



XML Technologies Dissected

Erik Wilde • Swiss Federal Institute of Technology, Zürich

The lack of well-defined information models in many XML technologies can generate compatibility problems and lower the odds of their long-term success.

XML technologies have become ubiquitous in applications for computer communications, business-to-business IT architectures, and Web-based services. They are evolving rapidly, with developer requirements fueling both the maturation of core technologies and the emergence of new ones. However, many of these XML technologies lack well-designed support, making life harder for developers and creating possible obstacles to ultimate success.

Here, I systematically examine the information and data models of several influential XML technologies. I define *information model* and *data model* in keeping with the terms discussed in RFC 3444.¹ In doing so, my hope is that the clarification that this RFC offers can both benefit XML technologies and help identify areas for future work.

Information and Data Models

I've adapted the following definitions to fit XML and a discussion of passive data structures; specifically, the definitions exclude the notion of operations, which is only relevant for modeling objects.

- An *information model* is an abstract description of a given application area's entities and their relations. The information model's goal is to identify, describe, and relate the target entities, rather than define a specific means for representing, accessing, or manipulating them.
- A *data model* can be regarded as a lower-level implementation of the higher-level information model. Data models are based on two things: the information model they're implementing and their particular implementation method. This method might define a syntax for serial-

izing the entities or an API for accessing the entities from within applications.

It's important to clearly distinguish between the properties of an information model and those of its data model. For example, I'm concerned here only with passive data structures. This might at times seem contradictory. The Document Object Model's application programming interface, for example, clearly provides operations for such things as adding attributes to elements. However, this is not a feature of the DOM *information model* of XML, which simply defines that elements can have any number of attributes. The DOM API as a *data model* provides functions for inserting attributes, but this is essentially the same as using the XML serialization data model (that is, an XML 1.0 document) and inserting the attribute using a text editor. In both cases, the information model behind the data model is modified to include a new attribute, using a mechanism that the particular data model enables.

XML Technologies

Researchers and developers have always hotly debated the question of whether an XML information model should be clearly defined. Given the success of many Internet technologies without an explicitly defined information model – including XML – an information model is clearly not required as a prerequisite for success. However, an information model is beneficial for the sake of proper layering, well-defined semantics, and the ease of defining new data models. Widely used XML technologies should also support at least two data models: one for representation, and another for program access through an API. Although

technologies can evidently thrive without meeting these requirements, such models offer benefits for extensibility and portability.

XML Documents

XML 1.0 basically defines a data model (a representation) without an explicit information model. Whether this is a boon or a bane is the subject of ongoing and lively discussions within various circles. XML 1.0 does contain an implicit information model (through its markup structures), but — as specification writers discovered when they attempted to define other XML 1.0 data models — there's no clear consensus about exactly what that information model is or should be. Because various communities hold different views of XML, there are now several XML information models.

Document Object Model. Developers introduced DOM for HTML, then adapted it to XML. Different versions implicitly defined different information models; only the latest version, DOM level 3,² includes enough information to make its information model a superset of the *XML information set*.

A DOM information model's data model includes DOM itself, which is available in many programming language mappings. The Simple API for XML (SAX; see www.saxproject.org) implements only a subset of DOM's information model, but developers can easily extend SAX2 to support the full DOM information model. The JDOM API, designed specifically for Java, also supports DOM's information model (see www.jdom.org).

DOM has no specific serialization data model. Some DOM implementations proposed a persistent DOM (PDOM) as a serialization format, but these formats were proprietary and never caught on. Developers can, however, use XML 1.0 document syntax as a serialization format.

XML Information Set. The XML Infoset (www.w3.org/TR/xml-infoset) is an attempt to define an XML information model. Although the infoset is the foundation for other information models (most notably XPath and XML Schema), it omits information from XML documents that is indispensable in some application scenarios — most notably, information about an XML document's entity structure and about CDATA sections. Furthermore, only documents that are compliant with XML namespaces have an infoset.

Data models for the infoset are DOM3 and SAX2. Despite subtle differences (such as the treatment of namespace attributes) all information available

through the infoset can be accessed through these models (although SAX2 requires extensions). Data models for infoset serialization are XML 1.0 and Canonical XML, which is the normalized encoding of an infoset as an XML 1.0 document.

XML Path Language 1.0. XPath 1.0 (www.w3.org/TR/xpath) is the foundation for several other XML technologies, in particular XSL Transformations (XSLT; see www.w3.org/TR/xslt) 1.0. XPath 1.0 uses a slight simplification of the infoset's information model and is defined in terms of infoset information items. Consequently, the XPath 1.0 information model excludes the same information as the infoset.

XPath 1.0's most popular data model uses XPath 1.0's path-based expression syntax. XPath 1.0 has been designed so that other specifications can extend it; XSLT 1.0 and the XML Pointer Language (XPointer), for example, extend XPath by adding constructs and functions. Developers can also use XPath 1.0 syntax with DOM3's XPath module. The only established data model for XPath 1.0 serialization is XML 1.0 itself, even though some XSLT processors can consume and produce SAX events, thus pipelining multiple XSLT processing steps without repeated serialization and parsing.

