

Schematron Step-By-Step Guide

Octavian Nadolu, Syncro Soft
octavian_nadolu@oxygentools.com
@OctavianNadolu

tcworld, 2018
© 2018 Syncro Soft SRL. All rights reserved.



schematron
Structured
editing
XML
review
XQuery
Publish
PDF
IDREFS
WebDAV
DTD DocBook
oXygen
authoring
XML Editor
XSD SCH XSD Single
XPR RNC FO Source
frameworks
Profiling
WSDL
styles
visual
WebHelp
DITA
TEI
XSL
PHP
Ant
Js



About the Author

- Software architect
- 15+ years XML technologies experience
- Contributor to a number of XML-related open source projects
- Editor of Schematron QuickFix specification developed by a W3C community group

Overview

- Schematron history
- Schematron step-by-step
- Schematron in the development process
- Schematron for technical and non-technical users
- Schematron Quick Fixes

What is Schematron?

- A natural language for making assertions in documents



Schematron is an Open Standard

- Schematron is an Open Standard adopted by hundreds of major projects in the past decade



Schematron History

- Schematron was invented by Rick Jelliffe



- Schematron 1.0 (1999), 1.3 (2000), 1.5 (2001)
 - ISO Schematron (2006, 2010, 2016) The ISO logo, which consists of a globe icon and the word "ISO" in blue capital letters.



Why Schematron?

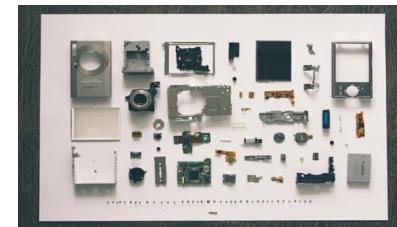
- You can express constraints in a way that you cannot perform with other schemas (like XSD, RNG, or DTD).
 - XSD, RNG, and DTD schemas define structural aspects and data types of the XML documents
 - Schematron allows you to create custom rules specific to your project

Schematron Usage

- Verify data inter-dependencies
- Check data cardinality
- Perform algorithmic checks

Used in Multiple Domains

- Financial
- Insurance
- Government
- Technical publishing



Schematron is Very Simple

- There are only 6 basic elements:

assert

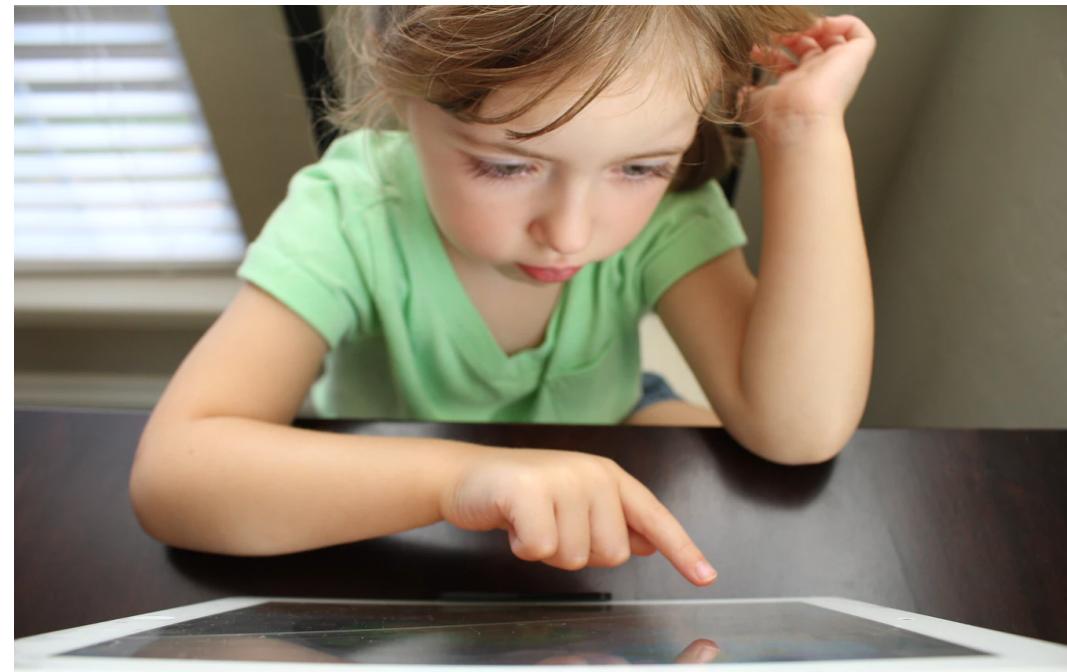
report

rule

pattern

schema

ns





Schematron Step-by-Step

XML

A list of books from a bookstore:

```
...
<book price="75">
  <title>XSLT 2.0 Programmer's Reference</title>
  <author>Michael Kay</author>
  <isbn>0764569090</isbn>
</book>
...
```

<assert>

An **assert** generates a message when a test statement evaluates to **false**

```
<assert test="@price > 10">The book price is too small</assert>
```

<report>

A **report** generates a message when a test statement evaluate to **true**.

```
<report test="@price > 1000">The book price is too big</report>
```

<rule>

A **rule** defines a context on which a list of assertions will be tested, and is comprised of one or more **assert** or **report** elements.

```
<rule context="book">
  <assert test="@price > 10">The book price is too small</assert>
  <report test="@price > 1000">The book price is too big</report>
</rule>
```

<pattern>

A set of rules giving constraints that are in some way related

```
<pattern>
  <rule context="book">
    <assert test="@price > 10">The book price is too small</assert>
    <report test="@price > 1000">The book price is too big</report>
  </rule>
</pattern>
```

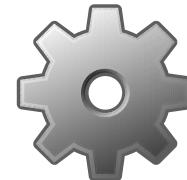
<schema>

The top-level element of a Schematron schema.

