Taking Schematron QuickFix To The Next Level

Octavian Nadolu
octavian_nadolu@oxygenxml.com
@OctavianNadolu

syncro soft

<oXygen/> XML Editor

Overview

- Schematron QuickFix Language
- Use Cases – SQF with Regexp, XPath, XSLT, Java
- Abstract Quick Fixes
- SQF Second Draft

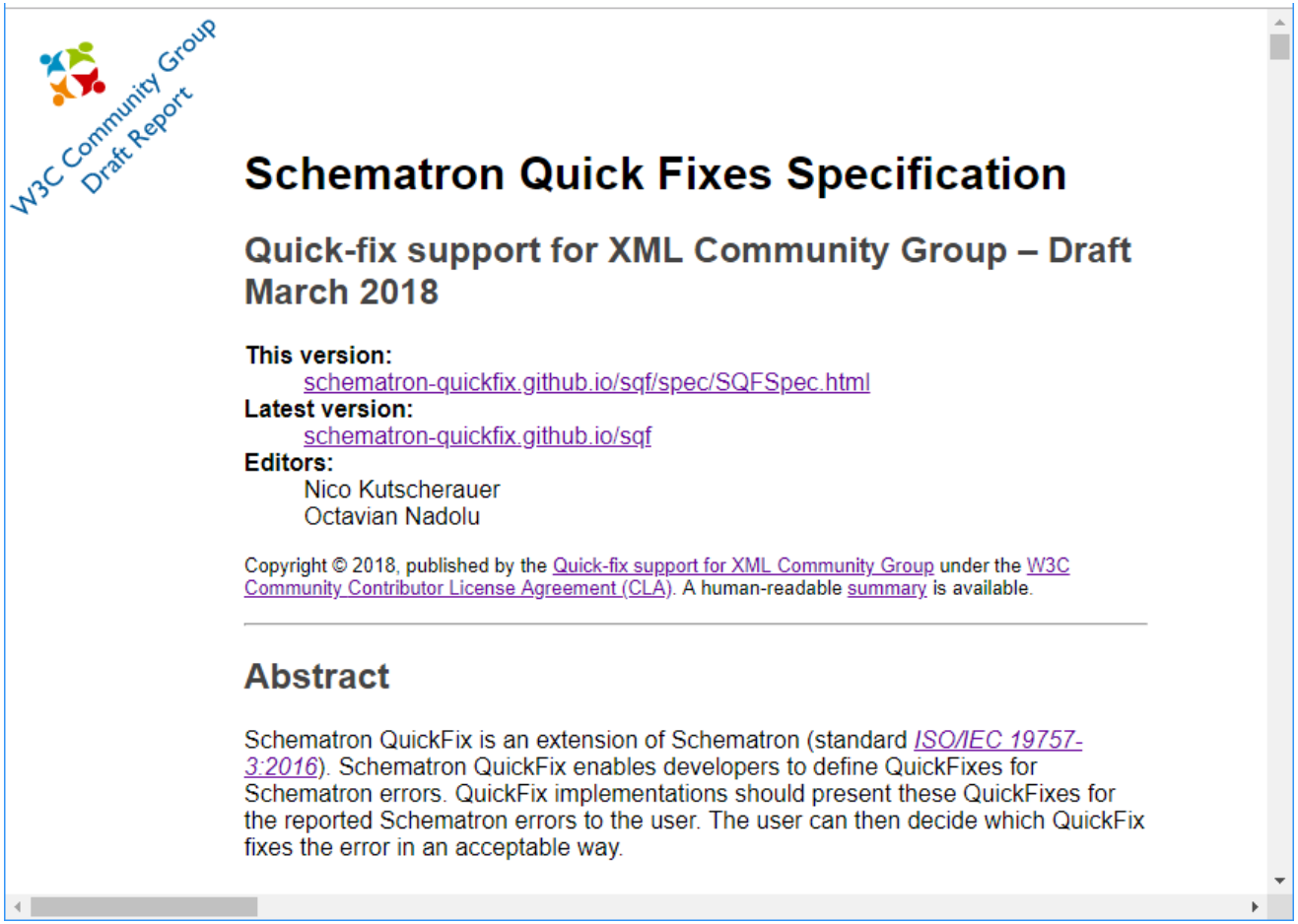
What Is SQF?

