Alternatives to XML Schema, XLink, XPointer

Patryk Czarnik

XML and Applications 2013/2014
Week 15 – 27.01.2014

Some drawbacks of XML Schema

- Verbose syntax, specification hard to understand
- Restricted power of expression
  - deterministic model required
  - no choice between:
    - text content and element content
    - attribute and element
  - content-aware model not available (in 1.0)
  - no value-aware constraints other than identity constraints (in 1.0)
- Extending types only by appending model fragments at the end of a sequence
  - no direct support for adding new elements to a choice
  - available with other techniques: element groups or substitution groups

Alternatives

- DTD – obviously
- RELAX NG (Regular Language for XML Next Generation)
  - by James Clark and Murata Makoto
  - OASIS (2001) and ISO (2003) standard
- Schematron
  - by Rick Jelliffe (1999), developed at Academia Sinica (Taiwan), ISO standard (2006)
- Examplotron
  - by Eric van der Vlist, project active in 2001-2003
- XML Schema 1.1
  - W3C Recommendation, 2012
  - borrows some ideas from the alternatives, mainly Schematron

Simple example in all standards

DTD

```
<!ELEMENT person (first-name+, last-name)>
<!ELEMENT first-name (#PCDATA)>
<!ELEMENT last-name (#PCDATA)>
```

XML Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="person" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="first-name" type="xs:string"
               maxOccurs="unbounded" />
            <xs:element name="last-name" type="xs:string" />
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

Simple example in all standards

Relax NG – XML syntax

```
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
   <start>
      <element name="person">
         <oneOrMore>
            <element name="first-name">
               <text/>
            </element>
         </oneOrMore>
         <element name="last-name">
            <text/>
         </element>
      </element>
   </start>
</grammar>
```

Relax NG – compact syntax

```
element person {
   element first-name { text }+
   element last-name { text }
}
```

Simple example in all standards

Schematron

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
   <pattern>
      <title>Person rules</title>
      <rule context="/person">
         <assert test="count(first-name) >= 1">Person
            should contain one or more first names.</assert>
      </rule>
      <rule context="/person">
         <assert test="count(last-name) = 1">Person
            should contain exactly one last name.</assert>
      </rule>
      <rule context="/person">
         <assert test="not(*[local-name() != 'first-name'
               and local-name() != 'last-name'])">Person
            should contain no other elements.</assert>
      </rule>
   </pattern>
</schema>
```

Note: This is not the primary intended use of Schematron

Simple example in all standards

Examplotron :)

```
<person>
   <first-name>Adam</first-name>
   <first-name>Maria</first-name>
   <last-name>Abacki</last-name>
</person>
```

Relax NG – basic ideas

- Clear theoretical basis: Tree automata with regular expressions specifying content in each node
  - The same model with appropriate restrictions is used by theoreticians to model DTD or XML Schema, but as a kind of “reverse engineering”
- Compact and readable XML syntax
  - Even more compact plain text syntax available
- Model components such as elements, attributes, or text nodes may be mixed together in definitions
- Modularisation available through define / ref mechanism
  - Equivalent to DTD parameter entities or XSD groups, but with some additional operations to enhance convenience
- No direct support for simple types, but referring to XML Schema types is possible.

Schematron – basic ideas

- Approach different from grammar-based DTD, XSD, and RelaxNG:
  - XPath expressions specify assertions that must hold for instance documents (and elements within them)
  - High power of expression
  - Less convenient (than grammar rules) to write structural definitions
- Official implementation:
  - translation of Schematron scheme to XSLT
  - XSLT evaluates expressions and report errors
- Other implementations available, also for Schematron rules embedded in XML Schema definitions
- XML Schema 1.1 covers most typical Schematron use cases. Probably XSD 1.1 will replace Schematron at all...

Non-deterministic model

- Ambiguous model:
  - It is not possible to state which particle of the model definition is matched, even if whole document is known.
  - example: (A,A,A)+|(A,A)+ for document AAAAAA
- Non-deterministic model:
  - It is not possible to determine the appropriate definition particle just seeing the start tag of an element, when a document is being read.
  - Some models may be determined just by definition rearrangement, e.g.: A, A?, A? → A, (A, A?)?
  - Some models may not,
    notable example: (A, B)*, A?
- XML Schema avoids indeterministic models!
  - DTD allows parsers to revoke indeterministic models too.

Model forbidden in XML Schema, valid in Relax NG

```
<list>
   <odd/>
   <even/>
   <odd/>
   <even/>
   <odd/>
   ...
</list>
```

