, BiM achieves compression ratios comparable to plain text compression algorithms [1][3]. On top of that, BiM enables streaming of XML-based data, supports manipulation of the data in the binary domain, and standardizes different commands to update XML data in an optimized way.

It is our belief that, in order for binary encoded XML data to be well adopted, it should not add any additional degree of complexity for the application developers. Nowadays, application developers that need to handle (plain text) XML data, use an XML parser. It is desirable to use the same parser if they want to handle binary encoded XML data. Moreover, they should not need to be aware of the

Acknowledgements. The research activities described in this paper were funded by Ghent University, the Interdisciplinary Institute for Broadband Technology (IBBT), the Fund for Scientific Research-Flanders (FWO-Flanders), the Belgian Federal Science Policy Office (BFSPO), and the European Union.

[1] More information on the W3C XML Binary Characterization Working Group can be found on http://www.w3.org/XML/Binary/

fact they are using binary encoded or plain text XML. It is the parser's duty to handle the XML data correctly.
The remainder of this paper is organized as follows. First, in Section 2, we highlight related work. Next, in Section 3, we briefly explain the functionality of MPEG-7 BiM. Section 4 introduces the XML parser architecture capable of handling plain text XML and binary encoded XML. Section 5 discusses the use cases against which we will evaluate the parser. The results of this evaluation are shown and discussed in Section 6. Finally, Section 7 concludes this paper.

2 Related Work

In the sequel, we give a brief overview of related technologies and standards. Several standardization bodies such as ITU-T[2], ISO/IEC[3] or W3C have recognized the need for an alternative XML serialization. The W3C has recently started with this activity which resulted in a first working draft of relevant use cases [5]. It describes which application areas would benefit from such an alternative XML serialization. Other activities include the efficient transportation of non-XML-based data within XML-based data only, e.g., SOAP messages with attachments [6].
Besides the binary XML efforts within MPEG, ISO/IEC has put some joint efforts with ITU-T towards an alternative XML serialization within the ASN.1 (Abstract Syntax Notation One) group[4]. Therefore, mapping rules between XML Schemas and ASN.1 schemas are defined [7][8] and for ASN.1 instances efficient binary encoding schemes such as Packed Encoding Rules (PER) are available [9]. In practice, however, no common API capable of handling both types of data is available and transformation between the two representations is uneconomical.
Finally, the Web service community has also recognized this issue and is currently developing alternative XML serialization schemes known as Fast Infoset [10] and Fast Web Services [11]. The latter is built upon ANS.1 as described above. The former uses an indexing mechanism which associates an index to each XML element enabling its usage for further occurrences of the same XML element, i.e., highly repetitive content will benefit from this approach. However, for small and complex XML documents the index table is again a burden and performance results comparing these two approaches with other binary XML encoding schemes are currently not available.

3 MPEG-7 Binary Format for Metadata

BiM was initially designed to binary encode MPEG-7 descriptions only. Currently, ISO/IEC has amended its specification to support all kind of XML-based data as long as an XML Schema or DTD is available.
The main features of BiM can be summarized as follows:

- High compression ratio
- Streaming capabilities
- Fast random access
- Filtering and parsing within the binary domain
- Dynamic (partially) updates.

MPEG-7 BiM is an XML Schema aware encoding scheme for XML documents [1], i.e., it uses information from the XML Schema to create an efficient alternative serialization of XML documents within the binary domain. This schema knowledge enables the removal of structural redundancy, e.g., element and attribute names, which achieves high compression ratios with respect to the document structure. Furthermore, element and attribute names as well as data are encoded by using dedicated codecs based on the data type (integer, float, string) which further increases the compression ratio. However, one of the main features of BiM is that it provides streaming capabilities for XML-based data which is one of the main disadvantages of plain text XML. Therefore, BiM divides the XML tree into access units (AUs) containing one or more fragment update units (FUUs). Each FUU includes the FU *command*, FU *context*, and FU *payload* which are described briefly in the following:

- The *command* specifies the decoder action for the corresponding fragment which can be either add, delete, replace, or reset, i.e., BiM also provides partial updates of an XML document.
- The *context* is used to uniquely determine the location of the fragment in the XML document.
- The *payload* contains the actual XML data according to the context.