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
  <pattern>
    <rule context="book">
      <assert test="@price > 10">The book price is too small</assert>
      <report test="@price > 1000">The book price is too big</report>
    </rule>
  </pattern>
</schema>
```

Apply Schematron

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
  <pattern>
    <rule context="book">
      <assert test="@price > 10">The book price is too small</assert>
      <report test="@price > 1000">The book price is too big</report>
    </rule>
  </pattern>
</schema>
```



The price is too small

```
...
<book price="7">
  <title>XSLT 2.0 Programmer's Reference</title>
  <author>Michael Kay</author>
  <isbn>0764569090</isbn>
</book>
...
```

<ns>

Defines a namespace prefix and URI

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
  <ns uri="http://book.example.com" prefix="b"/>
  <pattern>
    <rule context="b:book">
      <assert test="@price > 10">The book price is too small</assert>
      <report test="@price > 1000">The book price is too big</report>
    </rule>
  </pattern>
</schema>
```

Schematron Uses XPath

- XPath is very important in Schematron:
 - Rule context is expressed using an XPath expression

```
<rule context="book">
```
 - Assertions are expressed using XPath expressions that are evaluated as *true* or *false*

```
<assert test="@price > 10">
```
- *@queryBinding* – determines the XPath version
 - Possible values: *xslt*, *xslt2*, *xslt3*

```
<schema queryBinding="xslt2">
```

What is XPath?

- A language for addressing parts of an XML document (XML Path Language)
- A W3C Recommendation in 1999
www.w3.org/TR/xpath
- XPath versions 1.0, 2.0, 3.0, 3.1 (2017)
- Tutorials:
 - www.data2type.de/en/xml-xslt-xslfo>xpath/
 - www.xfront.com>xpath/
 - www.zvon.org/comp/m>xpath.html



Conclusion

- Schematron is simple (6 basic elements)
- Used in multiple domains
- Schematron uses XPath



Examples of Schematron Rules



Check Number of List Items

- Example of a rule that checks the number of list items in a list

A list must have more than one item



```
ul
• li p Gerbera - is a genus of ornamental plants from the sunflower family
(Asteraceae). It was named in honor of the German naturalist Traugott Gerber.
p li
ul
```



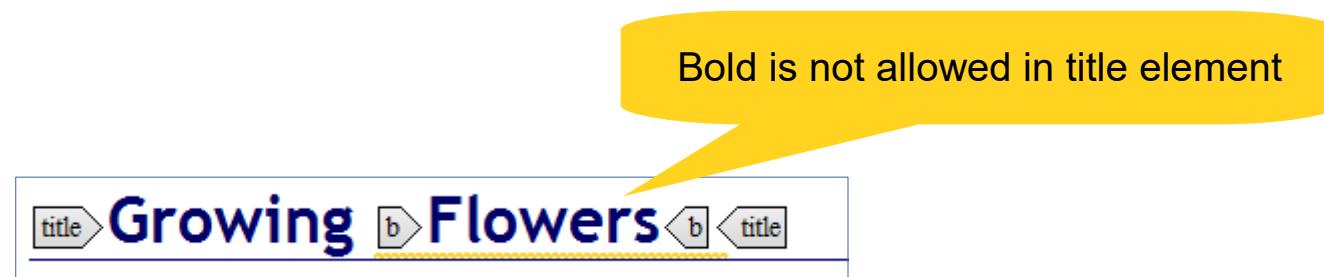
Check Number of List Items

- The number of list items from an unordered list should be greater than one

```
<sch:rule context="ul">
  <sch:assert test="count(li) > 1">
    A list must have more than one item</sch:assert>
</sch:rule>
```

Styling in Titles

- Example rule that checks for styling in titles



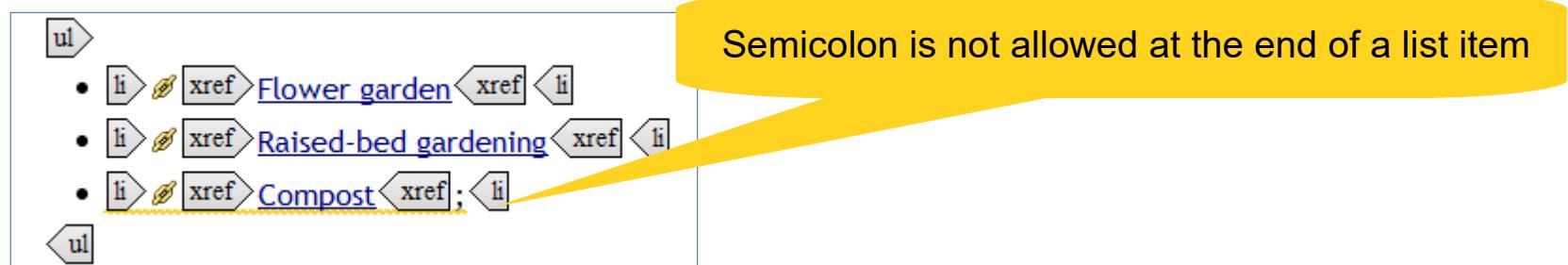
Styling in Titles

- The **b** element should not be used in a title

```
<sch:rule context="title">
  <sch:report test="b">
    Bold is not allowed in <sch:name/> element</sch:report>
</sch:rule>
```

Semicolon at End of List Items

- Example of rule that checks if a semicolon is at the end of a list item



Semicolon at End of List Items

- The last text from a list item should not end with semicolon

```
<sch:rule context="li">
  <sch:report test="ends-with(text()[last()], ';')">
    Semicolon is not allowed after list item</sch:report>
</sch:rule>
```

Check External Link Protocol

- Example of rule that check the external link protocol

The external link should start with http(s)

p>Most of the information was taken from  www.wikipedia.org/ 



Check External Link Protocol

- The `@href` attribute value it is an external link and should start with http or https.

