XML SCHEMA LANGUAGES COMPARED

Eric van der Vlist (vdv@dyomedea.com)

XML schema languages compared

XML Prague 2005

May 2005

Copies : http://dyomedea.com/papers/2005-xmlprague

WHAT'S A XML SCHEMA LANGUAGE?

Not what you're expecting!

The name “XML schema” language is misleading and they are not what you'd expect them in plain English.

Not always a representation of the class instances

In plain English, a schema is a representation of an object, however, some XML schema languages do not try to represent XML instance documents.

Not simpler than the documents

In plain English, a schema is a simpler representation of an object, however, XML schemas are usually more complicated than the XML instance documents.

XML schemas are:

If they're not schemas, what can they be?

Filters or firewalls

They can be considered as filters or firewalls that protect applications from the wide diversity of XML documents.

Transformations producing a validation report and/or an augmented infoset

XML schema languages can be seen as transformations taking instance documents as their input and producing a validation report and, in some cases, an infoset augmented with the information (type information, default values, ...) gathered during the validation.

Two different approaches

Firewalls too can be either open of closed.

Everything which is not forbidden is allowed

For open schemas (or firewalls), everything which is not forbidden is allowed. An open schema is thus a list of rules defining what's forbidden.

Or everything which is not allowed is forbidden

For closed schemas, on the contrary, everything which is not allowed is forbidden. A closed schema is thus a list of rules defining what's allowed.

In practice, both are needed

In real world applications, we need to find a balance between openness and closeness. This is often done by opening closed schemas to add some of the extensibility which gives its name to XML, the eXtensible Markup Language.

1 XML DOCUMENT OUT OF 10 CONTAINS AT LEAST 1 ERROR

Figure given by two independent sources

This figure is frightening and it comes from two different authors working in two different domains.

Social security in Belgium

Reference: Isabelle Boydens - "Informatique, normes et temps". Bruxelles : Bruylant, 1999.

Banking systems (UK)

Reference: Simon Riggs, XML Europe 2003.

Can we afford this error rate?

Probably not!

Validation is not optional

If validation can decrease this rate we can't afford to ignore it and must do our best to improve its efficiency.

What kind of validation?

Decreasing the error rate in XML documents isn't an easy job and the nature of the tests to perform is very diverse.

Structure (imbrication of elements and attributes)

A first type of validation consists in checking the structure, ie the imbrication of elements and attributes. This validation acts at markup level and do not test the content of the text nodes or attributes.

Datatypes

A second category of validation consists in checking the content of text nodes and attributes independently of each other. With the exception of qualified names (QNames) in the content, an unfortunate practice that creates a dependency between the markup and the content, datatype validation ignores the markup and tests only the content of the document.

Integrity constraints

A third category of validation consists in checking identifiers to verify that they are unique and references to check that they refer to existing identifiers. Integrity constraints may be performed internally to a document or between documents (link checking).

Business rules

What's left after these three categories is often called business rules. Business rules can be as simple as checking that a date of death is, when it exists, greater than the date of birth or as complex as spell checking.

Short (and incomplete) history of schema languages

A long list starting before XML was invented...

Schema languages do not start with XML and, even though the clean distinction between features that are specific to validation and other features is relatively new, SGML had already its own schema languages.

DTD family

This first family takes its roots from SGML.

W3C XML Schema family

XML-Data (W3C note, January 98 by people from Microsoft, DataChannel, Arbortext, Inso Corporation and University of Edinburgh.)

Includes (most of) the basic concepts developed by W3C XML Schema

+ internal and external entity definitions

+ the mapping with RDF and object oriented structures.

XML-Data-Reduced (XDR) (W3C note, July 98, by editors from Microsoft and University of Edinburgh)

Goal: "refine and subset those ideas down to a more manageable size in order to allow faster progress toward adopting a new schema language for XML"

Implemented by Microsoft (Biztalk).

Document Content Description (DCD) (W3C note, July 98, by editors from Textuality, Microsoft and IBM)

"subset of the XML-Data Submission ... expresses it in a way which is consistent with the ongoing W3C RDF (Resource Description Framework) [RDF] effort".

No mapping but consistency with RDF.

Schema for Object-Oriented XML (SOX)

(W3C Note, September 98 (2nd vers. July 99), by Veo Systems/Commerce One)

Very influenced by object oriented design (concepts of interface and implementation).

