April 2007

Is That a Fish in Your Ear? A Universal Metalanguage for Multimedia

Joseph Thomas-Kerr
University of Wollongong, jak09@uow.edu.au

I. Burnett
University of Wollongong, ianb@uow.edu.au

C. H. Ritz
University of Wollongong, critz@uow.edu.au

S. Devillers
France Telecom

D. de Schrijever
Ghent University, Belgium

See next page for additional authors

Recommended Citation

Thomas-Kerr, Joseph; Burnett, I.; Ritz, C. H.; Devillers, S.; de Schrijever, D.; and Van de Walle, R.: Is That a Fish in Your Ear? A Universal Metalanguage for Multimedia 2007.
https://ro.uow.edu.au/infopapers/522

John R. Smith
IBM

Is That a Fish in Your Ear? A Universal Metalanguage for Multimedia

Joseph A.I. Thomas-Kerr, Ian S. Burnett, and Christian H. Ritz
University of Wollongong, Australia

Sylvain Devillers
France Telecom

Davy De Schrijver and Rik Van de Walle
Ghent University—IBBT, Belgium

"What's this fish doing in my ear?"
"It's translating for you. It's a Babel fish. Look it up in the book if you like."
"The Babel fish," said The Hitchhiker's Guide to the Galaxy quietly, "is small, yellow and leech-like, and probably the oddest thing in the Universe. [I]f you stick a Babel fish in your ear you can instantly understand anything said to you in any form of language."

—Douglas Adams, *The Hitchhiker's Guide to the Galaxy*

Researchers have not, as yet, come up with anything quite as ... err ... ergonomic as the Babel fish. However, they have made substantial efforts toward Universal Multimedia Access (UMA)—the idea of being able to access any multimedia content, anywhere, and at any time.[1] Much of this effort is aimed at devising means to interact with multimedia content—store it, deliver it, process it, consume it, and so on—independently of the content's representation. This format-independent approach to multimedia is increasingly important as the number of encoding formats and diversity of multimedia devices and networks grows.

This article looks at one of the foundations of UMA: a metalanguage for describing the structure of multimedia data. The Bitstream Syntax Description Language (BSDL)[2] lets devices perform operations on data without requiring specific software for every format they might encounter (see the sidebar for links to further information). This is in contrast to current technologies, where the release of a new multimedia content format (H.264/Advanced Video Coding [AVC],[3] for example) requires multimedia software and hardware providers to (largely independently) develop new modules for their products to take advantage of the format.

Editor's Note

Universal Multimedia Access promises to adaptively deliver multimedia content to users according to their needs—whether it's their device, context, or preferences. Central to UMA is the development of metadata standards for describing multimedia resources to allow their adaptation. In this article, the authors report on the development of the Bitstream Syntax Description Language (BSDL) and describe applications for scalable content adaptation, format independent streaming, and delivery and configurable media coding.

—*John R. Smith*

The Bitstream Syntax Description Language

BSDL was originally developed as part of the MPEG-21 multimedia framework's *adaptation* component. In this context, it describes the macro-level structure of scalable content, so adaptation software can perform temporal, spatial, or signal-to-noise ratio (SNR) adaptation simply by identifying which portions of the content to transmit and which to drop.[1] Consequently, once the software identifies a target frame rate, resolution, or quality level, it can execute the adaptation without specific information about the bitstream beyond what BSDL gives it.

Researchers are now starting to use BSDL's format-independence in other parts of the multimedia delivery chain as well (see Figure 1). A format-independent framework for streaming multimedia content,[4] built on top of BSDL, lets streaming servers deploy content in new formats as they're developed, without needing additional software. Work is also underway on a decoder architecture that lets both special- and general-purpose multimedia devices render content in any format, without prior software support.[5] Applications using this architecture will perform bitstream parsing based on a BSDL Description.

We'll return to each of these application domains later. First, though, we look at BSDL in more detail, illustrating the model it uses to

1070-986X/07/$25.00 © 2007 IEEE

Published by the IEEE Computer Society

describe bitstream syntax and its foundations in XML Schema.

A model for Bitstream Syntax Description

BSDL uses XML to describe the structure of multimedia data. Figure 2a (next page) is an example BSDL Description for a video in H.264/AVC format. In general, we can describe multimedia data as a sequence of binary symbols of arbitrary length—some symbols contain a single bit, while others contain many bytes. The BSDL Description indicates these binary symbols' values in a human- and machine-readable format—for example, using hexadecimal values (as for `startCode` in Figure 2a), integers, or strings. It also organizes the symbols into a hierarchical structure that reflects the data's semantic interpretation.