- Schematron QuickFix (SQF) is an extension of the ISO Schematron standard
- User-defined fixes for Schematron errors

SQF History

- XML Prague 2014 – the SQF idea started to take shape
- February 20, 2014 - W3C Community Group "Quick-Fix Support for XML"
- April 2015 - First Draft
- March 2018 – Second Draft

Schematron Quick Fixes Spec

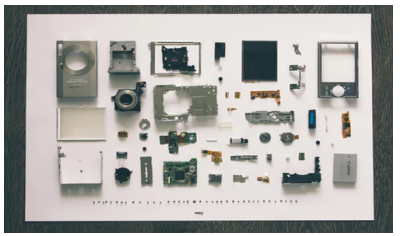


www.w3.org/community/quickfix

schematron-quickfix.github.io/sqf

Used in Multiple Domains

- Financial
- Insurance
- Government
- Technical publishing



Users

- Content Writer – A subject matter expert, but doesn't know XML very well
- XML Expert – Knows XML structure, but doesn't know much about the actual content

Using the quick fixes created by the XML Expert the Content Writer can correct the problem

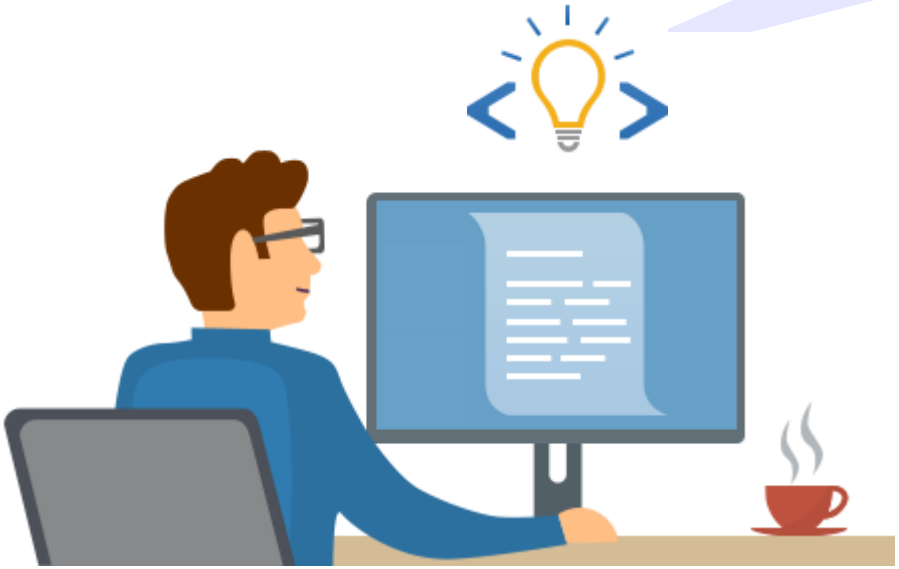
Use Cases

Generate Valid ID

All sections should have an ID

Add ID to the current section

Add ID to the all sections from document



Generate ID for Current Section

- Use the title value as ID or generate a random ID

```
<sqf:fix id="addID">
  <sqf:description>
    <sqf:title>Add ID to the current section</sqf:title>
  </sqf:description>


  <sqf:add node-type="attribute" target="id"
    select="if (exists(title) and string-length(title) > 0)
    then substring(lower-case(replace(replace(normalize-space(string(title)), '\s', '_'),
'[^a-zA-Z0-9_]', '')), 0, 50)
    else generate-id()"/>
</sqf:fix>
```

Generate ID for all Sections

- Match all the section elements from the document and add an ID

```
<sqf:fix id="addIDs">
  <sqf:description>
    <sqf:title> Add ID to the all entries from document</sqf:title>
  </sqf:description>


  <sqf:add match="//section[not(@id)]" node-type="attribute" target="id"
    select="if (exists(title) and string-length(title) > 0)
    then substring(lower-case(replace(replace(normalize-space(string(title)), '\s', '_'),
'[^a-zA-Z0-9_]', '')), 0, 50)
    else generate-id()"/>
</sqf:fix>
```

Use Current File Name as ID

- Get the current document file name and use it as ID

```
<sqf:fix id="addFileID">
  <sch:let name="reqId" value="substring-before(tokenize(document-uri(/), '/')[last()], '.')"/>
  <sqf:description>
    <sqf:title>Set "<sch:value-of select="$reqId"/>" as ID</sqf:title>
  </sqf:description>
  <sqf:add node-type="attribute" target="id" select="$reqId"/>
</sqf:fix>
```

Conclusion

- Use XPath expressions to generate a valid ID
- Change the matching context and add an ID for all section elements