```
<element name="list">
   <oneOrMore>
      <element name="odd"/>
      <element name="even"/>
   </oneOrMore>
   <optional>
     <element name="odd"/>
   </optional>
</element>
```

Choice between text and element

- We'd like to allow both formats:

```
<phone>1234567</phone>
<phone><cc>48</cc><ac>22</ac><main>123456</main><int>1313</int></phone>
```

- No possibility in XML Schema (other than mixed content)
- No problem in RelaxNG:

```
<element name="phone">
   <choice>
      <text/>
      <group>
         <optional><element name="cc"/></optional>
         <optional><element name="ac"/></optional>
         <element name="main"/>
         <optional><element name="int"/></optional>
      </group>
   </choice>
</element>
```

Choice between attribute and element

- Similarly as before, we want to allow both

```
<section title="Introduction">
  ...
```

and

```
<section>
  <title>What does <q>Be or not to be</q> mean in fact?</title>
  ...
```

Relax NG solution:

```
<element name="section">
   <choice>
      <attribute name="title"/>
      <element name="title">
         <ref name="text-model"/>
      </element>
   </choice>
   <ref name="text-model"/>
</element>
```

Value-aware constraints

- Relations and constraints other than equality

```
<order>
   <order-date>2014-01-20</order-date>
   <delivery-date>2014-01-25</delivery-date>
   <item>
      ...
      <value>199.90</value>
   </item>
   <item>
      ...
      <value>20</value>
   </item>
   <value>219.90</value>
</order>
```

Value-aware constraints – Schematron solution

```
<pattern>
   <title>Order</title>
   <rule context="order">
     <assert test="delivery-date >= order-date">
        Delivery is not possible before order.</assert>
   </rule>
   <rule context="order">
     <assert test="value = sum(item/value)">
        Order value must be equal to the sum of item values.</assert>
   </rule>
</pattern>
```

Value-aware constraints – XSD 1.1 solution

```
<xs:complexType name="Order">
   <xs:sequence>
      ...
      <xs:element name="order-date" type="xs:date"/>
      <xs:element name="delivery-date" type="xs:date"/>
      ...
      <xs:element name="item" type="OrderItem" maxOccurs="unbounded"/>
      <xs:element name="value" type="xs:decimal"/>
      ...
   </xs:sequence>
   <xs:assert test="delivery-date >= order-date">
   <xs:assert test="value = sum(item/value)">
</xs:complexType>
```

Content-aware model

- We want to choose one of models depending on the value of a field (attribute or element), e.g:

```
<order>
   ...
   <payment method="transfer">
      <due-date>2014-02-07</due-date>
   </payment>
</order>
```

```
<order>
   ...
   <payment method="card">
      <card-no>4400-1234-1234-1234</card-no>
      <owner>Adam Abacki</owner>
      <exp-date>2014-12-31</exp-date>
   </payment>
</order>
```

Value-aware constraints – XSD 1.1 solution

```
<xs:element name="payment" type="Payment">
  <xs:alternative test="@method = 'transfer'" type="TransferPayment" />
  <xs:alternative test="@method = 'card'"  type="CardPayment" />
  <xs:alternative test="not(@method = ('transfer', 'card', 'cash'))"
     type="xs:error" />
</xs:element>

<xs:complexType name="Payment">
...
```

- Expressible also in Schematron, but less natural (analogous to the first “person” example).

XPointer

- The standard defines addressing XML documents and their fragments using standard URI syntax:
  - http://www.sejm.gov.pl/ustawa.xml#def-las
- 3 W3C recommendations dated 2002-2003:
  - XPointer Framework http://www.w3.org/TR/xptr-framework/
  - XPointer element() Scheme http://www.w3.org/TR/xptr-element/
  - XPointer xmlns() Scheme http://www.w3.org/TR/xptr-xmlns/
  - XPointer xpointer() Scheme http://www.w3.org/TR/xptr-xpointer/
    - (neverending?) Working Draft

XPointer – xpointer scheme

- xpointer scheme allows to address elements using XPath:
  - http://www.sejm.gov.pl/ustawa.xml#xpointer(/art[5]/par[2])
- xmlns scheme adds namespace declarations to the above:
  - ustawa.xml#xmlns(pr=http://www.sejm.gov.pl/prawo) xpointer(/pr:art[5]/pr:par[2])

XPointer – element scheme

- Element carrying ID attribute with given value:
  - document.xml#element(def-las)
- Element with given position (absolute or relative to element carrying ID with given value):
  - document.xml#element(/1/4/3)
  - document.xml#element(def-las/2/3)
- Short syntax:
  - document.xml#def-las
  - document.xml#/1/4/3
  - document.xml#def-las/2/3

XInclude

- Including external XML documents (or their fragments) in another XML document.
- Similar to entities, but:
  - normal element markup, no special syntax,
  - no need to declare anything in DTD, nor to have DTD at all
- Main capabilities:
  - including complete documents (identified by URL) or their fragments (pointed by XPointer)
  - including XML tree (default) or raw text
  - defining content to be used in case of an error
- Supported by many parsers, including Java (JAXP).

XInclude – example