Multimedia bitstreams typically contain millions or billions of symbols, and for many BSDL applications it's neither necessary nor efficient to describe every one. Instead, BSDL lets you describe large segments of data using pointer-like structures, indicating the offset and length of the segment in the raw content. In this way, BSDL hides unnecessary details from an application by encapsulating them in a "black box," represented by the pointer. In our example (Figure 2a), the `payload` field is a pointer to data beginning at byte 5 and continuing for 100 bytes.

In other words, we can fully customize the BSDL Description's level of granularity to the application's requirements. For the same multimedia data, one application can use BSDL that describes some or all of the fields that another application hides in a black box. As a result, the BSDL Description doesn't replace the original data, but instead provides additional information (or metadata) to help an application parse and process the content. Finally, BSDL doesn't mandate the names of the elements in the BSDL Description; the application assigns names that provide meaningful semantics for the description at hand.

Foundations in the XML Schema

In Figure 2a, BSDL describes the values of binary symbols in the multimedia data. However, by itself, this description lacks certain details that are fundamental to an application needing to process the data. Specifically, although the example describes one particular bitstream, it doesn't give any information about how we might derive a similar description for a different bitstream encoded using the same format. Also, if the aim is to reconstruct the binary data from a processed description, further information is missing, such as the number of bits to use when writing each symbol.

Because all the multimedia data in a given format conforms to a single (although potentially flexible) structure, this structural information is expressed, not in the BSDL Description itself, but in a separate schema. This *BSDL Schema* specifies the encoding associated with all instances of data in the given format.

BSDL builds on top of the XML Schema, a World Wide Web Consortium (W3C) tool for expressing constraints on a family of XML documents' structure and data types. The BSDL Schema adds further constraints to an XML Schema that let it elegantly describe not only the XML (the BSDL Description), but also the binary data (the bitstream) and the mapping between the two. Figure 2b shows the BSDL Schema associated with the BSDL Description in Figure 2a.

The BSDL Schema lets an application parse a bitstream, generating a description for subsequent processing. Furthermore, the BSDL Schema lets applications reconstruct a processed version of the multimedia data from its description.

Further Information

The Bitstream Syntax Description Language (BSDL) is a part of the MPEG-21 multimedia framework (ISO/IEC 21000-7). It's available at http://www.iso.org.

Software for BSDL is publicly available from http://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_IEC_21000-8_2006_Reference_Software/21000-7_DIA/DIA-RefSW-A-1_0_4.zip.

Example BSDL Schemata for H.264/AVC and other formats are available at http://multimedialab.elis.ugent.be/BSDL/.

For examples of the streaming and delivery framework, go to http://whisper.elec.uow.edu.au/people/jtkerr/bbl.html.

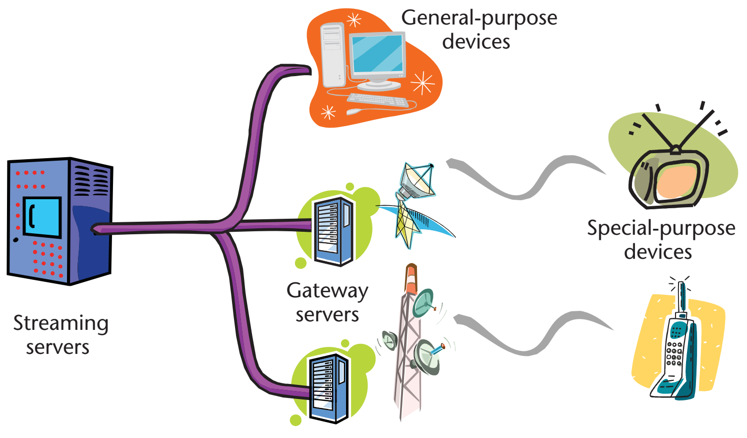

Figure 1. The Bitstream Syntax Description Language (BSDL) is now used at multiple points in the multimedia delivery chain.

```
<NALUnit>
 <startCode>00000001</startCode>
 <forbidden0bit>0</forbidden0bit>
 <nalReference>3</nalReference>
 <nalUnitType>20</nalUnitType>
 <payload>5 100</payload>
</NALUnit>


<NALUnit>
 <startCode>00000001</startCode>
 <!— and so on... —>
</NALUnit>
```