```
<rule context="xref">
  <sch:assert test="matches(@href, '^http(s?)://')">
    An link should start with http(s).</sch:assert>
</rule>
```

Missing Tables Cells

- Missing cells in a table

Flower	Type	
Chrysanthemum	perennial	well drained
Gardenia	perennial	
Gerbera	annual	sandy, well-drained
Iris		

Cells are missing from table



Missing Tables Cells

- Check that are the same number of cells on each row

```
<sch:rule context="table">
  <sch:let name="minColumnsNo" value="min(//row/count(entry))"/>
  <sch:let name="reqColumnsNo" value="max(//row/count(entry))"/>

  <sch:assert test="$minColumnsNo >= $reqColumnsNo">
    Cells are missing. (The number of cells for each row must be
    <sch:value-of select="$reqColumnsNo"/>)
  </sch:assert>
</sch:rule>
```

Link in Text Content

- Links are added directly in text content

Link detected in the current element



Note: Most of the information was taken from <http://www.wikipedia.org> the free encyclopedia.



Link in Text Content

- Check if a link is added in text content but is not tagged as a link

```
<sch:rule context="text()">
  <sch:report test="matches(., '(\http|www)\S+')
    and local-name(parent::node()) != 'xref'">
    The link should be a xref element</sch:report>
  </sch:rule>
```

Conclusion

- Simple to complex Schematron rules
- Additional Schematron elements used:
 - `<let>` - A declaration of a named variable
 - `<value-of>` - Finds or calculates values from the instance document
 - `<name>` - Provides the names of nodes from the instance document



Integrating Schematron in the Development Process



Validate XML with Schematron

- Associate Schematron in the XML file

```
<?xml-model href="books.sch" type="application/xml"  
    schematypens="http://purl.oclc.org/dsdl/schematron"?>
```

- Use tool-specific association options

- Associate Schematron file with a set of XML files (all files with a specific namespace, or from a directory)
- Associate Schematron with all XML files from a project

Embed Schematron in XSD or RNG

- Schematron can be added in the XSD **appinfo** element

```
<xsd:appinfo>
  <sch:pattern>
    <sch:rule context="...">
      <sch:assert test="...">>Message.</sch:assert>
    </sch:rule>
  </sch:pattern>
</xsd:appinfo>
```

- Add Schematron checks in any element on any level of an RNG schema

```
<grammar
  xmlns="http://relaxng.org/ns/structure/1.0"
  xmlns:sch="http://purl.oclc.org/dsdl/schematron">
  <sch:pattern>
    <sch:rule context="...">
      <sch:assert test="...">>Message.</sch:assert>
    </sch:rule>
  </sch:pattern>
  <start>
  .....
  </start>
</grammar>
```

Run Schematron Validation

- From an XML editing framework
 - Check the XML files as you type
 - Run the Schematron validation on multiple files
- Using W3C's XProc pipeline language through its "validate-with-schematron" step

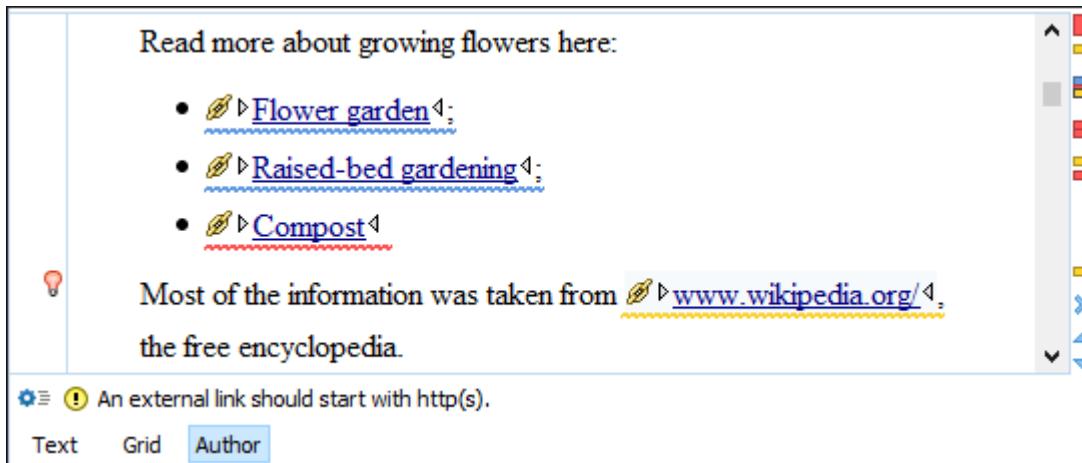


- Using Apache Ant, from bat/shell
- <https://github.com/Schematron/ant-schematron>



Schematron Validation Result

- Messages presented in the editing framework



- As an XML file describing the validation errors, normally in Schematron Validation Reporting Language (SVRL)

```
<svrl:failed-assert test="matches(@href, '^http(s?)://')" role="warn"
                     location="/topic[1]/body[1]/section[1]/p[3]/xref[1]">
  <svrl:text>An external link should start with http(s).</svrl:text>
</svrl:failed-assert>
```

Schematron Messages Severity

- Severity level can be set in the value of the `@role` attribute from an `assert` or `report` element (`fatal`, `error`, `warn`, or `info`)
- Depending on the severity, the message can be rendered differently in an editing framework

The screenshot shows a web-based editing environment. At the top, there's a header bar with a logo and some navigation links. Below the header, the main content area has a title "Growing Flowers" with a yellow decorative underline. Under the title is a section titled "Introduction" with a small downward arrow icon. The introduction text reads: "With just a little bit of care and preparation, any flower garden can be a vibrantly colored environment. Flowers can be selected for specific blooming seasons, colors and shapes. Both annual and perennial flower gardens can be planted depending on climate and specific needs." Below the introduction, there's a link "Read more about growing flowers here:". Underneath that link is a bulleted list of three items, each preceded by a small icon of a flower in a pot. The items are: "Flower garden", "Raised-bed gardening", and "Compost". To the right of the main content area, there's a vertical sidebar with several colored buttons (blue, red, yellow) and some other UI elements.

Multilingual Support in Schematron

- Based on the Schematron diagnostic element
- A diagnostic element is used for each language
- Multilingual support defined in the Schematron specification

