Schematron - define and enforce your business rules

George Bina

@georgebina
georgebina@oxygenxml.com

syncro soft

<oXygen/> XML Editor

Overview

What is Schematron?

Technical details

oXygen's support for Schematron

Use cases

Ideas for the future

Conclusions

What is Schematron?

An ISO standard

ISO/IEC 19757 - DSDL Document Schema Definition Language -
Part 3: Rule-based validation - Schematron

An XML schema language

Rule-based, using XPath to express assertions

A simple and very useful technology

The most simple XML schema language

Express and enforce business rules

Technical details

Schematron and XPath

Schematron structure

Schematron implementations

Schematron and XPath

XPath is used in Schematron for

- specifying context for rules
- tests for assertions and reports
- display additional information in error messages

schema/@queryBinding will determine the XPath version used in a Schematron schema

Schematron structure

Pattern

Rule

(Assertion or Report)

…

…

…

Validation phases

Diagnostics

Patterns

Patterns assign an ID to a collection of rules

```
<pattern id="test">
  <rule … > …  </rule>
  <rule … > …  </rule>
</pattern>
```

Abstract patterns

define parameterized rule templates that can be instantiated with specific parameter values

Rules

Define a context

Specify assertions and reports to be checked

```
<rule context=”e[@id]>
    <assert …> … </assert>
    <report …> … </report>
</rule>
```

Important:

Only the first rule from a pattern that matches the current node will be active

Rule activation

```
<pattern>
    <!--1--><rule context="e[@x]"> … </rule>
    <!--2--><rule context="e"> … </rule>
    …
</pattern>
```

Rule 2 matches on element e but it will be active only on elements that do not have an x attribute, its context is equivalent to e[not(@x)]

Assertions and Reports

Assertions check if some condition is met, if not an error is reported.

We assert that the number of people is less than or equal to 10:

```
<assert test="count(person) &lt;= 10"> error… </assert>
```

Reports are reverse assertions, an error is reported if the condition is met.

We report an error is the number of people is more than 10:

```
<report test="count(person) > 10">error… </report>
```

Validation phases

Specifies active patterns for a specific validation phase:

```
<phase id="basicValidation">
    <active pattern="p1"/>
    <active pattern="p2"/>
</phase>
```

The built-in phase #ALL includes all patterns

Diagnostics

Can be referred from an assertion or report

Provide additional details/messages to help understanding the error

Allow for reuse, multiple assertions/reports can refer to the same diagnostic

```
<assert test="@att" diagnostics="x">att is missing!</assert>
…
<diagnostic id="x">
    The element <value-of select="@*/name()"/> should have an attribute named "att", but we have instead attributes <value-of select="@*/name()"/>!
</diagnostic>
```

Composability

This is a weak point for Schematron…

You cannot include a schema, you can include only schema parts:

```
<include href="schemaFragment.sch"/>
```

The entire content of schemaFragment.sch will replace the include instructions

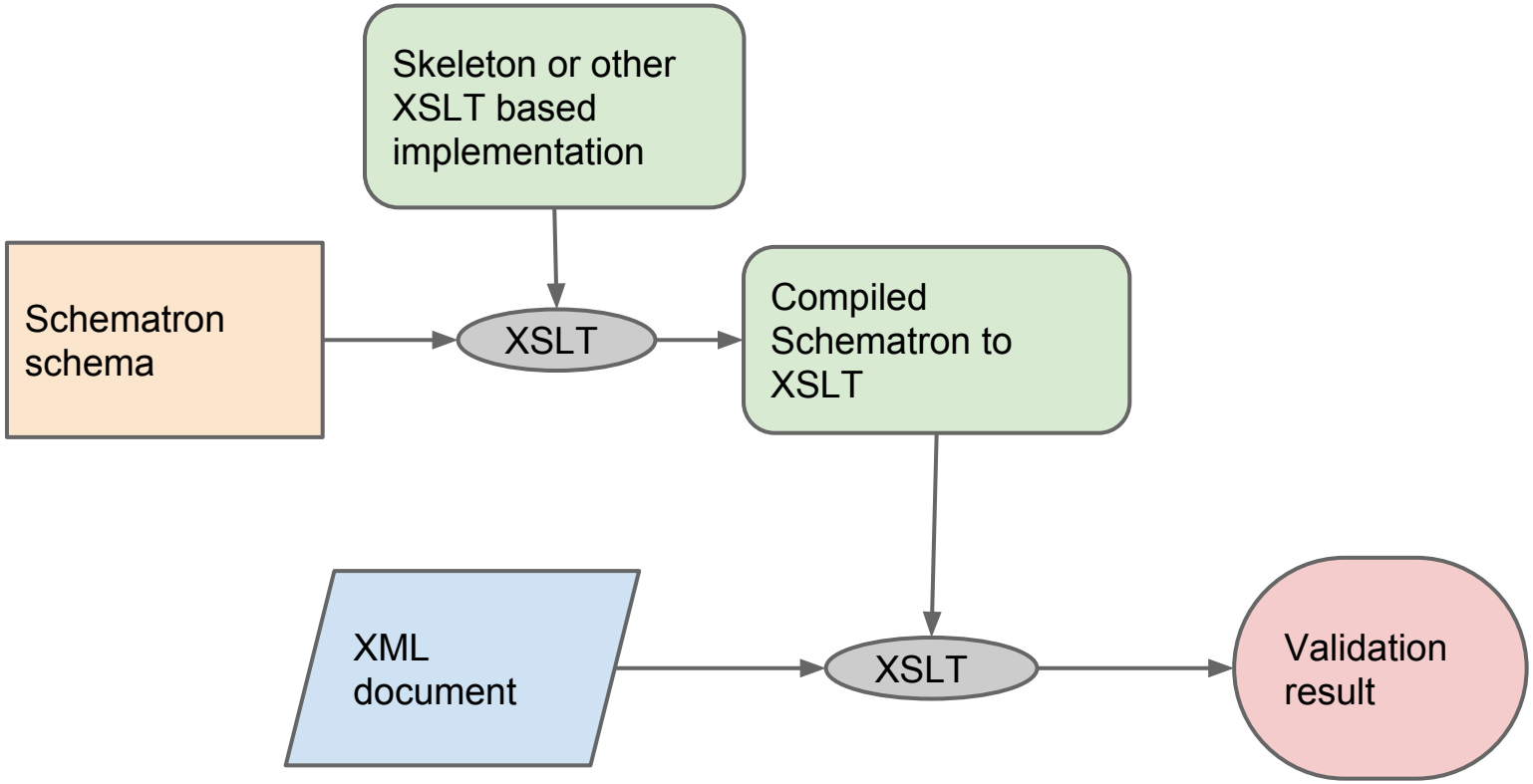
Maybe a better support will be added in the next version