Figure 1 illustrates how an XML document is divided into AUs and is streamed over the network. In particular, it shows how a sub-tree of the whole XML document is transmitted over the network and added to the description tree at the receiver side (cf. dotted line).

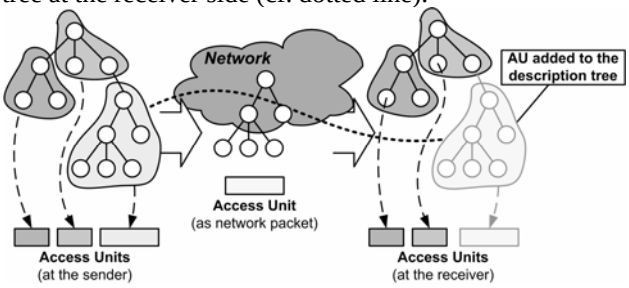

Figure 1: Streaming XML Documents over the Network by using Access Units.

By definition, each AU can be decoded separately while ensuring validity against the XML Schema. The FUUs are processed according to the FU command, i.e., added to, deleted, or replaced from the (partially) instantiated XML document. The reset command resets the BiM decoder and starts again with the initial description tree. Especially the replace command enables selective updates of (parts of) a document which is for example useful when

[2] International Telecommunication Union (http://www.itu.int/home/)
[3] International Organization for Standardization (http://www.iso.org), International Electrotechnical Commission (http://www.iec.ch/)
[4] The ANS.1 group is part of ISO/IEC JTC 1/SC 6/WG 7. See http://www.jtc1sc06.org/ for further details.

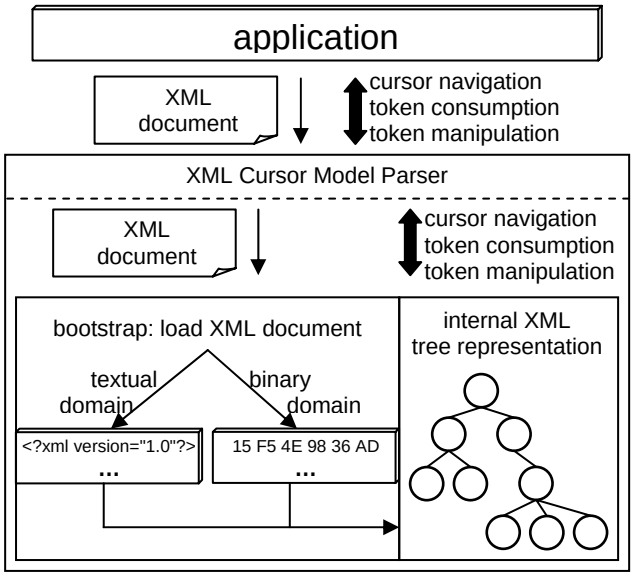

Figure 2: XML Cursor Model Parser Architecture.

transmitting updates of the usage environment from the consumer to the provider as described in Section 5.2.

Finally, the FUU specification allows to perform filter operations within the binary domain, i.e., by means of simple bit pattern matching instead of time-consuming string comparisons. For further information the reader is referred to [1] and [2].

4 Handling Binary Encoded XML data

As explained in the introduction, it is desirable that using binary encoded XML data in applications, does not add an additional layer of complexity and that, optimally, the applications using the XML data should not need to be aware of the fact that the XML data is binary encoded or regular plain text. Current applications are using an XML parser to handle the XML data stream and it is desirable that applications should be able to use the same parser for binary encoded XML. Then, it would be up to the parser to process binary encoded XML data. Former evaluations of different XML parser models [4] have shown that the Curser Model seems to be perfectly suitable to parse XML data, both in the plain text as well as binary domain. Furthermore, the Cursor Model allows one to exploit all functionalities MPEG-7 BiM offers.

As Java does not have a parser compliant to the Cursor Model at this time, we created an interface that defines the signatures of the necessary methods. In our implementation of this interface, the bootstrap method is capable of parsing plain text XML documents and BiM encoded XML documents. This results in an internal XML tree representation as illustrated in Figure 2. The tree can be traversed by using the cursor navigation methods, read by the token consumption methods, and written by the token manipulation methods.