```
<sch:assert test="bone" diagnostics="d_en d_de">
    A dog should have a bone.
</sch:assert>
....
<sch:diagnostics xml:lang="en">
    <sch:diagnostic id="d_en">A dog should have a bone.</sch:diagnostic>
</sch:diagnostics>
<sch:diagnostics xml:lang="de">
    <sch:diagnostic id="d_de">Das Hund muss ein Bein haben.</sch:diagnostic>
</sch:diagnostics>
```

Conclusion

- Multiple ways to associate and apply the validation
- Validation results as you type or in a report
- Messages with different severity
- Multilingual support



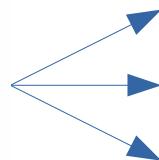
Schematron for Non-Technical and Technical Users

Schematron for Non-Technical Users

- Schematron is used in multiple domains
- There are more and more non-technical users that want to develop their own rules
- Providing a user interface to create the Schematron rules will be more appropriate for them

Library of Rules

- A solution can be to provide a library of rules
- The user can choose the rule that he wants to add

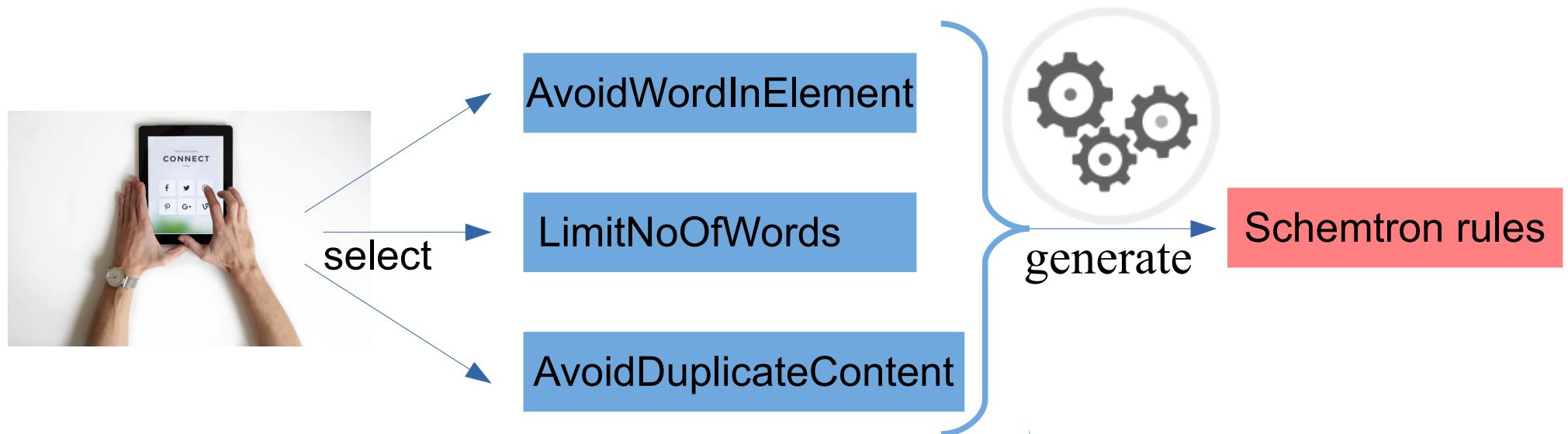


Avoid a word in a certain element
Limit the number of words
Avoid duplicate content



Abstract Patterns

- Library of rules implemented using abstract patterns



Abstract Pattern

- Allows to reuse patterns by making them generic

```
<pattern id="LimitNumberOfWords" abstract="true">
  <rule context="$parentElement">
    <let name="words" value="count(tokenize(normalize-space(.), ' '))"/>
    <assert test="$words le $maxWords">
      $message You have <value-of select="$words"/> word(s).
    </assert>
    <assert test="$words ge $minWords">
      $message You have <value-of select="$words"/> word(s).
    </assert>
  </rule>
</pattern>
```

Pattern Instantiation

- Refer an abstract pattern and specifies values for parameters

```
<pattern is-a="LimitNumberOfWords">
  <param name="parentElement" value="shortdesc"/>
  <param name="minWords" value="1"/>
  <param name="maxWords" value="50"/>
  <param name="message" value="Keep short descriptions between 1 and 50 words!"/>
</pattern>
```

```
<pattern is-a="LimitNumberOfWords">
  <param name="parentElement" value="title"/>
  <param name="minWords" value="1"/>
  <param name="maxWords" value="8"/>
  <param name="message" value="Keep titles between 1 and 8 words."/>
</pattern>
```

Create Business Rules Using DIM

Business Rule X

Search for business rule

- avoidWordInElement
- avoidEndFragment
- avoidAttributeInElement
- recommendElementInParent
- restrictWords
- restrictCharacters
- restrictNesting
- restrictNumberOfChildren
- restrictChildrenElements
- avoidDuplicateContent
- requireContentAfterElement
- dita-allowedElementsForClass
- dita-allowOnlyBlockElements

Issue a warning if a an element end with a specified fragment or character

This pattern allows you to advise users not to use a specific end sequence to end an element.

As parameters we have **fragment** that points to the text that we need to check, **element** that points to the element we will check to not end with that fragment and **message** that contains the message we should display to the user in case the fragment appears at the end of the specified element.

OK Cancel

Rule	avoidEndFragment
element	li
fragment	:
message	Avoid ; at the end of a list item

Dynamic Information Model (DIM)

- An implementation of an intelligent style guide
- Describes and enforces rules
- Open-source project available on GitHub
<https://github.com/oxygenxml/dim>

Schematron for Technical Users

Environment to develop Schematron schemas

- Editor
- Validation
- Search and refactoring
- Working with modules
- Unit testing

Editor

- Syntax highlighting - to improve the readability of the content
- Content completion assistant - offering proposals that are valid at the cursor position
- Documentation - with information about the particular proposal

The screenshot shows an XML editor interface with the following details:

- Code Area:** Displays an XML snippet with syntax highlighting. A tooltip is shown over the `<sch:assert` element.
- Completion List:** A dropdown menu lists several schema elements:
 - `= sch:assert` (selected)
 - `● sch:extends`
 - `↳ sch:include`
 - `$v sch:let`
 - `¶ sch:p`
 - `# sch:report`
- Documentation Pop-up:** A yellow tooltip provides information about the `<sch:assert>` element:
 - An assertion made about the context nodes. The data content is a natural-language assertion. The natural-language assertion shall be a positive statement of a constraint.
 - NOTE: The natural-language assertion may contain information about actual values in addition to expected values and may contain diagnostic information. Users should note, however, that the diagnostic element is provided for such information to encourage clear statement of the natural-language assertion.

Validate

- Validate Schematron schemas, and highlight problems directly in the editor
- Validate Schematron modules in context
- Support for validation as you type

The screenshot shows a code editor with Schematron validation feedback. The XML code is as follows:

```
<sch:pattern>
    <!-- Report cases when the lines in a codeblock exceeds 90 characters -->
    <sch:rule context="*[contains(@class, ' pr-d/codeblock ')]" role="warn">
        <sch:let name="offendingLine" value="oxyF:lineLengthCheck(string(), 90)"/>
        <sch:report test="string-length($offendingLines) > 0">
            Lines (<sch:value-of select="$offendingLines"/>) in codeblocks should not exceed 90
        </sch:report>
    </sch:rule>
</sch:pattern>
```

A validation message is displayed in a tooltip:

Validation:
! Variable offendingLines has not been declared (or its declaration is not in scope)

Press F2 for focus

Search and Refactoring

- Find references to variables, pattern, phases, or diagnostics
- Support for renaming all references from the current document, or in the entire hierarchy
- Preview the changes

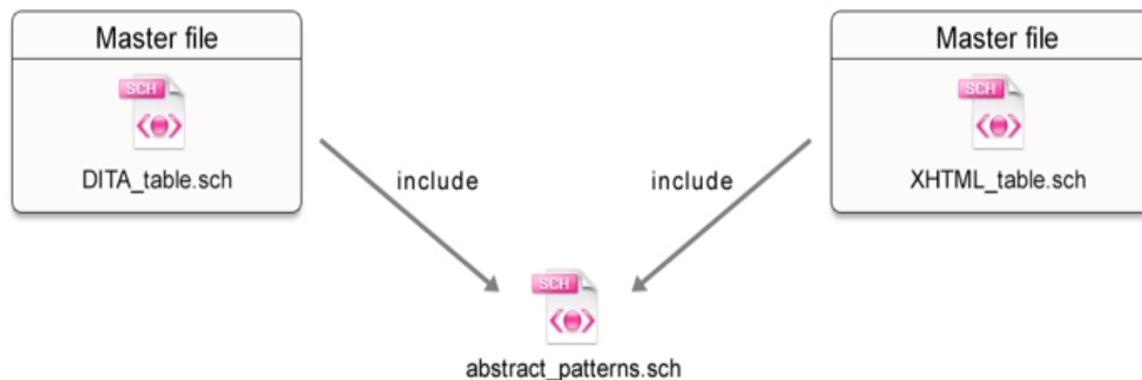
The screenshot shows an XML editor interface with the following details:

- XML Code:**