XML Schema. XML Schema (www.w3.org/xml/schema) is a rather complex schema language that uses Post Schema Validation infoset (PSVI) contributions to extend the infoset with information from XML Schema processing. Formally, XML Schema defines XML Schema processing in terms of infoset augmentation, rather than as processing an XML document. Consequently, the PSVI lacks the same information that is missing in the infoset.

Unfortunately, there is no agreed-upon data model for the PSVI part of XML Schema's information model. This is true for representations as well as for APIs. For a representation, it would be necessary to represent the considerable amount of PSVI information, and there are some examples for doing this using XML 1.0 syntax, but these have not caught on. On the API side, there are only proprietary proposals. Although DOM3's modular structure would be ideally suited to define a PSVI module, the DOM working group has yet to support PSVI access.

XML Query Language 1.0 and XML Path Language 2.0. For XQuery 1.0 (www.w3.org/TR/xquery) and XPath 2.0, developers had to define a new information model (because XQuery developers wanted

to add data-type support). They're still building the model – which they unfortunately called “data model” in the specification – but it clearly builds on top of the XML Schema information model (that is, the infoset with PSVI augmentations) with some extensions, such as document fragments and atomic values. Basically, the XQuery 1.0 and XPath 2.0 information model is a stripped-down version of the XML Schema information model, with only partially visible PSVI information.

As for data models, the XQuery 1.0 and XPath 2.0 information model takes the same approach as the XPath 1.0 information model: it serves as a foundation for other specifications, including XQuery 1.0 and XSLT 2.0. Perhaps later versions of DOM will include an XPath 2.0 module. For serialization, the same limitations apply as those for the XML Schema information model: developers can easily serialize

in the abstract questions behind the current diversity of XML information and data models, it's important that they keep them in mind. The fact that different XML technologies are built on different information models directly influences how a developer views and processes XML documents.

XML Schema Languages

Developers use XML schema languages to express XML document constraints. In many cases, schema languages are grammar-based – such as Document Type Definitions and XML Schema – but other approaches have proven useful for certain application scenarios (such as the rule-based Schematron language, used for expressing XPath-based constraints). In the following list, I focus on the schema languages' information and data models, rather than on how validation might affect the validated document.

Document Type Definition. The XML 1.0 specification implicitly defines the DTD information model, and XML 1.0 syntax is the only officially defined DTD data model. Unfortunately, there is no standardized data model for accessing DTDs through an API, even though DOM and SAX can access them with proprietary extensions.

XML Schema. XML Schema is defined using a component-based information model. The XML Schema specification specifies the components (such as simple and complex types), their possible relations, and their semantics. The specification also defines an XML syntax for component serialization, which is XML Schema's serialization data model. XML Schema has no standardized API data model, but there are proprietary DOM extensions that provide access to XML Schema components.

Relax NG. Developers wanted Relax NG (www.relaxng.org) to be as simple as possible, so it doesn't define an explicit information model. It also lacks an API data model, making it impossible to gain model-based access to Relax NG schemas from within applications. On the representation side, Relax NG defines two data models: the initially defined XML-based syntax, and a non-XML compact syntax,³ which is much easier for users to read and write than the XML variant.

Discussion. Although XML Schema is still criticized, the fact that both XQuery 1.0 and XPath 2.0 are based on XML Schema's information model will certainly help promote it. Relax NG promoters point

4-line pull quote here

the infoset part as XML 1.0, but the additional information (a subset of XML Schema's PSVI contributions) has no agreed-upon serialization format.

Discussion. Although the question of whether XML is primarily a syntax or an information model representation remains open, it's clear that to define an XML data model (and thus provide access to XML beyond XML 1.0's own syntax) we must make assumptions about the underlying information model. Furthermore, building XML architectures out of preexisting components affects the information model because individual components might support different models. For example, if developers use an XSLT component, any CDATA sections will disappear because they're not part of XSLT's (that is, XPath 1.0's) information model. To avoid this, they could use a proprietary version of XSLT that includes CDATA in its information model. Another option would be to map CDATA sections to markup structures that XSLT's information model preserves (such as elements, processing instructions, or comments), and then remap them after XSLT processing.

Regardless of whether developers are interested

to its simplicity and the fact that it does not change the infoset. In the long run, however, Relax NG's success is more likely if developers can show how to obtain an XQuery 1.0 and XPath 2.0 information model using Relax NG schema validation. (With this information model, the type information is key; because Relax NG supports data type libraries, it might be possible – for simple types, at least – to use Relax NG to augment an infoset with XQuery 1.0 and XPath 2.0 information model types.)

XML Formatting

Currently, XML document formatting depends on two technologies: Cascading Style Sheets and the Extensible Stylesheet Language. In regards to formatting, there are three distinct models: the model of the source document (the document to be formatted), the model of the formatting mechanism, and the model of the formatted result.