An application uses the bootstrap method to parse an XML document without being aware if this document is plain text or BiM encoded. It is the bootstrap method's responsibility to figure out how the received XML document is encoded. This can be done either trivially by looking at the file extension, or more advanced by using the mime type information or by inspecting the first bytes of the stream. Other detection algorithms are possible.

Once the bootstrap method has identified the encoding, it can process the data. Our parser implementation uses the very fast and lightweight XML Pull Parser[5] to handle the plain text data. The BiM encoded data is handled by a modified version of the MPEG-7 BiM reference software (July 2004 version) [12]. The modification makes it possible that, during decoding, the internal XML tree representation is created on-the-fly. The internal tree is created by reading the source data only once. This is true for both plain text as well as binary encoded XML data streams.

Another issue that arises when using BiM is the fact that the BiM decoder must dispose of the correct XML Schemas. For this, the current reference software implementation requires a *decoder initialization* file. As it is intolerable that the application should provide this file, we solved this issue by creating a repository of all XML Schemas used in the tests. Thus, it is the parser implementation that provides the decoder initialization information and not the application.

5 Evaluation of the Architecture: Use Cases

In order to asses the usefulness of the parser described in the previous section, we evaluate it against two distinct use cases: (1) data storage and (2) usage environment notification.

5.1. Use Case 1: Data Storage

The first use case evaluates the usefulness of binary encoded data as a way to reduce the required disk space to store XML data, but maintaining fast access and manipulation of the data. BiM has the advantage over traditional plain text compression techniques that the compressed data does not need to be decompressed before handling. When using the parser described in the previous section, the application is not even aware whether or not the data was binary encoded. In this use case, the overhead associated to encode the data and to bootstrap the parser are most important. Memory requirements are less important as long as they are not exuberant.

In this use case we analyze six files which uses the same XML Schemas, but differ in file size: from tiny and small (2 – 10 kB) over medium (230 kB) to large and very large (500 – 4000 kB).

5.2. Use Case 2: Usage Environment Notification

Multimedia resource adaptation is becoming more and more important in order to achieve Universal Multimedia Access [13]. Video and audio resources are dynamically

[5] Information about XML Pull Parser can be found on http://www.xmlpull.org/

modified so that the modified resource is optimized for the target application or device. MPEG-21 Part 7, Digital Item Adaptation (DIA), standardizes various tools that can help perform multimedia resource modifications [14]. DIA specifies, among others, description formats to describe the usage environment/context in which, for example, such audio/video streams are consumed. These usage environment description (UED) formats provide means for describing information about the (end) user, the terminal, the network, and the natural environment. In particular, the UED defines XML Schemas together with a textual description thereof. In addition to the MPEG-21 DIA standard, other standards have emerged allowing applications or devices to describe their usage context [15]. All most recent standards have one thing alike: XML is used to structure the usage context. However, for this use case we use the UED part of the MPEG-21 DIA standard.

In practice, once a device or application has collected its usage context, it sends this information to the server. The server uses the information as one of the inputs for an adaptation engine. The adaptation engine adapts the multimedia resource appropriately so the originating device or application retrieves an optimized multimedia resource.

In this scenario there are two issues that need to be considered:

1. Usually, the usage context is not static, e.g., network conditions such as the available bandwidth fluctuate. When this occurs, the server (or any intermediary node which is capable to benefit from this information) must be informed of these changes.
2. Most end user devices requiring an adaptation of the multimedia content have usually limited memory or processing capabilities, e.g., handhelds or cell phones. Furthermore, these devices connect to a server over a constrained network connection where, in some cases, the end user pays a fee based on the number of bytes transferred, e.g., when using a General Packet Radio Service (GPRS) connection.