```
<rule context="$parentElement">
    <let name="characters" value="string-length(if ($normalize = ('true', 'true()', 'yes')) then normalize-space(.) else .)" />
    <assert test="$characters &lt;= $maxChars" role="warn">
        $message
        It is recommended to not exceed $maxChars
        <value-of select="if ($maxChars=1) then ' character' else ' characters'" />
        You have <value-of select="$characters"/>
        <value-of select="if ($characters=1) t
    </assert>
    <assert test="$characters &gt;= $minChar
        $message
        It is
        recommended to have at least $minChars
        <value-of select="if ($maxChars=1) the
        You have <value-of select="$characters
        <value-of select="if ($characters=1) t
    </assert>
```
- Context Menu:** A context menu is open over the word "characters".
 - Variable: 'characters' Scope: Master Files**
 - Rename Component in...** (highlighted)
 - Search Declarations (Ctrl+Shift+D)
 - Search References
 - Change scope...
 - Variable: 'characters' Scope: Current File**
 - Rename Component** (Alt+Shift+R)
 - Search Occurrences (Ctrl+Shift+U)
- Description Box:** A yellow box provides a tooltip for the "Rename Component in..." option: "Renames the component and updates all its references. Scope: Master Files".

Working with Modules

- Visualize the hierarchy of modules
- Support for editing a Schematron module in the context
- Moving and renaming modules



Test Schematron Rules

- Developing Schematron schema also involves testing
- Make sure Schematron rules work as expected
- Apply multiple XML examples over the Schematron rules and check the results



XSpec

- Solution to create Schematron unit tests
- Unit test and BDD (Behaviour Driven Development) framework
 - <https://github.com/xspec/xspec/>
- Oxygen XSpec Helper View plugin
 - github.com/xspec/oXygen-XML-editor-xspec-support
- Tutorials
 - github.com/xspec/xspec/wiki/Getting-Started-with-XSpec-and-Schematron
 - oxygenxml.com/events/2018/webinar_xspec_unit_testing_for_xsbt_and_schematron.html

Conclusion

- Library of rules using abstract patterns
- User interface for non-technical users
- Editing, validation, and refactoring support for technical users
- Test cases for Schematron





Schematron Quick Fixes

Schematron Fix Proposals

- Schematron assertion messages are not always enough for the user to find a solution
- It is better to have some proposals to correct the Schematron reported problem
- Similar to spell check proposals



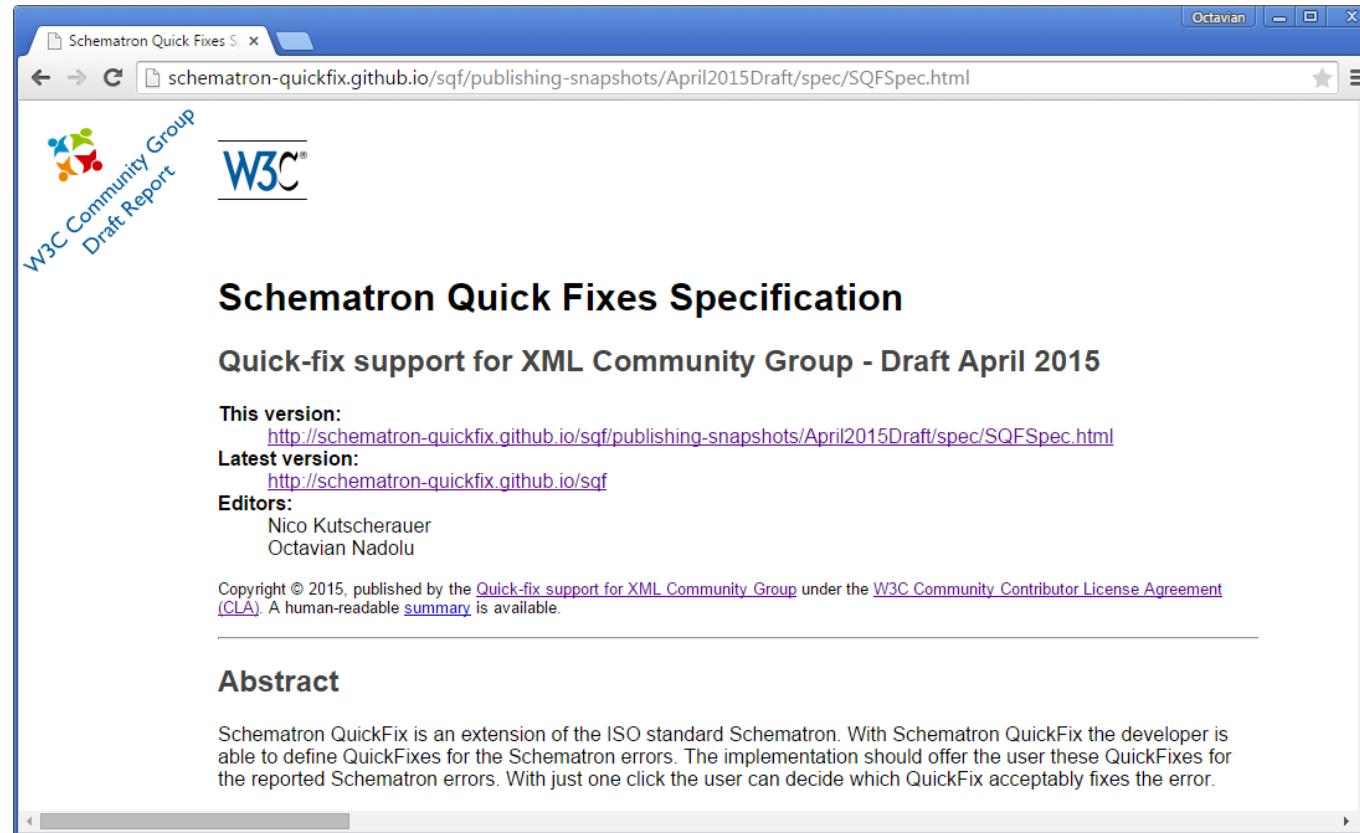
Schematron Quick Fix Proposals

- User-defined fixes for Schematron errors
- Schematron Quick Fix (SQF) language
 - Extension of the Schematron language
 - SQF initiated by Nico Kutscherauer



www.schematron-quickfix.com
github.com/schematron-quickfix/sqf

Schematron Quick Fixes Spec



The screenshot shows a web browser window titled "Schematron Quick Fixes S...". The address bar displays the URL: "schematron-quickfix.github.io/sqf/publishing-snapshots/April2015Draft/spec/SQFSpec.html". The page content includes the W3C Community Group Draft Report logo and the W3C logo. The main title is "Schematron Quick Fixes Specification" and the subtitle is "Quick-fix support for XML Community Group - Draft April 2015". It provides links for "This version" (<http://schematron-quickfix.github.io/sqf/publishing-snapshots/April2015Draft/spec/SQFSpec.html>) and "Latest version" (<http://schematron-quickfix.github.io/sqf>). It also lists "Editors": Nico Kutscherauer and Octavian Nadolu. A copyright notice at the bottom states: "Copyright © 2015, published by the [Quick-fix support for XML Community Group](#) under the [W3C Community Contributor License Agreement \(CLÄ\)](#). A human-readable [summary](#) is available."



www.w3.org/community/quickfix

schematron-quickfix.github.io/sqf

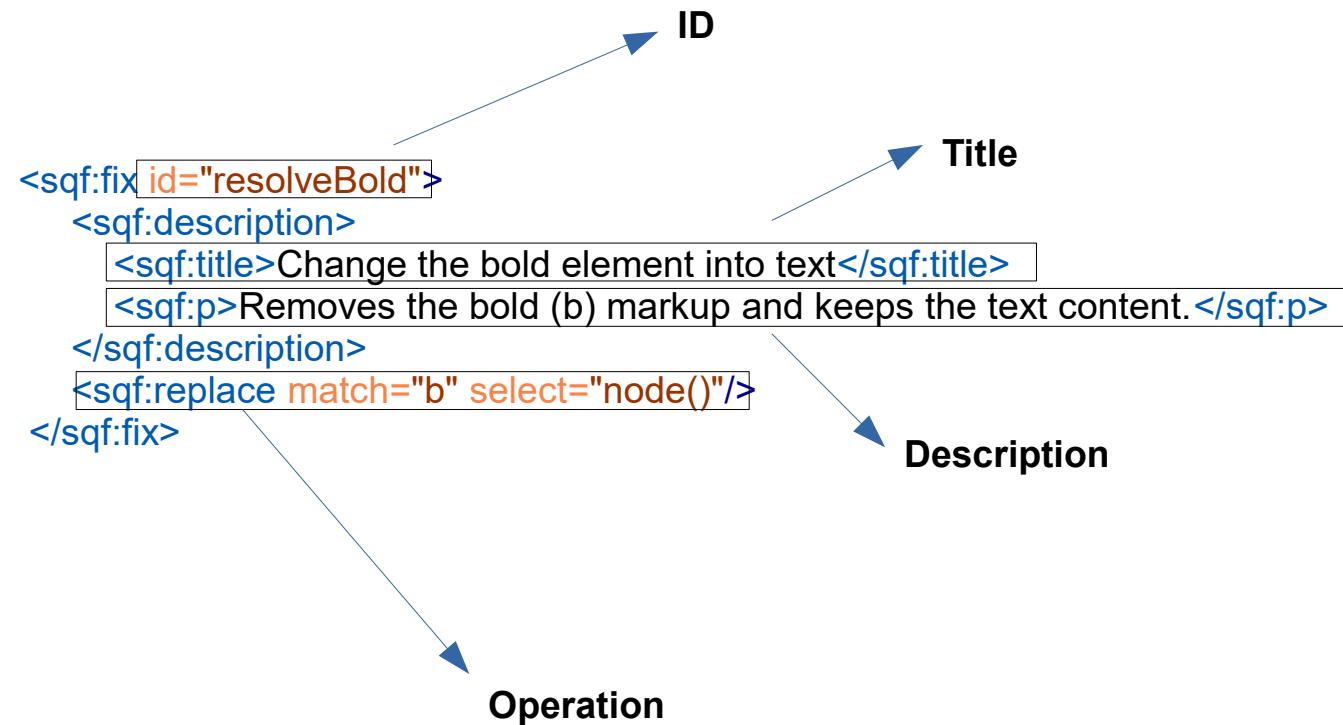
SQF Extension of Schematron

- Added as Schematron annotations
- Associate fixes with assert and report elements