Influenced by the DTDs (parameters)

Widely used by Commerce One.

Document Definition Markup Language (DDML or Xschema) (W3C Note, January 99, XML-DEV edited by Ronald Bourret, John Cowan, Ingo Macherius and Simon St.Laurent)

"encodes the logical (as opposed to physical) content of DTDs in an XML document"

Great attention paid to back and forward conversions back between DTD

"Experimental chapter proposing Inline DDML elements

Clear distinction between structures and data (left apart).

W3C XML Schema (W3C Recommendation, May 2001)

Acknowledges the influence of DCD, DDML, SOX, XML-Data and XDR

Picked pieces from each of these proposals (compromise)

Proponents of living ancestors have all announced their support

Should become the only living member of this family.

RELAX FAMILY

RELAX (REgular LAnguage description for XML) (Published in March 2000 as a Japanese ISO Standard by MURATA Makoto)

Simple ("Tired of complicated specifications? You just RELAX !)

Solid (adaptation of the hedge automaton theory to XML trees

Approved as an ISO/IEC Technical Report in May 2001.

XDuce (pron. "transduce") (First announced in March 2000)

Not a schema but a programming language

Its typing system has influenced schema languages

TREX (Tree Regular Expressions for XML) (James Clark, January 2001)

"basically the type system of XDuce with an XML syntax and with a bunch of additional features

A TREX schema reads almost as plain English

Consistent treatment between elements and attributes

RELAX NG (May 2001-December 2001, OASIS TC specification)

Now an ISO/IEC Standard (DSDL Part 2)

Merge between RELAX and TREX

Coedited by James Clark and MURATA Makoto

Schematron

(proposed in September 1999 by Rick Jelliffe)

Validation rules using XPath expressions.

Examplotron (proposed in March 2001 by the author of this presentation)

Example of instance structures used as schemas

Borrows to Schematron and RELAX NG.

Document Schema Definition Languages (DSDL)

Too complex for a single language

Standing for "Document Schema Definition Languages", DSDL is a recognition that the validation of XML documents is a subject too wide and complex to be covered by a single language and that the industry needs a set of simple and dedicated languages to perform different validation tasks and a framework in which these languages may be used together.

Part 1: Overview

This is a kind of roadmap describing DSDL itself and introducing each of the parts.

Part 2: Regular-grammar-based Validation

This part is RELAX NG itself. It is a rewriting of the Relax NG Oasis Technical Committee specification to meet the requirements of ISO publications. Its wording is more formal than the Oasis specification but the features of the language is the same and any RELAX NG implementation conform the one of these two documents should also be conform to the other.

DSDL Part 2 is now a "Final Draft International Standard" (FDIS), i.e. an official ISO standard.

Part 3: Rule-based Validation

This part will describe the next release of the rule based schema language known as Schematron. Schematron has been defined by Rick Jelliffe and other contributors and its home page is http://www.ascc.net/xml/schematron/.

Part 4: Selection of Validation Candidates

Although Relax NG provides a way to write and combine modular schemas, it is often the case that you need to validate a composite document against existing schemas which can be written using different languages: you may want for instance to validate XHTML documents with embedded RDF statements. In this case, you need to split your documents into pieces and validate each of these pieces against its own schema.

Part 5: Datatypes

The goal of this part is to define a set of primitive datatypes with their constraining facets and the mechanisms to derive new datatypes from this set and it is fair to say that it's probably the least advanced and more complex part of DSDL.

Part 6: Path-based Integrity Constraints

The goal of this part is basically to define a feature covering integrity constraints.

Part 7: Character Repertoire Validation

This part will allow to specify which characters may be used in specific elements and attributes or within entire XML documents.

Part 8: Declarative Document Architectures

This part is still the most mysterious to me. The idea here is to allow to add information to documents (such as default values) depending on the structure of the document and the only input considered for Part 8 so far is known as "Architectural Forms", an old promising-but-never-used-that-much technology.

Part 9: Namespace and Datatype-aware DTDs

There were plenty of good things in DTDs, especially in SGML DTDs and many people are still using them and do challenge the need to put them to trash and define new schema languages to support namespaces and datatypes. DSDL Part 9 is for these people who would like to rely on years of usage of DTDs without loosing all of the goodies of newer schema languages. Despite a burst of discussion in April 2002, this part hasn't really advanced yet.