Automatic Tagging

Link detected in text

Convert text link to xref

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

Automatic Tagging

```
<sqf:fix id="convertToLink">
  <xsl:variable name="linkValue">
    <xsl:analyze-string select="." regex="(http|www)\S+">
      <xsl:matching-substring>
        <xsl:value-of select="regex-group(0)"/>
      </xsl:matching-substring>
    </xsl:analyze-string>
  </xsl:variable>

  <sqf:description>
    <sqf:title>Convert '<sch:value-of select="$linkValue"/>' text link to xref</sqf:title>
  </sqf:description>
  <sqf:stringReplace regex="(http|www)\S+">
    <xref href="{$linkValue}" format="html"/>
  </sqf:stringReplace>
</sqf:fix>
```

Automatic Tagging

- Get the current link value

```
<sqf:fix id="convertToLink">
  <sqf:description>
    <sqf:title>Convert text link to xref</sqf:title>
  </sqf:description>
  <sqf:stringReplace regex="(http|www)\S+">
    <xref href="{regex-group(0)}" format="html"/>
  </sqf:stringReplace>
</sqf:fix>
```

- Get the current link value in Oxygen

```
<sqf:fix id="convertToLink">
  <sqf:description>
    <sqf:title>Convert text link to xref</sqf:title>
  </sqf:description>
  <sqf:stringReplace regex="(http|www)\S+">
    <xref href="$0" format="html"/>
  </sqf:stringReplace>
</sqf:fix>
```

Conclusion

- Use regular expressions to match text content
- Replace the text content with tags

Add Missing Cells

Cells are missing

Add enough empty cells on each row

Media	Description		
mp3	Moving Picture Experts Group Layer-3 Audio	audio	
wav			
pcm	audio	Width	
embedded video	Embedded Iframe Code	iframe	Width & Height

Add Missing Cells

```
<sqf:fix id="addCells">
  <sqf:description>
    <sqf:title>Add enough empty cells on each row</sqf:title>
  </sqf:description>

  <sch:let name="reqColumsNo" value="max(.//row/count(entry))"/>
  <sqf:add match="row" position="last-child">
      <sch:let name="columnNo" value="count(entry)"/>
      <xsl:for-each select="1 to xs:integer($reqColumsNo - $columnNo)">
         <entry/>
      </xsl:for-each>
  </sqf:add>
</sqf:fix>
```

Conclusion

- Perform complex operations
- Use XSLT to generate content

Ignore Schematron Check

The section should have an ID attribute

Ignore current rule

Ignore rule in the entire document

body section title **Introduction** title

p 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. p section

Ignore Rule Using PI

- Add a processing instruction with the ID of the rules

```
<?SuppressRule idCheck fullStopCheck boldCheck?>
```

Ignore Rule Fix

- Quick fix that adds a processing instruction with the rule ID before the current element

```
<sqf:fix id="ignoreRule" role="delete">
  <sqf:description>
    <sqf:title>Ignore current rule</sqf:title>
  </sqf:description>
  <sch:let name="ignoredRulePI"
           value="preceding-sibling::processing-instruction()[name() = 'SuppressRule']"/>

  <sqf:delete match="$ignoredRulePI" use-when="$ignoredRulePI"/>
  <sqf:add position="before">
    <xsl:processing-instruction name="SuppressRule"
                                select="concat($ignoredRulePI, ' ', $ruleCheckId)"/>
  </sqf:add>
</sqf:fix>
```

Ignore Rule Check

- Function that checks if there is a PI with the rule ID before the current node or at the end of the document

```
<xsl:function name="func:isRuleIgnored" as="xs:boolean">
  <xsl:param name="node"/>
  <xsl:param name="ruleId"/>
  <xsl:value-of
    select="($ruleId = $node/preceding-sibling::processing-instruction()[name() =
'SuppressRule']/tokenize(., ' ')
      or $ruleId = $globalIgnoredRules)"
  />
</xsl:function>
```