(a)

```
<element name=”NALUnit”
         bs2:ifNext=”00000001” cmc:port=”nal”>
 <complexType>
    <sequence>
      <element ref=”startCode”/>
      <element ref=”forbidden0bit”/>
      <element ref=”nalReference”/>
      <element ref=”nalUnitType”/>
      <element ref=”payload”/>
    </sequence>
 </complexType>
</element>

<element name=”startCode”>
 <simpleType>
     <restriction base=”hexBinary”>
       <length value=”4”/>
      </restriction>
 </simpleType>
</element>

<element name=”payload” type=”bs1:byteRange”/>

 <!— and so on... —>
```

(b)

Figure 2. Examples of BSDL Descriptions and Schemata. (a) A BSDL Description exposes the fields in a multimedia bitstream, in this case an H.264/AVC video. (b) A BSDL Schema describes the structure common to all H.264/AVC video bitstreams.

BSDL applications

Several applications currently use BSDL to process multimedia. One uses BSDL to adapt scalable content, while others use it in format-independent streaming and delivery and in configurable media coding.

Scalable content adaptation

The UMA approach lets content providers adapt multimedia to better fit the capabilities and preferences of the delivery channel, receiving terminal, and end user.[1] This is increasingly important as multimedia devices become more and more diverse: one user can view streaming high-definition video on a PC with a broadband Internet connection, while another watches the same content on a cellular phone, over a low-bandwidth, error-prone wireless network. A content provider could store two versions of the video to deliver to each of these clients, but this approach quickly becomes infeasible as the number of videos that must be stored would increase exponentially.

A better approach is to use audio and video encoding methods (such as H.264/AVC and its extensions) that let an application drop large sections from the compressed bitstream while still letting the receiver decode what's left (albeit at a lower resolution or quality). Then, the content provider only needs to store a single version of each piece of content, which the server (or nodes within the network) can adapt as required.

The MPEG-21 multimedia framework provides a format-independent adaptation architecture (see Figure 3)[2] that enables adaptation without specific knowledge of the bitstream format being processed. This architecture facilitates interoperability for new content formats by enabling adaptation without modifying the existing infrastructure.

The process is based on performing the adaptation on a high-level representation of the bitstream (a BSDL Description) rather than on the bitstream itself. An adaptation processor first generates the BSDL Description from the bitstream via a generic processor, aided by a BSDL Schema file. It then transforms the description according to a set of instructions (such as Streaming Transformations for XML [STX]), and guided by additional metadata such as descriptions of the usage environment or delivery channel. The transformation process typically involves dropping sections of the bitstream and then adjusting header fields to reflect the dropped data. Finally, the processor reassembles an adapted version of the content, based on the transformed description (which contains pointers to the original data), and the BSDL Schema.

Streaming and delivery

A second application that uses BSDL to describe multimedia syntax is a format-independent streaming and delivery framework,[4] depicted in Figure 4. The motivation for such a framework is twofold.

First, as discussed earlier, the ever-growing number of multimedia coding formats and increasing diversity of devices and networks presents an escalating challenge to interoperability, because hardware and software vendors must upgrade their products for each new format if it's to be supported. If multimedia delivery infrastructures are to serve current and future demands, we need new techniques to deal with this increasing complexity. BSDL can help to abstract the details of content delivery away from program code and into a separate data file. Such a data file instructs the delivery processor as to how content of a particular format should be delivered over a specific channel, and lets the existing infrastructure deliver content in new formats, simply by disseminating a new instruction file.

Second, several recent standards (including MPEG-21 and TVAnytime) establish the *virtual multimedia container* concept. Such containers organize collections of content and associated metadata into semantic structures. These containers are format-agnostic—that is, users of the standard encode both content and metadata in any format of their choice. As a result, delivering a virtual container to an end user requires a new—similarly format-agnostic—approach. Delivery might involve streaming the content over a certain channel, or, alternatively, packaging content into one or more file formats to be delivered by some other means. Such packages could be MPEG-21 digital items, for example, or files belonging to one of the recent MPEG multimedia application formats.[6]

The first stage in the streaming or packaging process is *fragmenting* the input data into semantic units appropriate for the intended delivery. This might be audio frames, video content slices, scalability layers, or pieces of metadata. This process is format-independent, so the processor doesn't contain detailed knowledge of any particular format. Input data and metadata can be XML, or an arbitrary binary encoding. In the second case, one or more BSDL Schemata identify the actual encoding's syntax. In this context, BSDL operates as an abstraction that lets the processor handle binary content using the same syntax as native XML.

The next stage is *packetizing* the data fragments by assigning a delivery time stamp and other parameters specific to the output format (such as the marker bit or timestamp cut in the case of the widely used Real-time Transport Protocol [RTP]).