Cascading style sheets. Originally created for HTML, CSS (see www.w3.org/style/css) has been around for a while. CSS works by defining *selectors*, which developers can use to identify certain document parts, and *declarations*, which developers use to assign property values. These properties are then used to format the document. A number of modules in the current version, CSS3, define properties for several different areas (such as fonts, tables, and colors). CSS3's source-document information model is defined rather vaguely, but it's based on the assumption that CSS is used for XML documents. CSS3 would certainly benefit from a more precisely defined foundation – most likely the infoset, perhaps with extensions to cater to pseudo-classes and other dynamic aspects.

DOM2's style specification provides data model support for CSS, but the specification is built on CSS2 and does not fully support CSS3. Developers planned to provide generic style sheet support in DOM3, with specific CSS support in an additional module, but have yet to produce results. As for serialization data models, CSS defines its own non-XML syntax. CSS syntax is easily human-readable and more compact than XML, but XML-based software cannot process it.

Extensible Stylesheet Language. While CSS simply selects document parts, XSL (www.w3.org/style/xsl) defines a two-step process. It first defines a dedicated language, XML Transformations (XSLT), which transforms a source document into a tree of XSL formatting objects (XSL-FO).

The XSL model's advantage over CSS is that it lets users modify documents structurally. The transformation step is conceptually independent from the actual formatting, which users produce by rendering a presentation from the XSL-FO document.

The source information model of XSLT and XSL-FO is the XPath 1.0 information model (which is the XSLT stage's input and output). XSLT and XSL-FO currently lack an API data model, and there are no plans to provide one with DOM3. The serialization data model is XML: the XSLT step of the formatting process produces XSL-FO, and the XSL-FO formatter's result is entirely implementation-dependent.

Discussion. CSS and XSL have much in common. Although browsers don't yet support XSL (Internet Explorer and Mozilla support only XSLT), it would be good to unify CSS and XSL-FO, or at least base them on a shared underlying model. CSS3's box model and XSL-FO's area model are technically aligned, as are many (but not all) properties of CSS3 and XSL-FO. If developers unified both technologies' visual formatting models, implementers could use the same formatting code to support CSS3 and XSL-FO; the only difference would be how source documents were processed (that is, how the system processed selectors for CSS3 and transformations for XSL).

XML Hyperlinking

The XML Linking Language (XLink; see www.w3.org/XML/linking) has been defined as a way to embed hypermedia links in XML documents. It extends HTML's link model in several ways, most notably by allowing multiended and external links. The XLink specification defines its syntax in terms of a namespace and several global attributes; it defines the information model only implicitly.

XLink's only available data model is its XML-based serialization. Having DOM and CSS modules for XLink would be quite useful: DOM could provide application access, and CSS could provide link formatings. So far, however, neither DOM nor CSS plan to add XLink support.

XLink is an interesting case study of an information models' usefulness. Because XLink's serialization does not work well with XHTML's requirements, XHTML has defined HLink, its own linking mechanism,⁴ rather than pushing for an XLink information model definition and defining an alternative serialization data model for it. So, implementers must now support both XLink and

Write for Spotlight

Spotlight focuses on emerging technologies, or new aspects of existing technologies, that will provide the software platforms for Internet applications. Spotlight articles describe technologies from the perspective of a developer of advanced Web-based applications. Articles should be 2,000 to 3,000 words. Guidelines are at www.computer.org/internet/dept.htm.

Contact department editor Siobhán Clarke at siobhan.clarke@cs.tcd.ie.

HLink as two different (yet similar) linking mechanisms, rather than as two representations for identical underlying link semantics.

Conclusion

XML technologies are successful and rapidly expanding. Despite the undisputable advantages of all the technologies mentioned here, however, a more systematic approach might benefit XML technology development. At very least, it might help developers identify areas in which harmonization is possible.

Most promising, particularly from a developer's viewpoint, would be to create DOM3 modules that support DTDs, XML Schema components, PSVI, and XLink. There are some activities, but for other technologies, the current lack of ready-to-use data models (APIs in particular), might prove a considerable obstacle to long-term success.

The September 2003 W3C workshop on binary XML infosets will probably heat up the debate surrounding the information versus data model question, and might serve as an indication about future directions of the foundations underlying XML technologies. □

References

1. A. Pras and J. Schönwälder, "On the Difference between Information Models and Data Models," IETF RFC 3444, Jan. 2003; www.rfc-editor.org/rfc/rfc3444.txt.
2. A. Le Hors et al., "Document Object Model (DOM) Level 3 Core Specification," W3C working draft, June 2003; www.w3.org/TR/2003/WD-DOM-Level-3-Core-20030609.
3. J. Clark, "RELAX NG Compact Syntax," Organization for the Advancement of Structured Information Standards (Oasis), committee specification, Nov. 2002; www.oasis-open.org/committees/relax-ng/compact-20021121.html.
4. S. Pemberton and M. Ishikawa, "HLink: Link Recognition for the XHTML Family," W3C working draft, Sept. 2002; www.w3.org/TR/hlink.

Erik Wilde is a researcher and lecturer at the Computer Engineering and Networks Laboratory at the Swiss Federal Institute of Technology, Zürich. He received his PhD from the Swiss Federal Institute of Technology, Zürich. Contact him at net.dret@dret.net.