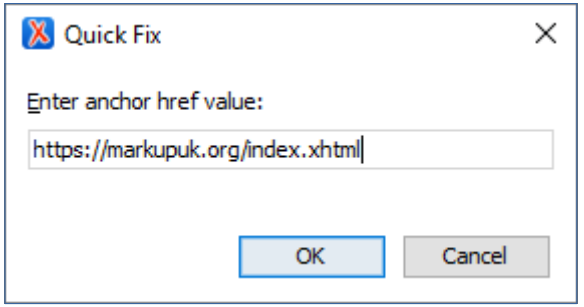
Conclusion

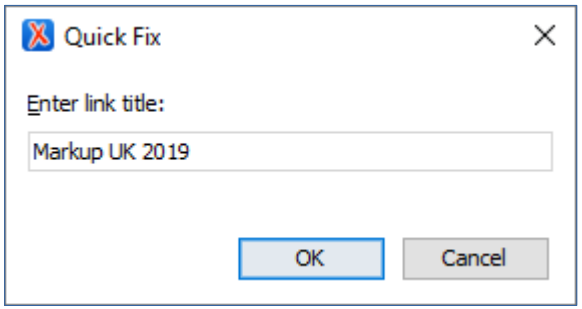
- Use Schematron quick fixes to implement an ignore rule mechanism
  - Ignore current rule
  - Ignore current rule in the entire document

Display Dialogs

Link is required for images

Surround image with a hypertext link





User Entry

```
<sqf:fix id="addAnchorAndTitle">
   <sqf:description>
      <sqf:title>Surround image with a hypertext link</sqf:title>
   </sqf:description>
   <sqf:user-entry name="href">
      <sqf:description>
         <sqf:title>Enter anchor href value:</sqf:title>
      </sqf:description>
   </sqf:user-entry>
   <sqf:user-entry name="linkTitle">
      <sqf:description>
         <sqf:title>Enter link title:</sqf:title>
      </sqf:description>
   </sqf:user-entry>
   <sqf:replace>
      <a href="{$href}" title="{$linkTitle}">
         <xsl:copy-of select="."/>
      </a>
   </sqf:replace>
</sqf:fix>
```

Java Dialog

```
<sqf:fix id="addSrc">
    <sqf:description>
        <sqf:title>Browse and add @src attribute</sqf:title>
    </sqf:description>
    <sqf:add node-type="attribute" target="src">
      <xsl:call-template name="browse"/>
    </sqf:add>
</sqf:fix>
```

```
<xsl:template name="browse">
   <xsl:variable name="dialog" select="jBrowse:new()"/>
   <xsl:variable name="result" select="jBrowse:showOpenDialog($dialog, $dialog)"/>
   <xsl:if test="$result = 0">
       <xsl:value-of select="jBrowse:getSelectedFile($dialog)"/>
   </xsl:if>
</xsl:template>
```

Conclusion

- Get the user input using “sqf:user-entry”
- Use java dialog to get the user input

Library of Quick Fixes

- A library of generic quick fixes, such as:
  - Wrap content in element
  - Unwrap element content
  - Rename element

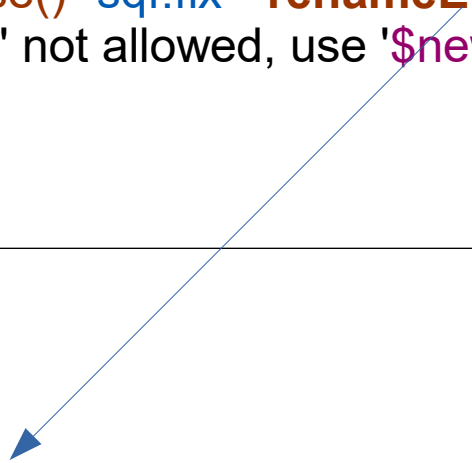
Abstract SQF

```
<sqf:fix id="mergeSiblings" role="replace">
   <sqf:param name="element" abstract="true"/>
   <sqf:description>
      <sqf:title>Merge flowing '$element' element into current one</sqf:title>
   </sqf:description>
   <sqf:add position="last-child" select="following-sibling::element()[1]/node()"/>
   <sqf:delete match="following-sibling::element()[1]"/>
</sqf:fix>
```

```
<sqf:fix id="renameElement" role="replace">
   <sqf:param name="element" abstract="true"/>
   <sqf:param name="newName" abstract="true"/>
   <sqf:description>
      <sqf:title>Rename '$element' element in '$newName'</sqf:title>
   </sqf:description>
   <sqf:replace match="." target="$newName" node-type="element" select="node()"/>
</sqf:fix>
```

Abstract Pattern

```
<sch:pattern id="renamePattern" abstract="true">
   <sch:rule context="$element">
      <sch:assert test="false()" sqf:fix="renameElement">
         Element '$element' not allowed, use '$newName' instead.
      </sch:assert>
   </sch:rule>
</sch:pattern>
```



Refers abstract quick fix

Abstract Pattern Implementation

```
<sch:pattern is-a="renamePattern">
   <sch:param name="element" value="orderedlist"/>
   <sch:param name="newName" value="itemizedlist"/>
</sch:pattern>
```

```
<sch:pattern is-a="renamePattern">
   <sch:param name="element" value="ol"/>
   <sch:param name="newName" value="ul"/>
</sch:pattern>
```

SQF Second Draft – March 2018

- Multilingual support for quick fixes
- Generate quick fixes dynamically
- Changed sqf:keep in sqf:copy-of
- Added @flags for sqf:stringReplace
- ...