Due to the dynamic partial update capabilities and high compression ratios, BiM seems to be the ideal candidate for dealing with the above issues. Firstly, BiM is capable of encoding only the UED information that has changed and needs to be transmitted over the network. This eliminates the need to send the full usage context information whenever a change thereof occurs. And secondly, with regards to the constrained network connection, BiM reduces the number of bytes to be sent in two ways: by binary encoding of the information and by only sending the updated information. This can result in a huge saving, not only in required bandwidth, but also in the fee the end user pays. However, to achieve these possible advantages, the device itself must be able to encode XML data by a MPEG-7 BiM compliant encoder. Therefore, the required memory for the encoding is also important in this scenario.

To evaluate this scenario we created a conceivable usage context compliant to MPEG-21 DIA UED which is binary encoded using the BiM reference software. Both the plain text version and the binary encoded version of the usage context is sent as an attachment of a Simple Object Access Protocol (SOAP)[6] message to a server. When receiving the SOAP message, the server bootstraps the parser with the file in the attachment. Thereafter, three kinds of updates are sent from the device to the server:

- Small modification: the bandwidth of the network changes.
- Medium modification: the terminal's display information changes.
- Big modification: the user and the natural environment change.

For the plain text encoding, these modifications are applied to the initial complete usage context description and result in three new full UEDs. For the BiM encoding, three access units are created. The small modification AU contains one FUU that replaces the existing node in the initial complete UED with the updated one. The medium modification requires two FUUs: one to replace the existing information about the display with the new information and a second FUU to remove information about a second display that was present in the initial description. The big modification also contains two FUUs replacing the existing user information and the natural environment information respectively.

6 Results and Discussion

We have evaluated each use case by measuring the *time* and the *maximum heap size* that is required to bootstrap the parser with an XML document, either in plain text or BiM encoded. Additionally, the same statistics are provided for the BiM encoding process.

The run time results were obtained as follows. Due to the fact that the BiM en-/decoder constructs a finite state automaton based on the used XML Schemas – which is a time consuming process – we provide the measurements for the *first run* and an *average run* based on five consecutive runs. Additionally, each run was performed four times and the average was calculated thereof. This was necessary to eliminate possible time fluctuations during the measurements, e.g., due to a run of the garbage collector. The automaton does not need to be reconstructed unless the XML Schema changes. We have not considered a modification of the XML Schema in our use cases. However, for further information on the creation and usage of the automaton the reader is referred to [2].

The experiments were performed on a machine with an Intel Pentium 4 Northwood processor running at 3.2 GHz with hyperthreading disabled and 512 MB RAM. All the tests ran on Windows XP Professional Service Pack 2 with Java version 1.4.2. The JProfiler[7] 3.1 provided the memory measurements, i.e., the maximum heap size.

[6] SOAP is a W3C Recommendation that specifies a lightweight protocol intended for exchanging XML data in a decentralized, distributed environment (see also http://www.w3.org/TR/soap/ for further information).

[7] Java Profiler, JProfiler, can be found on http://www.ej-technologies.com/.

Table 1: Results for encoding/bootstrap time and memory requirements for use case 1.

		File 1: tiny		File 2: small		File 3: medium		File 4: large		File 5: very large	
		Plain text	*BiM*	*Plain text*	*BiM*	*Plain text*	*BiM*	*Plain text*	*BiM*	*Plain text*	*BiM*
File size (bytes)		1,055	149	11,580	1,937	30,667	2,793	550,677	94,332	4,122,543	1,136,997
Encoding time (ms)	*First run*	-	613.3	-	882.8	-	2793.0	-	4,773.8	-	21,750.0
	Average run	-	93.1	-	155.5	-	1489.8	-	3,386.8	-	20,586.8
Bootstrap time (ms)	*First run*	11.8	344.0	19.5	344.0	109.5	347.5	238.3	359.5	37,738.5	535.0
	Average run	0.8	25.1	4.0	26.7	75.1	28.1	185.2	34.5	37,680.5	161.8
Maximum heap size (Kbytes)	*Bootstrap*	1,900	1,900	1,900	1,900	4,250	1,900	8,750	1,900	65,000	6,500
	BiM encoding	3,150		3,350		7,890		13,950		65,000	

Note that the maximum heap size is not the actually consumed memory size, but the maximum size of heap the Java Virtual Machine (JVM) has reserved. However, we used this value as the amount of memory the JVM needs in order to execute the desired operations, even though not all of the reserved memory is actually used.
For binary encoding of the XML documents we used the MPEG-7 BiM reference software. It is important to emphasize that the reference software is *not optimized* in terms of performance, neither in memory usage nor processing speed.
Finally, this evaluation does not focus on the compression efficiency of BiM which is discussed intensively in [1][3].