Part 10: Validation Management

Least but not last, Part 10 (formerly known as Part 1: Interoperability Framework) is the cement which will let you use together the different parts from DSDL together with external tools such as XSLT, W3C XML Schema or your favourite spell checker to come back to an example given in the introduction to this chapter.

SAMPLE DOCUMENT

The following example will be used during the presentation:

```
<?xml version="1.0"?>
<library>
  <book id="_0836217462">
    <isbn>
      0836217462
    </isbn>
    <title>
      Being a Dog Is a Full-Time Job
    </title>
    <author-ref id="Charles-M.-Schulz"/>
    <character-ref id="Peppermint-Patty"/>
    <character-ref id="Snoopy"/>
    <character-ref id="Schroeder"/>
    <character-ref id="Lucy"/>
  </book>
  <book id="_0805033106">
    <isbn>
```

```
    0805033106
  </isbn>
  <title>
    Peanuts Every Sunday
  </title>
  <author-ref id="Charles-M.-Schulz"/>
  <character-ref id="Sally-Brown"/>
  <character-ref id="Snoopy"/>
  <character-ref id="Linus"/>
  <character-ref id="Snoopy"/>
</book>
<author id="Charles-M.-Schulz">
  <name>
    Charles M. Schulz
  </name>
  <nickName>
    SPARKY
  </nickName>
  <born>
    November 26, 1922
  </born>
  <dead>
    February 12, 2000
  </dead>
</author>
<character id="Peppermint-Patty">
  <name>
    Peppermint Patty
  </name>
  <since>
    Aug. 22, 1966
  </since>
  <qualification>
    bold, brash and tomboyish
  </qualification>
</character>
<character id="Snoopy">
  <name>
    Snoopy
  </name>
  <since>
    October 4, 1950
  </since>
  <qualification>
    extroverted beagle
  </qualification>
</character>
<character id="Schroeder">
  <name>
```

```
      Schroeder
    </name>
    <since>
      May 30, 1951
    </since>
    <qualification>
      brought classical music to the Peanuts strip
    </qualification>
  </character>
  <character id="Lucy">
    <name>
      Lucy
    </name>
    <since>
      March 3, 1952
    </since>
    <qualification>
      bossy, crabby and selfish
    </qualification>
  </character>
  <character id="Sally-Brown">
    <name>
      Sally Brown
    </name>
    <since>
      Aug, 22, 1960
    </since>
    <qualification>
      always looks for the easy way out
    </qualification>
  </character>
  <character id="Linus">
    <name>
      Linus
    </name>
    <since>
      Sept. 19, 1952
    </since>
    <qualification>
      the intellectual of the gang
    </qualification>
  </character>
</library>
```

FACT SHEET: DTD

DTD: Basic facts

Author: W3C.

Status: Recommendation (XML 1.0).

Page: http://www.w3.org/TR/REC-xml

PSVI: yes.

Structure: yes.

Datatype: yes (weak).

Integrity: local with restrictions (ID/IDREF).

Other rules: no

Vendor support: excellent

Notes

Specific syntax.

No support for namespaces.

No support for content-sensitive content models.

Include other features (entities).

Reference to the DTD needs to be included in the instance document.