Multilingual Support in SQF

- The name and description of a quick fix are defined by sqf:title and sqf:p elements
- The @ref attribute specifies IDs or keys for alternative localization

```
<sqf:fix id="addBone">
   <sqf:description>
      <sqf:title ref="fix_en fix_de">Add a bone</sqf:title>
      <sqf:p ref="fix_d_en fix_d_de">Add a bone as child element</sqf:p>
   </sqf:description>
   <sqf:add node-type="element" target="bone"/>
</sqf:fix>
```

Localization Using Diagnostics

- Implementation of quick fix localization using Schematron diagnostics
  - A diagnostic element is used for each language
  - The @ref attribute refers diagnostic IDs

```
<sqf:fix id="addBone">
   <sqf:description>
      <sqf:title ref="fix_en fix_de">Add a bone</sqf:title>
   </sqf:description>
   <sqf:add node-type="element" target="bone"/>
</sqf:fix>
....
<sch:diagnostics>
   <sch:diagnostic id="fix_en" xml:lang="en">Add a bone</sch:diagnostic>
   <sch:diagnostic id="fix_de" xml:lang="de">Fügen Sie einen Knochen hinzu</sch:diagnostic>
</sch:diagnostics>
```

Localization Using Property Files

- Implementation of quick fix localization using Java Property Files
  - A property file for each language “fileName_lang.xml”
  - The @ref attribute refers the property key

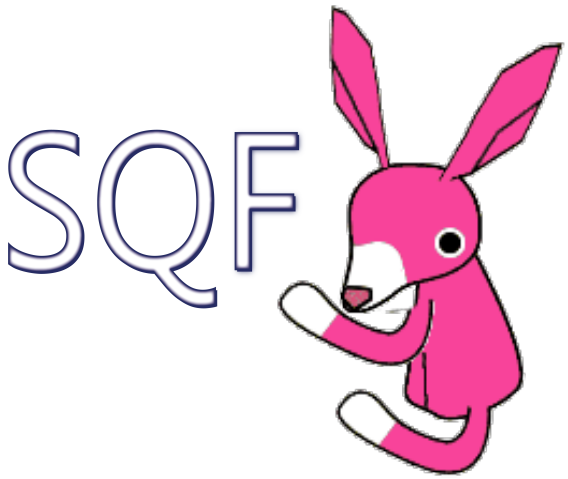
```
<sqf:fix id="addBone">
   <sqf:description>
      <sqf:title ref="dog.addBone.title" xml:lang="en">Add a bone</sqf:title>
      <sqf:p ref="dog.addBone.p">Add bone element as child</sqf:p>
   </sqf:description>
   <sqf:add node-type="element" target="bone"/>
</sqf:fix>

<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
   <entry key="sample.dog.addBone.title">Füge einen Knochen hinzu</entry>
   <entry key="sample.dog.addBone.p">Der Hund wird einen Knochen erhalten.</entry>
</properties>
```

Generate Quick Fixes Dynamically

- Added support to generate quick fixes dynamically using the @use-for-each attribute
- Generate a quick fix for each match of the @use-for-each attribute

```
<sqf:fix id="removeAnyItem" use-for-each="1 to count(li)">
   <sqf:description>
      <sqf:title>Remove item #<sch:value-of select="$sqf:current"/></sqf:title>
   </sqf:description>
   <sqf:delete match="li[$sqf:current]"/>
</sqf:fix>
```

SQF Became a Powerful Language and made Schematron More Powerful



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

Resources

- Schematron Quick Fix specification
  http://schematron-quickfix.github.io/sqf
- Schematron Quick Fix official site
  http://www.schematron-quickfix.com/
- Schematron official site
  http://schematron.com/
- Schematron specification
  https://standards.iso.org/ittf/PubliclyAvailableStandards

THANK YOU!

Any questions?

octavian_nadolu@oxygenxml.com
@OctavianNadolu