```
<rule context="title">
  <report test="b" sqf:fix="resolveBold">
    Bold is not allowed in title element</report>
```

```
<sqf:fix id="resolveBold">
  <sqf:description>
    <sqf:title>Change the bold element into text</sqf:title>
    <sqf:p>Removes the bold (b) markup and keeps the text content.</sqf:p>
  </sqf:description>
  <sqf:replace match="b" select="node()" />
</sqf:fix>
</rule>
```

Schematron Quick Fix (SQF)



SQF Operations

The following 4 types of operations are supported:

- <sqf:add> - To add a new node or fragment in the document
- <sqf:delete> - To remove a node from the document
- <sqf:replace> - To replace a node with another node or fragment
- <sqf:stringReplace> - To replace text content with other text or a fragment

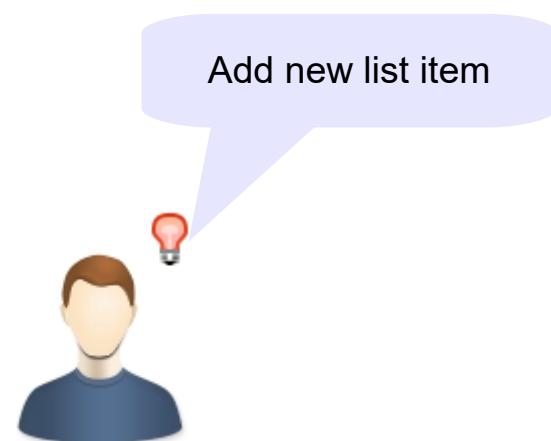
Introduction to SQF Through Examples

1. SQF “add” Operation

- Example of using the “add” operation: add new list item in a list

List contains only one item

- Summer Flowers
 - Gardenia - is a genus of about 250 species of flowering plants in the coffee family, Rubiaceae, native to the tropical and subtropical regions of Africa, southern Asia, Australasia and Oceania.



1. SQF “add” Operation

- <sqf:add> element allows you to add one or more nodes to the XML instance

```
<rule context="ul">
    <assert test="count(li) > 1" sqf:fix="addListItem">A list must have more
    than one item.</assert>

    <sqf:fix id="addListItem">
        <sqf:description>
            <sqf:title>Add new list item</sqf:title>
        </sqf:description>
        <sqf:add node-type="element" target="li" position="last-child"/>
    </sqf:fix>
</rule>
```

2. SQF “delete” Operation

- Example of using the “delete” operation: remove redundant link text

Text in the link and the value of the @href are the same

 Most of the information was taken from   <http://www.wikipedia.org> , the free encyclopedia. 

Remove redundant link text



2. SQF “delete” Operation

- <sqf:delete> element specifies the nodes for the deletion

```
<rule context="xref">
  <report test="@href = text()" sqf:fix="removeText">
    Link text is same as @href attribute value. Please remove.</report>