SAMPLE DTD

DTD for our example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT author (name, nickName, born, dead)>
<!ATTLIST author
  id ID #REQUIRED
>
<!ELEMENT author-ref EMPTY>
<!ATTLIST author-ref
  id IDREF #REQUIRED
>
<!ELEMENT book (isbn, title, author-ref*, character-ref*)>
<!ATTLIST book
  id ID #REQUIRED
>
<!ELEMENT born (#PCDATA)>
<!ELEMENT character (name, since, qualification)>
<!ATTLIST character
  id ID #REQUIRED
>
<!ELEMENT character-ref EMPTY>
<!ATTLIST character-ref
  id IDREF #REQUIRED
>
<!ELEMENT dead (#PCDATA)>
<!ELEMENT isbn (#PCDATA)>
<!ELEMENT library (book+, author*, character*)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT nickName (#PCDATA)>
<!ELEMENT qualification (#PCDATA)>
<!ELEMENT since (#PCDATA)>
<!ELEMENT title (#PCDATA)>
```

FACT SHEET: W3C XML SCHEMA

W3C XML Schema: Basic facts

Author: W3C.

Status: Recommendation.

Page: http://www.w3.org/TR/xmlschema-0/

PSVI: yes.

Structure: yes.

Datatype: yes.

Integrity: local (ID/IDREF, key, keyref).

Other rules: no

Vendor support: potentially excellent- still immature.

Notes

Paranoiac about determinism.

Borrows ideas from OO design.

Considered complex.

Presented as a foundation of XML.

Controversial way of dereferencing namespace URIs.

SAMPLE W3C XML SCHEMA

W3C XML Schema for our example:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xml="http://www.w3.org/XML/1998/namespace">
  <xs:element name="library">
```

```
    <xs:complexType>
      <xs:sequence>
        <xs:element name="book" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="isbn" type="xs:string"/>
              <xs:element name="title" type="xs:string"/>
              <xs:element name="author-ref" minOccurs="0"
                maxOccurs="unbounded">
                <xs:complexType>
                 <xs:attribute name="id" type="xs:IDREF"
                   use="required"/>
                </xs:complexType>
              </xs:element>
              <xs:element name="character-ref" minOccurs="0"
                maxOccurs="unbounded">
                <xs:complexType>
                 <xs:attribute name="id" type="xs:IDREF"
                   use="required"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute name="id" type="xs:ID" use="required"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="author" minOccurs="0"
maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="name"/>
              <xs:element name="nickName" type="xs:string"/>
              <xs:element name="born" type="xs:string"/>
              <xs:element name="dead" type="xs:string"/>
            </xs:sequence>
            <xs:attribute name="id" type="xs:ID" use="required"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="character" minOccurs="0"
          maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="name"/>
              <xs:element name="since" type="xs:string"/>
              <xs:element name="qualification" type="xs:string"/>
            </xs:sequence>
            <xs:attribute name="id" type="xs:ID" use="required"/>
```

```
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="name" type="xs:string"/>
</xs:schema>
```

FACT SHEET: RELAX NG

RELAX NG: Basic facts

Author: OASIS RELAX NG TC & ISO DSDL WG

Status: OASIS Specification, ISO/IEC Standard

Page: http://relaxng.org/

PSVI: no.

Structure: yes.

Datatype: pluggable.

Integrity: through keys and key references.

Other rules: no

Vendor support: improving.

Notes

Merge between RELAX and TREX: TREX based syntax including the full semantics of RELAX.

Two syntaxes are available

SAMPLE RELAX NG (XML SYNTAX)

RELAX NG Schema for our example:

```
<?xml version="1.0"?>
```

```
<grammar datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"
  xmlns:="http://relaxng.org/ns/structure/1.0"
  xmlns:xml="http://www.w3.org/XML/1998/namespace">
  <start>
      <ref name="library"/>
  </start>
  <define name="library">
    <element name="library">
      <oneOrMore>
        <ref name="book"/>
      </oneOrMore>
      <zeroOrMore>
        <ref name="author"/>
      </zeroOrMore>
      <zeroOrMore>
        <ref name="character"/>
      </zeroOrMore>
    </element>
  </define>
  <define name="author">
    <element name="author">
      <attribute name="id">
        <data type="ID"/>
      </attribute>
      <element name="name">
        <text/>
      </element>
      <element name="nickName">
        <text/>
      </element>
      <element name="born">
        <text/>
      </element>
      <element name="dead">
        <text/>
      </element>
    </element>
  </define>
```

```
 <define name="book">
   <element name="book">
     <ref name="id-attribute"/>
     <ref name="isbn"/>
     <ref name="title"/>
     <zeroOrMore>
       <element name="author-ref">
         <attribute name="id">
           <data type="IDREF"/>
         </attribute>
         <empty/>
       </element>
     </zeroOrMore>
     <zeroOrMore>
       <element name="character-ref">
         <attribute name="id">
           <data type="IDREF"/>
         </attribute>
         <empty/>
       </element>
     </zeroOrMore>
   </element>
 </define>
 <define name="id-attribute">
   <attribute name="id">
     <data type="ID"/>
   </attribute>
 </define>
 <define name="character">
   <element name="character">
     <ref name="id-attribute"/>
     <ref name="name"/>
     <ref name="since"/>
     <ref name="qualification"/>
   </element>
 </define>
 <define name="isbn">
   <element name="isbn">
     <text/>
   </element>
 </define>
 <define name="name">
   <element name="name">
     <text/>
   </element>
 </define>
```

```
  <define name="nickName">
    <element name="nickName">
      <text/>
    </element>
  </define>
  <define name="qualification">
    <element name="qualification">
      <text/>
    </element>
  </define>
  <define name="since">
    <element name="since">
      <data type="date"/>
    </element>
  </define>
  <define name="title">
    <element name="title">
      <text/>
    </element>
  </define>
</grammar>
```

