

Session 55

XML – Tools and Trends

Schematron

Tim Bornholtz

Schema languages

- Many people turn to schema languages when they want to be sure that an XML instance follows certain rules
 - DTD
 - XML Schema
 - Relax NG

What is wrong with that?

- These schema languages have very complicated grammars
- Not flexible enough to accurately reflect real business rules
- Often very difficult to map the business rules written in English to the technical rules defined by the Schema

What is Schematron

- Schematron is a language that allows you to directly express rules
 - Rules map closely to English
 - Reduce the need for ambiguous documentation that accompanies current XML Schemas

What is Schematron

- Has the ability to document the implied business rules
- Schematron works well with XML Schemas
 - Use Schematron to validate the things that XML Schemas can't do
 - Use a multi-phase approach to validating the XML

Why Schematron in Financial Aid?

- We trade many files with our partners and these are increasingly XML
- Schematron is more expressive than other schema languages
- Can handle more complicated dependencies that XML Schema cannot

Pre-requisites

- I'm going to assume that you're familiar with:
 - XML
 - XML Namespaces
 - XPath
 - XSLT

Components of a Schematron file

- The Schematron file is relatively simple and contains the following structure:
- `<schema>`
 - `<phase>` - top level construct
 - `<pattern>`
 - `<rule>` - defines the context
 - » `<assert>`
 - » `<report>`

<assert>

- Assert is the basic rule within a Schematron file to test for a valid condition
- <assert test="expression to evaluate">Message to display if the expression is false</assert>
- Any XSLT or XPath expression can be used including boolean logic and complicated formulas

<rule>

- Basic building block of a Schematron file.
<rule context="Location in the file"></rule>
 - <rule context="Award">
 - <rule context="Student">
- The context can be any valid XPath
 - XPath is a subset of XPath
 - XPath is the way to identify templates in XSLT
- All of the <assert> statements for a particular rule context are grouped together

<pattern>

- A <pattern> is a collection of related rules

<pattern name="Descriptive name of the rules">

- The rules do not need to work on the same elements
- The name of the pattern will be displayed in the output
- The name will help you identify which section of the document is failing the rules within the pattern

<schema>

- Root element of the document
- Schematron 1.5 must use the namespace:
<http://www.ascc.net/xml/schematron>
- ISO Schematron must use the namespace:
<http://purl.oclc.org/dsdl/schematron>

Very simple example

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
xmlns="http://www.ascc.net/xml/schematron">
  <title>Example 1</title>
  <pattern name="Document root">
    <rule context="/">
      <assert test="doc">Root element should
                          be "doc".</assert>
    </rule>
  </pattern>
</schema>
```

Running the examples

- There are implementations available for many languages: .Net, Java, Python, Perl, Ruby
- The 1.5 reference implementation compiles the Schematron file into a XSLT file that can be used against the XML instance document
- Most XSLT engines can be used with the reference implementation but the fastest are Saxon for Java and MSXSLT for .Net

Running the examples (Con't)

I'm using Saxon for Java but the process is similar for most XSLT processors

- 1) `java -jar saxon8.jar -o temp.xml file.sch schematron-basic.xml`
- 2) `java -jar saxon8.jar instance.xml temp.xml`
- 3) `rm temp.xml`

Basic assertions

- Schematron can validate many basic conditions
 - Presence of elements
 - Absence of elements
 - Sequence of elements within a complex element
 - Relative order of elements to each other
 - Validate for a certain number of elements

Presence or absence of tags

`<assert test="LoanType">`

- Require that the loan type be an element that is a direct child element of the current context

`<assert test="count(LoanType) = 0">`

- Require that the LoanType element is **not** present

`<assert test="count(*) = 5">`

- Require that there are exactly 5 child elements of the current context

Order of elements

```
<rule context="Student">
```

```
<assert test="following-  
sibling::*[1]/self::FFELPPLUS">
```

A "title" must be immediately followed by a
"subtitle"

```
</assert>
```

```
</rule>
```

Taking your validation further

- Schematron can do many things that XML Schemas cannot handle
 - Cross field relationships
 - Meaningful error messages
 - Conditional testing with phases

Cross Field Relationships

- XML Schemas can validate the contents and the data type of one element at a time
- Not able to easily validate relationships between elements
 - If field A contains a value X then field B must contain a value Y

PLUS Student cannot be Borrower

- `<rule context="FFELPPLUS">`
 `<assert test="Borrower/Index/SSN !=`
 `../Student/Index/SSN">Borrower for a PLUS`
 `loan must not be the student</assert>`
`</rule>`
- FFEPGradPLUS the borrower must be the student
- `<rule context="FFELPGradPLUS">`
 `<assert test="Borrower/Index/SSN =`
 `../Student/Index/SSN">Borrower for a Grad`
 `PLUS loan must be the student</assert>`
`</rule>`

Grad PLUS Grade Level

- If a student is getting a GradPLUS loan, they must be a graduate student

```
<rule context="FFELPGraduatePLUS">  
  <assert test="StudentLevelCode =  
    'FirstYearGraduate' or  
    StudentLevelCode='SecondYearGraduate' or  
    StudentLevelCode='ThirdYearGraduate' or  
    StudentLevelCode='FourthYearGraduate' or  
    StudentLevelCode='ContinuingGraduate'"> Student  
  must be a Graduate level for FFELPGradPLUS  
  award</assert>  
</rule>
```

Stafford Loan Borrower

- For a Stafford loan, the Borrower is the Student
- No Borrower section is sent (different than GradPLUS)

```
<rule context="FFELPStafford">
```

```
  <assert test="count(Borrower) = 0">
```

```
    Borrower section not necessary for Stafford  
loan</assert>
```

```
</rule>
```

<report>

- Use the <report> instruction to communicate information apart from the validation errors

<report test="condition">Message to display</report>

- The message will be displayed when the test condition is true
- <report test="@version="1.2.0">Validating version 1.2.0 of the XML</report>
- <report test="DocumentTypeCode">Document is of type <value-of/></report>

Meaningful messages

- Both `<assert>` and `<report>` allow you to display meaningful error messages
- The `<name/>` tag will display the actual element
 - `<assert test="Student">The <name/> element must exist</assert>`
- The `<value-of>` tag will display the actual value of the element.
 - `<report test="TransmissionData/Source/*/OrganizationName">File sent by <value-of/></report>`

Phases

- If we were to combine all of the business rules for CR:C into one schema it would be a very large file.
- A <phase> is a simple collection of patterns that are executed together.
- Example:
 - The CR:C rules can be split by loan type
 - Then based on that loan type only the appropriate rules are imported

Namespaces

- Schematron can be used to validate XML that uses namespaces
- Declare the namespace as a child of the <schema> element

<ns

uri="urn:org:pescc:message:CommonLineRequest:v1.1.0" prefix="req">

- Then use the namespace like normal

<assert test="count(req:LoanType) = 1">

Contact Information

We appreciate your feedback and comments.

I can be reached at:

Name: Tim Bornholtz

The Bornholtz Group

Phone: 540-446-8404

Email: tim@bornholtz.com

Web: <http://www.bornholtz.com>