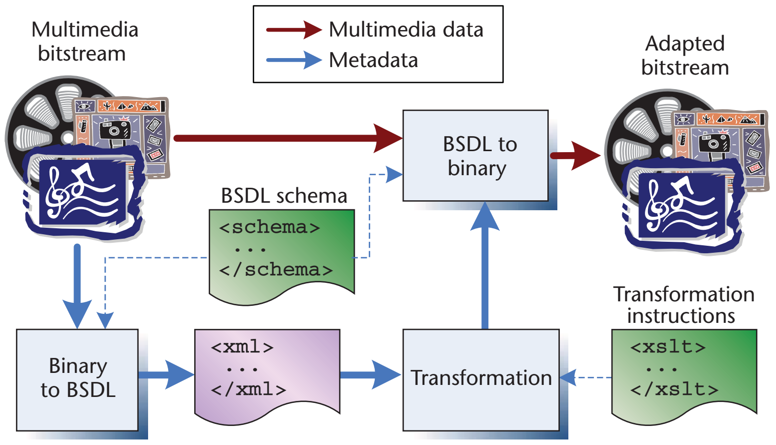

Figure 3. BSDL facilitates a format-independent adaptation framework for scalable content.

Finally, an *output handler* processes the packets for the delivery format. The handler mechanism is extensible—implementers of the framework can define additional handlers as required. Users initialize the handler with global parameters, such as a timescale or payload type code for RTP. The handler is then responsible for outputting each packet according to the parameters provided, in conformance with the standard to which it complies.

Configurable media coding

Over the past 20 years, MPEG has released numerous multimedia encoding formats that have been widely adopted across all digital media fields. Typically, the more recent standards build upon the coding tools developed for those preceding them. As a result, different MPEG formats have many components in common, but typically no backward compatibility exists from one format to another.

Also, MPEG coders strictly specify both the bitstream syntax and decoder operation. Although the more recent standards offer greater flexibility

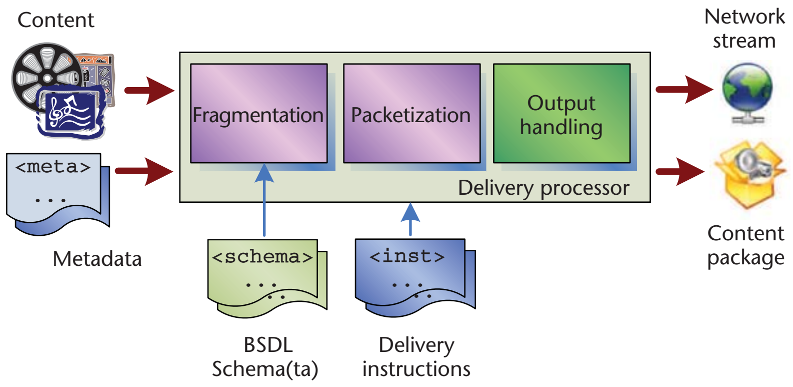

Figure 4. A format-independent multimedia streaming and delivery architecture uses BSDL to describe multimedia syntax.

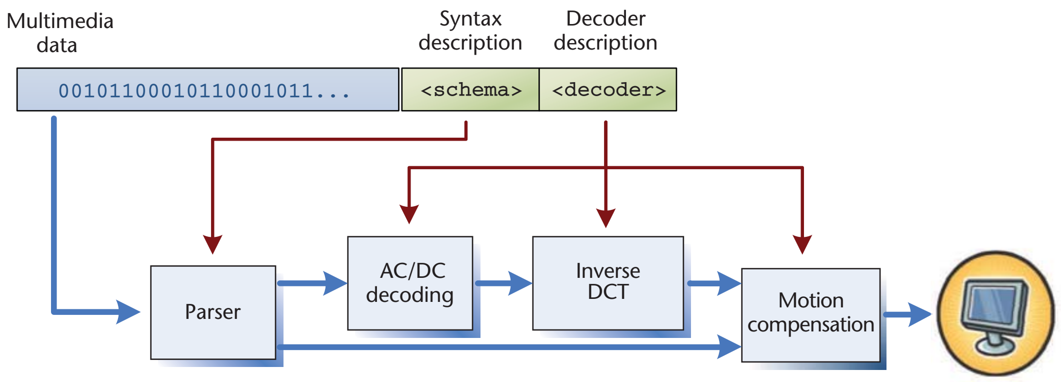

Figure 5. A BSDL Schema forms the syntax description of a reconfigurable bitstream.

through numerous optional features, they must still rigorously define the bitstream and decoder a priori to guarantee interoperability.