```
<recipe>
   <xi:include xmlns:xi="http://www.w3.org/2001/XInclude"
      href="salad.xml#xpointer(/recipe/title)">

      <xi:fallback>
         <error>No such recipe.</error>
      </xi:fallback>
    </xi:include>
</recipe>
```

XLink

- HTML links (`<a>`, `<img>`):
  - link two documents: link source and target
  - link source is always in the linking element
- XLink — an extended idea of linking:
  - link information represented in any element:
    - element name is not important
    - attributes coming from XLink namespace are
  - more than two ends of link (hyperlink → relation)
  - possibility to represent link outside linked resources
- Status:
  - historical roots: HyTime,
  - XLink 1.0 – W3C recommendation: 2001,
  - XLink 1.1 – current version (made official TR: May 2010).

Terminology

- **Resource** – any addressable unit of information or a service (file, program, query result).
- **Link** – a relation between participating resources, expressed explicitly with a linking element.
- **Arc** – information about traversal between labelled resources (in defined direction):
  - outbound arc – from a local resource to some external resource
  - inbound arc – from an external resource to some local resource
  - third party – between two external resources
- Note: a resource is regarded as remote when addressed by URI (even though it resides in the same document or linking element as the link which uses it).

Types of links

- Simple link:
  - is outbound
  - binds exactly two resources: a local one with an external one
  - contains exactly one arc between resources
- Extended link:
  - binds arbitrary number of local and external resources,
  - uses arcs to define methods of traversal between resources,
  - defines roles of participating resources,
  - defines roles of arcs.

Simple link – an example

```
<book xmlns:xlink="http://www.w3.org/1999/xlink">
   <author xlink:type="simple"
        xlink:href="http://www.example.com/
                    bookstore/authors/Cormen">Thomas H. Cormen</author>
   <title>Introduction to algorithms</title>
</book>
```

Extended link – an example

```
<family xlink:type="extended" xmlns:xlink="http://www.w3.org/1999/xlink">
   <person xlink:type="locator" xlink:href="joe.xml"
           xlink:label="parent" xlink:title="Joseph"/>
   <person xlink:type="locator" xlink:href="cathy.xml"
           xlink:label="parent" xlink:title="Katherine"/>
   <person xlink:type="locator" xlink:href="mikey.xml"
           xlink:label="child" xlink:title="Michael"/>
   <person xlink:type="locator" xlink:href="toya.xml"
           xlink:label="child" xlink:title="La Toya"/>
   <person xlink:type="locator" xlink:href="janet.xml"
           xlink:label="child" xlink:title="Janet"/>
   <link xlink:type="arc" xlink:from="parent" xlink:to="child"/>
</family>
```

Attributes in extended links

- **type** – role of the element in a link
  - simple | extended | locator | arc | resource | title | none
- **href** – URI of the external resource
- **role** – abstract identifier of the resource role (URI)
- **arcrole** – as above, but for an arc
- **title** – text label of the resource or arc
- **show** – presentation info: new | replace | embed | other | none
- **actuate** – activation info: onLoad | onRequest | other | none
- **label** – label used as identifier in from and to, not necessarily unique
- **from**, **to** – pointer (in an arc) for a certain resource label

Future of XLink

- Applications:
  - organization and association of resources even when no writing permission is granted
  - a new type of added value – link sets
- Scope:
  - local – link servers, link databases
  - Internet?
- Problems:
  - visualization of extended links
  - synchronization of links and resources (Internet)

Some popular XML applications

- Documents / text processing / publications:
  - DocBook
  - Text Encoding Initiative (TEI)
  - Darwin Information Typing Architecture (DITA)
  - Open Document (ODF, OASIS standard, ISO/IEC 26300)
  - Office Open XML (OOXML / OpenXML) (Ecma standard, ISO/IEC 29500)
- Metadata and knowledge representation:
  - Dublin Core
  - RDF
  - Topic Maps

Some popular XML applications

- Multimedia
  - Scalable Vector Graphics (SVG)
  - Mathematical Markup Language (MathML)
- Security
  - XML Signature
  - XML Encryption