6.1. Results

Table 1 shows the run times to encode and to bootstrap the parser for the first use case as well as the required maximum heap size. Additionally, the file sizes of both the plain text and BiM encoded XML documents are given. Table 2 shows the analogous results for use case 2 including the size of the SOAP messages.

6.2. Discussion

First, we discuss the run times results listed in Table 1 and Table 2, i.e., encoding and bootstrap times.
The difference between the *first run* and the *average run* is very big. When using BiM, this can be explained by the creation of the automaton as discussed earlier in this section. However, the gap is too broad and furthermore we also notice a difference in the first and average run of the plain text tests. This can be explained by the fact the all required files, i.e., class files and source XML document files, are already loaded at least once for the first run and thus they are still available in a cache. This results in less I/O operations – a very time consuming operation.
We further see the influence of the used XML Schemas on the encoding and bootstrap time: while the complete UED file of use case 2 is about the same size as the small file of use case 1, it takes significantly longer to process the UED file. The more complicated and larger UED XML Schema compared to the XML Schema used in use case 1 explains this difference. This also justifies our choice to use the same XML Schema for all five files in use case 1. Another observation is the fact that small plain text files are always processed faster then their binary encoded counterparts. Medium sized files are – for an average run – processed faster when they are binary encoded. However, when we take the time needed to encode these files into account, then only very large files profit from the BiM encoding in terms of processing speed. The reason why the bootstrap operation for update AUs in use case 2 is relatively slow compared to the file size can be explained by the fact that the existing internal XML tree representation is first cloned – which is necessary to support the "reset" FUU command – and that the FUU context must be located in the tree.
For the maximum heap size measurement the conclusion is as follows: the larger the input file, the more memory is required. In Table 2, the difference in required memory for the complete UED and the updates is due to the creation and storage of the clone of the existing XML tree. Additionally, Table 2 shows also the size of the SOAP messages used to send the information from a client device to the server. Using SOAP only implies a small and constant overhead penalty. Thus, SOAP can be used as a messaging protocol for binary encoded XML. Furthermore, the table shows that when sending the initial description and all updates in binary encoded format, only 15% of the bytes are required compared to doing the same with plain text encoding. In particular, this will result in higher expenses for the end user when using GPRS connections as mentioned in Section 5.2.

7 Conclusion

As XML is nowadays used to store even more (meta-)data the verbosity of the format is a disadvantage that can no longer be ignored. Creating a binary representation of the plain text XML data is a solution for this problem. This alternative XML representation should neither imply interoperability issues with existing applications nor should it add more complexity to new applications.

Table 2: Results for encoding/bootstrap time, SOAP message size, and memory requirements for use case 2.

		Complete UED		Small change		Medium change		Big change	
		Plain text	*BiM*	*Plain text*	*BiM*	*Plain text*	*BiM*	*Plain text*	*BiM*
File size (bytes)		8,701	1,714	8,704	305	8,459	677	13,594	1,835
Encode time (ms)	*First run*	-	1,335.8	-	1,328.0	-	1,328.0	-	1,547.0
	Average run	-	292.2	-	281.3	-	284.3	-	362.6
Bootstrap time (ms)	*First run*	12.0	992.3	19.3	355.3	19.3	363.4	15.3	453.0
	Average run	3.9	207.2	4.6	197.7	1.6	195.4	3.2	183.7
SOAP message size (bytes)		9,060	2,086	9,063	677	8,818	1,049	13,947	2,207
Maximum heap size (Kbytes)	*Bootstrap*	1900	4550	1900	7050	1900	7050	1900	7050
	BiM encoding	7650		7650		7650		7650	

In this paper, we presented a solution for the verbosity issue by applying MPEG-7 BiM to plain text XML and thus creating a compact representation of the XML-based data. We demonstrated an XML parser capable of handling plain text XML and BiM encoded XML. By creating a parser capable of handling both encoding types, applications making use of this parser do not need to be aware if the XML data they use in the application is BiM encoded or regular plain text.

