ENVIRONMENT AGENCY AUSTRIA umweltbundesamt

schematron

A language for making assertions about patterns found in XML documents

Schematron Validation

For INSPIRE Air Quality Data

Introduction

- Correctly structured and complete data essential for further processing
- XSD Schemata from INSPIRE Data Specifications provide basic syntax
- However, XSD Validation:
  - permits diverse syntactical errors
  - does not cover constraints from the data model
  - does not perform semantic validation

XSD Validated Errors

Missing values for mandatory elements:

- <base:namespace/>

Codelist entries going nowhere:

- <am:zoneType xlink:href="WRONG"/>

Duplicate INSPIRE Ids

- <base:localId>Zone</base:localId>
- <base:namespace>AT.UBA.AQD</base:namespace>

Invalid date intervals:

- <gml:beginPosition>2004-06-14</gml:beginPosition>
- <gml:endPosition>2003-06-14</gml:endPosition>

Schematron

- Schematron: rule-based validation language
- Makes assertions about presence/absence of patterns in XML

> "A feather duster to reach the parts other schema languages cannot reach" - Rick Jelliffe – Schematron developer

- Rules defined in XML, transformed to XSLT
- Utilizes XPath, XSLT and XQuery functionality
- Defined in ISO/IEC FDIS 19757-3
- Unfortunately no IDE available

Schematron Pattern Syntax

```
<sch:pattern>
    <sch:title>Zone Element Check </sch:title>
    <sch:rule context="XPath Expression">
        <sch:let name="myVal" value="XPath"/>
        <sch:assert test="XPath Expression">
            Free Text What's Wrong
        </sch:assert>
        <sch:report test="XPath Expression">
            Free Text What's Right
        </sch:report>
    </sch:rule>
</sch:pattern>
```

Schematron Pattern Syntax

- Title describing rule

```
<sch:pattern>
    <sch:title>Zone Element Check </sch:title>
    <sch:rule context="XPath Expression">
        <sch:let name="myVal" value="XPath"/>
        <sch:assert test="XPath Expression">
            Free Text What's Wrong
        </sch:assert>
        <sch:report test="XPath Expression">
            Free Text What's Right
        </sch:report>
    </sch:rule>
</sch:pattern>
```

Schematron Pattern Syntax

- Rule context shows where we‘re checking

```
<sch:pattern>
    <sch:title>Zone Element Check </sch:title>
    <sch:rule context="XPath Expression">
        <sch:let name="myVal" value="XPath"/>
        <sch:assert test="XPath Expression">
            Free Text What's Wrong
        </sch:assert>
        <sch:report test="XPath Expression">
            Free Text What's Right
        </sch:report>
    </sch:rule>
</sch:pattern>
```

Schematron Pattern Syntax

- Assign values to variables with let

```
<sch:pattern>
    <sch:title>Zone Element Check </sch:title>
    <sch:rule context="XPath Expression">
        <sch:let name="myVal" value="XPath"/>
        <sch:assert test="XPath Expression">
            Free Text What's Wrong
        </sch:assert>
        <sch:report test="XPath Expression">
            Free Text What's Right
        </sch:report>
    </sch:rule>
</sch:pattern>
```

Schematron Pattern Syntax

- Assert with error message when something is wrong

```
<sch:pattern>
    <sch:title>Zone Element Check </sch:title>
    <sch:rule context="XPath Expression">
        <sch:let name="myVal" value="XPath"/>
        <sch:assert test="XPath Expression">
            Free Text What's Wrong
        </sch:assert>
        <sch:report test="XPath Expression">
            Free Text What's Right
        </sch:report>
    </sch:rule>
</sch:pattern>
```

Schematron Pattern Syntax

- Report shows what‘s right (we need positiv feedback! ;))

```
<sch:pattern>
    <sch:title>Zone Element Check </sch:title>
    <sch:rule context="XPath Expression">
        <sch:let name="myVal" value="XPath"/>
        <sch:assert test="XPath Expression">
            Free Text What's Wrong
        </sch:assert>
        <sch:report test="XPath Expression">
            Free Text What's Right
        </sch:report>
    </sch:rule>
</sch:pattern>
```

Abstract Schematron Patterns

- Allow for reuse of rules
- Can be parameterized
- Requires 2 additional attributes in sch:pattern tag:
  - **abstract**: for abstract patterns, this value is set to "true"
  - **id**: a unique identifier for this abstract pattern. Required for later instantiation.