Schematron implementations

All based on XSLT:

- Skeleton
  - Reference implementation
- Jing
  - Only pre-ISO support
- oXygen
  - a fork of Skeleton to add ISO Schematron support and additional functionality

Schematron validation



oXygen's support for Schematron

XML development

- Create Schematron schemas

XML authoring

- Validate XML documents with Schematron

XML development

Create Schematron schemas

- Edit
  - content completion with embedded XPath support
  - visual editing mode
- Validate the Schematron schema
- Support embedded Schematron
  - as annotations in XML Schema and Relax NG
- Support for multiple Schematron versions
  - ISO Schematron, pre-ISO version 1.5 and 1.6

XML authoring

- Validation as you type and on request
- Batch validation
- DITA topics validation from a DITA map
- Highlight errors directly in XML documents
- Different error levels: fatal, error, warn, info
- Link an error to a URL for more details

Schematron at authoring time

Detecting errors as they appear

(authoring time)

- easy to fix by the same user that made the error, already in the error context, already at the location of the error → almost no cost associated with fixing them

is better than detecting them later on

(reviewing time)

- found by someone else, need to locate the error, understand the error context, fix it, double check that the error is fixed → large cost for fixing errors

Use cases

1. Provide error messages that normal people can understand
2. Check rules that cannot be specified in DTD, XML Schema or Relax NG
3. Controlled values - check against an external data source
4. Integrity checks across multiple files
5. Styleguide integration
6. Test transformation chains end-to-end
7. Allows anyone to define business rules

1. Error messages

Missing topic title

2. Check additional rules

Additional DITA specification checks, based on Jarno Elovirta Schematron schemas for DITA

Make sure we have whitespace after UI controls.

Controlled values

Check revision values in an external data source.

4. Integrity checks

Check references

Check a DITA cross reference from one topic to another, checking that

- the topic exists
- the topic ID exists
- the element ID is within the referred topic

Check specific design constraints in schemas

Check that the domains attribute contains all the domains values of the included domains in the DITA Relax NG based shells

5. Styleguide integration

The DITA styleguide by Tony Self

- avoid just one “p” in a note
- no text mixed with block elements
- A note should not start with “Note:”

Each check links to the styleguide topic that describes that issue in detail.

6. End to end transformations

Useful for complex transformation to make sure we do not lose content

Example:

If the input has 6 lists we should find 6 lists in the output

Ideas for the future

Trigger automatic fixes from Schematron

Automatic fixes

Link to one or more actions that can perform automatic fixes

Examples

The title is missing, please add a title at the beginning of this section!

Add title

A note should not start with the word “Note”, please delete this or change the note to a normal paragraph!

Delete “Note”, Rename note to para

Conclusions

- Schematron is an easy to use but very powerful technology, there is no reason why you should not include it in your projects
- Support for friendly error messages is a major advantage
- oXygen integrates support for Schematron to allow catching problems as they appear, cutting costs associated to late detection
- Learn more about Schematron at http://www.xfront.com/schematron/index.html

Thank you!

Questions?

Now, or later at:

@georgebina

george@oxygenxml.com