We evaluated the usefulness of this parser for two distinct use cases. The results show that using binary encoded XML data does not impose (extensive) additional memory requirements. Parsing the data is, however, slower compared to plain text XML parsing with the restriction that non-optimized reference software was used.

By ensuring that only the XML parser should be aware of the type of encoding, applications can support binary encoded XML directly and without any additional complexity. The loss of human readability of XML file is amply compensated for by the reduction in the file size, especially when sending this data over a network as described in the second use case.

References

[1] U. Niedermeier, J. Heuer, A. Hutter, W. Stechele, and A. Kaup, An MPEG-7 tool for compression and streaming of XML data, *Proc of the 2002 IEEE International Conference on Multimedia and Expo (ICME)*, vol. 1, Lausanne, Switzerland, 2002, 521–524.

[2] J. Heuer, C. Thienot, and M. Wollborn, Binary Format, in: B.S. Manjunath, P. Salembier, and T. Sikora (eds.), Introduction to MPEG-7: Multimedia Content Description Language, *John Wiley & Sons Ltd.*, NJ, 2002.

[3] C. Timmerer, H. Hellwagner, J. Heuer, C. Seyrat, and A. Hutter, BinaryXML – A Comparison of Existing XML Compression, *ISO/IEC JTC1/SC29/WG11 MPEG2004/M10718*, Munich, Germany, March 2004.

[4] R. De Sutter, C. Timmerer, H. Hellwagner, and R. Van de Walle, Evaluation of Models for Parsing Binary Encoded XML-based Metadata, *Proc. of the 12th IEEE Symposium on Intelligent Signal Processing and Communication Systems*, Seoul, Korea, 2004, 419 – 424.

[5] M. Cokus and S. Pericas-Geertsen (eds.), XML Binary Characterization Use Cases, *W3C Draft*, 2004.

[6] J. J. Barton, S. Thatte, and H. F. Nielsen (eds.), SOAP Messages with Attachments, *W3C Note*, 2000.

[7] ITU-T and ISO/IEC, Encoding Using XML or Basic ASN.1 Value Notation, *Rec. X.693 (2001), ISO/IEC 8825-4:2001*, 2001.

[8] ITU-T and ISO/IEC, Mapping W3C XML Schema Definitions into ASN.1, *ITU-T Rec. X.694 (2004), ISO/IEC 8825-5:2004*, 2004.

[9] ITU-T and ISO/IEC, Specification of Packed Encoding Rules (PER), *ITU-T Rec. X.691 (2002), ISO/IEC 8825-2:2002*, 2002.

[10] P. Sandoz, A. Triglia, and S. Pericas-Geertsen, Fast Infoset, Sun Developer Network Technical Article, June 2004. http://java.sun.com/developer/technicalArticles/xml/fastinfoset/.

[11] P. Sandoz, S. Pericas-Geertsen, K. Kawaguchi, M. Hadley, and E. Pelegri-Llopart, Fast Web Services, Sun Developer Network Technical Article, August 2003. http://java.sun.com/developer/technicalArticles/WebServices/fastWS/.

[12] ISO/IEC 15938-6:2003 Information technology - Multimedia content description interface - Part 6: Reference software, 2003.

[13] R. Mohan, J. R. Smith, and C.-S. Li, Adapting Multimedia Internet Content for Universal Access, *IEEE Transactions in Multimedia*, *1*(1), 1999, 104–114.

[14] A. Vetro and C. Timmerer, Overview of the Digital Item Adaptation Standard, to appear in *IEEE Transactions on Multimedia*, Special Issue on MPEG-21, Feb. or Apr. 2005.

[15] R. De Sutter, F. De Keukelaere, and R. Van de Walle, Evaluation of Usage Environment Description Tools, *Proc. of the 2004 International Conference on Internet Computing*, Las Vegas, NV, USA, 2004, 66–72.