SAMPLE RELAX NG (COMPACT SYNTAX)

The same RELAX NG Schema using the compact syntax:

```
start = library
library = element library { book+, author*, character* }
author =
  element author {
    attribute id { xsd:ID },
    element name { text },
    element nickName { text },
    element born { text },
    element dead { text }
  }
book =
  element book {
    id-attribute,
    isbn,
    title,
    element author-ref {
```

```
      attribute id { xsd:IDREF },
      empty
    }*,
    element character-ref {
      attribute id { xsd:IDREF },
      empty
    }*
  }
id-attribute = attribute id { xsd:ID }
character =
  element character { id-attribute, name, since, qualification }
isbn = element isbn { text }
name = element name { text }
nickName = element nickName { text }
qualification = element qualification { text }
since = element since { xsd:date }
title = element title { text }
```

FACT SHEET: SCHEMATRON

Schematron: Basic facts

Author: Rick Jelliffe and other contributors.

Status: unofficial, will become ISO DSDL Part 3

Page: http://www.ascc.net/xml/schematron/

PSVI: sort of (validation report can be customized).

Structure: not directly.

Datatype: not directly.

Integrity: not directly.

Other rules: yes

Vendor support: improving.

Notes

Rule expressions only.

SAMPLE SCHEMATRON

Schematron Schema for our example (partial):

```
<?xml version="1.0"?>
<sch:schema xmlns:sch="http://www.ascc.net/xml/schematron"
  xmlns:xml="http://www.w3.org/XML/1998/namespace">
  <sch:title>
    Schematron Schema for library
  </sch:title>
  <sch:pattern>
    <sch:rule context="/">
      <sch:assert test="library">
        The document element should be "library".
      </sch:assert>
    </sch:rule>
    <sch:rule context="/library">
      <sch:assert test="book">
        There should be at least a book!
      </sch:assert>
      <sch:assert test="not(@*)">
        No attribute for library, please!
      </sch:assert>
    </sch:rule>
    <sch:rule context="/library/book">
      <sch:assert test="not(following-sibling::book/@id=@id)">
        Duplicated ID for this book.
      </sch:assert>
      <sch:assert test="@id=concat('_', isbn)">
        The id should be derived from the ISBN.
      </sch:assert>
    </sch:rule>
    <sch:rule context="/library/*">
      <sch:assert test="name()='book' or name()='author' or
        name()='character'">
        This element shouldn't be here...
      </sch:assert>
    </sch:rule>
  </sch:pattern>
</sch:schema>
```

EMBEDDED LANGUAGES

Schematron is a good candidate to be embedded in other languages.

Schematron patterns can be included in xs:appinfo W3C XML Schema.

An "experimental validator" does the same with RELAX NG

Schematron is a good fit since it doesn't overlap that much with the other languages.

TOOL SUPPORT

All these languages come with open source implementations that can be integrated in your applications.

Integrated support by tools and framework is (today):

Best: DTD

Most promising: W3C XML Schema

Challenger: RELAX, RELAX NG

Niche: TREX, Schematron, Examplotron

FEATURES

Features supported by the schema languages:

XML Structure: DTD, W3C XML Schema, RELAX, TREX, RELAX NG, Examplotron.

Datatypes: DTD (weak), W3C XML Schema.

Integrity: DTD, W3C XML Schema, RELAX, TREX, RELAX NG

Rules: Schematron, Examplotron

EXPRESSIVENESS

Ability to describe a wide range of structures

The ranking by expressiveness or flexibility (i.e. ability to describe a wide range of structures):

Most flexible: Schematron (but the authors need to define all the rules one by one).

Most flexible structural languages: TREX, RELAX NG, RELAX.

Challenger: EXAMPLOTRON

Behind: W3C XML Schema

Less flexible: DTD (no support for namespaces).

CONCLUSION

There is no perfect XML Schema language, choose the best fitted for the job you have to do!