MPEG is currently working on *configurable media coding* (CMC), which is a new paradigm for multimedia coders.[5] With CMC, multimedia content becomes self-describing, in that neither the bitstream syntax nor the decoder are specified by any standard, but are instead provided as part of the bitstream itself. This approach drastically increases the flexibility available for media encoding. It lets users easily deploy new coding algorithms, without any requirement that the algorithms first undergo a lengthy standardization process. It also lets users substantially alter content's encoding to better meet the requirements of the transmission channel, receiving devices, or the media itself. Finally, CMC makes it straightforward to extend existing coding formats—for example, introducing new chroma sampling profiles or error-resilience mechanisms.

A BSDL Schema defines the syntax of the data in a CMC bitstream (see Figure 5). When the decoder receives the schema, it translates it into a parser block that will convert the raw content data into structured fields and objects that subsequent blocks will use. In the reference implementation currently under development, the decoder performs this translation using XSLT. However, CMC's normative specification will define the translation in abstract terms so implementers of the standard can tailor the process to their particular platform.

A CMC bitstream also contains a decoder description, which specifies the network of modules that make up the remainder of the decoder. A CMC bitstream can declare individual modules in one of two ways:

- It can draw modules from a library of blocks representing each of the atomic operations performed by existing MPEG decoders, such as AC/DC decoding, the Discrete Cosine Transform and its inverse (DCT/IDCT), or motion compensation (in the case of a video decoder).
- When existing blocks don't suffice, a CMC bitstream can define new blocks as part of the decoder description. It specifies these blocks in a language that can be directly synthesized into either hardware or software to instantiate the module.

As we discussed earlier, multimedia bitstreams are generally hierarchical in nature. In H.264/AVC, for example (Figure 2a), the video stream is composed of many network abstraction layer (NAL) units. Each NAL unit contains numerous header fields, such as a start code, and reference and type values. The CMC parser module uses the BSDL Schema to separate the raw bitstream into ordered structures (analogous to structs in the C programming language) that directly correspond to the syntax expressed by the Schema. Subsequent decoder blocks can access values in these structures (using the example in Figure 2) by statements such as

`NALUnit[n].startCode` or
`NALUnit[n].nalReference`

The `cmc:port` attributes within the Schema specify that the parser block detach instances of the structure on which they're declared from the structure tree and place them as tokens on the output port with the name corresponding to the attribute value. Downstream modules receive these tokens on one or more input ports and process their data. As a result, the relevant fragment of the BSDL Schema formally defines the interface at each module's ports.

Conclusion

Developing the code to parse and generate multimedia bitstreams has traditionally been a repetitive and error-prone task. It has also been an area of application development that defied the goal of software reuse. In contrast, BSDL abstracts the minutiae of bitstream parsing out of software code, into an interoperable data file (the BSDL Schema), allowing developers to concentrate on the functionality of their particular application.

BSDL's approach has demonstrated applications at numerous points in the multimedia delivery chain. In the future, this approach may be extended to still other processing tasks, such as transcoding and transmoding, or to types of binary data other than multimedia. **MM**

References

1. C. Timmerer and H. Hellwagner, "Interoperable Adaptive Multimedia Communication," *IEEE MultiMedia*, vol. 12, no. 1, 2005, pp. 74-79.
2. ISO/IEC, *21000-7 IT—Multimedia Framework (MPEG-21)—Part 7: Digital Item Adaptation*, Int'l Org. for Standardization/Int'l Electrotechnical Commission, 2007; http://www.iso.org (also see http://www.chiariglione.org/mpeg).
3. H. Kalva, "The H.264 Video Coding Standard," *IEEE MultiMedia*, vol. 13, no. 4, 2006, pp. 86-90.
4. ISO/IEC, *21000-18, IT—Multimedia Framework (MPEG-21)—Part 18: Digital Item Streaming*, Int'l Org. for Standardization/Int'l Electrotechnical Commission, 2007; http://www.iso.org.
5. ISO/IEC, "Working Draft 23002-4 Information Technology—MPEG Video Technologies—Part 4: Video Tool Library," Int'l Org. for Standardization/Int'l Electrotechnical Commission, 2007.
6. K. Diepold et al., "MPEG-A: Multimedia Application Formats," *IEEE MultiMedia*, vol. 12, no. 4, 2005, pp. 34.

Readers may contact Joseph A.I. Thomas-Kerr at joetk@elec.uow.edu.au.

Contact John R. Smith at jsmith@us.ibm.com.