Abstract Schematron Patterns

```
<sch:pattern abstract="true" id="PatternID">
    <sch:title>Zone Element Check </sch:title>
    <sch:rule context="XPath Expression">
        <sch:assert test="XPath Expression">
            Free Text What's Wrong
        </sch:assert>
        <sch:report test="XPath Expression">
            Free Text What's Right
        </sch:report>
    </sch:rule>
</sch:pattern>
```

```
<sch:pattern is-a="PatternID" id="MyPatternID">
    <sch:param name="ParamName" value="Value"/>
    <sch:param name="StringName" value="'StrValue'"/>
</sch:pattern>
```

Xpath Functions

Xpath Functions can be integrated into statements:

```
<sch:let name="date"
value="string(number(translate(./gml:timePosition, '-
:+TZ', ''))))"/>

<sch:assert test="((string-length($date)>0) and ($date!=
'NaN'))">
```

Xquery to select specific elements

Xquery can be used to select specific instances

```
<sch:let name="sampleFound"
value="/gml:FeatureCollection//aqd:AQD_Sample/aqd:inspireId/base:Identifier[base:localId = $sampleRef]" />


<sch:let name="inDictionary" value =
"$codelist/rdf:RDF/skos:Concept[contains(@rdf:about, $language)] " />
```

Xquery to select specific elements

Xquery can be used to select specific instances with complex criteria

```
<sch:let name="LTO_AOT40c_V" value=
"./aqd:assessmentThreshold/aqd:AssessmentThreshold/aqd:environmentalObject
ive/aqd:EnvironmentalObjective/aqd:objectiveType[@xlink:href =
'http://dd.eionet.europa.eu/vocabulary/aq/objectivetype/LTO']/../aqd:repor
tingMetric[@xlink:href =
'http://dd.eionet.europa.eu/vocabulary/aq/reportingmetric/AOT40c']/../aqd:
protectionTarget[@xlink:href =
'http://dd.eionet.europa.eu/vocabulary/aq/protectiontarget/V']/.."/>
```

Example: Check Existence

Assign content of XPath to variable

Check if variable exists and is longer than 0

```
<sch:let name="zoneContent" value=
"/gml:FeatureCollection/gml:featureMember/aqd:AQD_Zone"/>

<sch:assert test="($zoneContent and
       (string-length($zoneContent)>0))">
```

Can also check for multiple fields in one assert

Example: Check Correct Value

In certain cases, an element must always have a specific value

```
<sch:let name="media" value="ef:mediaMonitored"/>
<sch:assert
      test="($media and (compare($media, 'air') = 0))">
```

In other cases, one instance of the element must be provided with a specific value

```
<sch:let name="timeCoverage"
value="./om:resultQuality/gmd:DQ_DomainConsistency/gmd:result/gmd:DQ_ConformanceResult/gmd:explanation[gco:CharacterString = 'Time Coverage']"/>
<sch:assert test="string-length($timeCoverage)>0" >
```

Example: Check INSPIRE Id

In order to check if the INSPIRE Id is unique within the XML document, we search for all occurrences with the current ID and check to make sure that the total number is 1

```
<sch:let name="localId"
value="./base:Identifier/base:localId"/>
<sch:let name="namespace"
value="./base:Identifier/base:namespace"/>
<sch:let name="uniqueId"
value="count(/gml:FeatureCollection//base:Identifier[base:
namespace = $namespace][base:localId = $localId]/..)"/>
<sch:assert test="( (string-length($localId)>0) and
(string-length($namespace)>0) and ($uniqueId =1) )">
```

Example: Check Codelist Target

If an external codelist is available in an XML based format (in this case RDF), we can check if a specific entry exists

```
<sch:let name="zoneType" value=
"aqd:aqdZoneType/@xlink:href"/>
<sch:let name="codelist"
value="document('http://dd.eionet.europa.eu/vocabulary/aq/
zonetype/rdf')"/>
<sch:let name="inDictionary"
value="$codelist/rdf:RDF/skos:Concept[@rdf:about =
$zoneType]"/>
<sch:assert test="$inDictionary" >
```

Example: Check XLink Target

After parsing the xlink, we can check if the target exists within the file. This can also be done for a different file.

```
<sch:let name="content" value="substring-
after(@xlink:href, 'AQD/')"/>
<sch:let name="featureCollection"
value="/gml:FeatureCollection"/>
<sch:let name="valueProvided"
value="$featureCollection/gml:featureMember/aqd:AQD_Attain
ment/aqd:inspireId/base:Identifier[base:localId =
$content]"/>
<sch:assert test="$valueProvided">
```

Conclusion

- Schematron is very valuable for XML document validation beyond schema validation
- Schematron rules are often difficult to develop as no IDE for support
- As many validation checks are similar, patterns can be reused

For more info, see:
http://inspireaq.jrc.ec.europa.eu/wiki/index.php/Schematron

This work was performed under JRC contract Ares (2013)3113059

Contact & Information

Katharina Schleidt

Katharina.Schleidt@umweltbundesamt.at

Barbara Magagna

Barbara.Magagna@umweltbundesamt.at

Umweltbundesamt
www.umweltbundesamt.at

INSPIRE-GWF
Lissabon ■ 27.05.2015