  <sqf:fix id="removeText">
    <sqf:description>
      <sqf:title>Remove redundant link text</sqf:title>
    </sqf:description>
    <sqf:delete match="text()" />
  </sqf:fix>
</rule>
```

3. SQF “replace” Operation

- Example of using the “replace” operation: replace bold element with text



3. SQF “replace” Operation

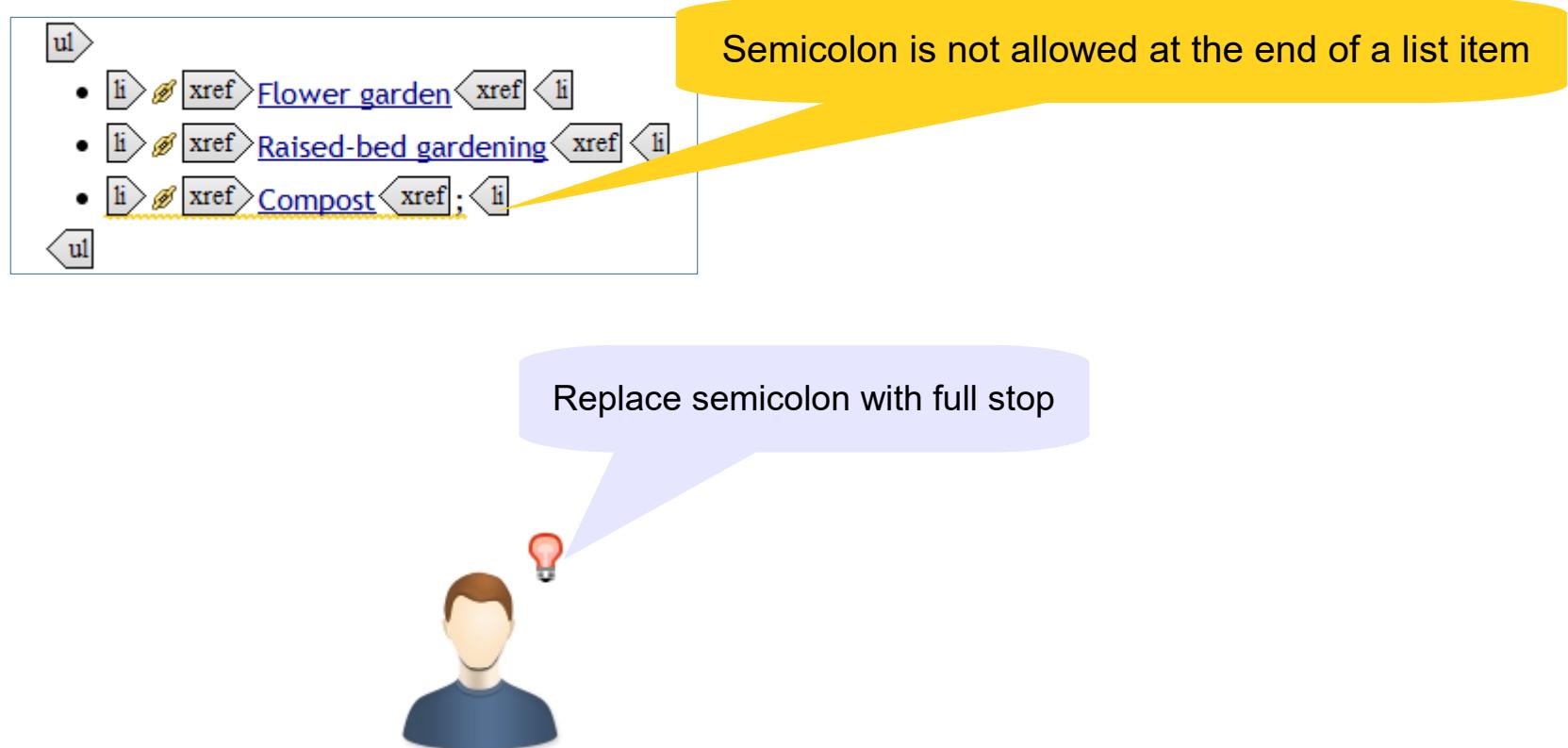
- <sqf:replace> element specifies the nodes to be replaced and the replacing content

```
<rule context="title">
    <report test="b" sqf:fix="resolveBold">
        Bold is not allowed in title element.</report>

    <sqf:fix id="resolveBold">
        <sqf:description>
            <sqf:title>Change the bold into text</sqf:title>
        </sqf:description>
        <sqf:replace match="b" select="node()" />
    </sqf:fix>
</rule>
```

4. SQF “stringReplace” Operation

- Example of using the “stringReplace” operation: replace semicolon with full stop



4. SQF “stringReplace” Operation

- <sqf:stringReplace> element defines the nodes that will replace the substrings

```
<rule context="li">
  <report test="ends-with(text()[last()], ';')" sqf:fix="replaceSemicolon">
    Semicolon is not allowed after list item</report>

  <sqf:fix id="replaceSemicolon">
    <sqf:description>
      <sqf:title>Replace semicolon with full stop</sqf:title>
    </sqf:description>
    <sqf:stringReplace match="text()[last()]" regex=";" select=".;" />
  </sqf:fix>
</rule>
```

Conclusions

- Schematron Quick Fix language is simple
- You can define custom fixes for your project
- Just 4 types of operations



SQF Implementations

- <oXygen/> XML Editor validation engine

<http://www.oxygenxml.com>

- Escali Schematron engine

http://schematron-quickfix.com/escali_xsm.html

- Escali Schematron command-line tool
 - Oxygen plugin for invoking Escali Schematron

Projects Using SQF



Thieme - publishing company uses a custom framework to create and edit XML documents



parsX - a product developed by pagina GmbH used to facilitate EPUB production



ART-DECOR - an open-source tool suite that supports SDOs active in the healthcare industry
Sample SQF embedded in XSD



ATX custom framework – used by a major automotive manufacturer

Resources

- Schematron official site
- Schematron specification
- Schematron Quick Fix specification
- Sample files:
github.com/octavianN/Schematron-step-by-step

THANK YOU!

Any questions?

<oXygen/> XML Editor
<http://www.oxygenxml.com>
octavian_nadolu@oxygenxml.com
@OctavianNadolu

STUTTGART, 13. – 15. NOVEMBER

STUTTGART, NOVEMBER 13–15

Your opinion is important to us! Please tell us what you thought of the lecture. We look forward to your feedback under

<http://ta14.honestly.de>

or scan the QR code



The feedback tool will be available even after the conference!