Custom Business Rules for DITA Projects

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

About me

Software Architect at Syncro Soft

octavian.nadolu@oxygenxml.com

- 15+ years of XML technology experience
- Contributor for various XML-related open source projects
- Editor of Schematron QuickFix specification developed by a W3C community group

Agenda

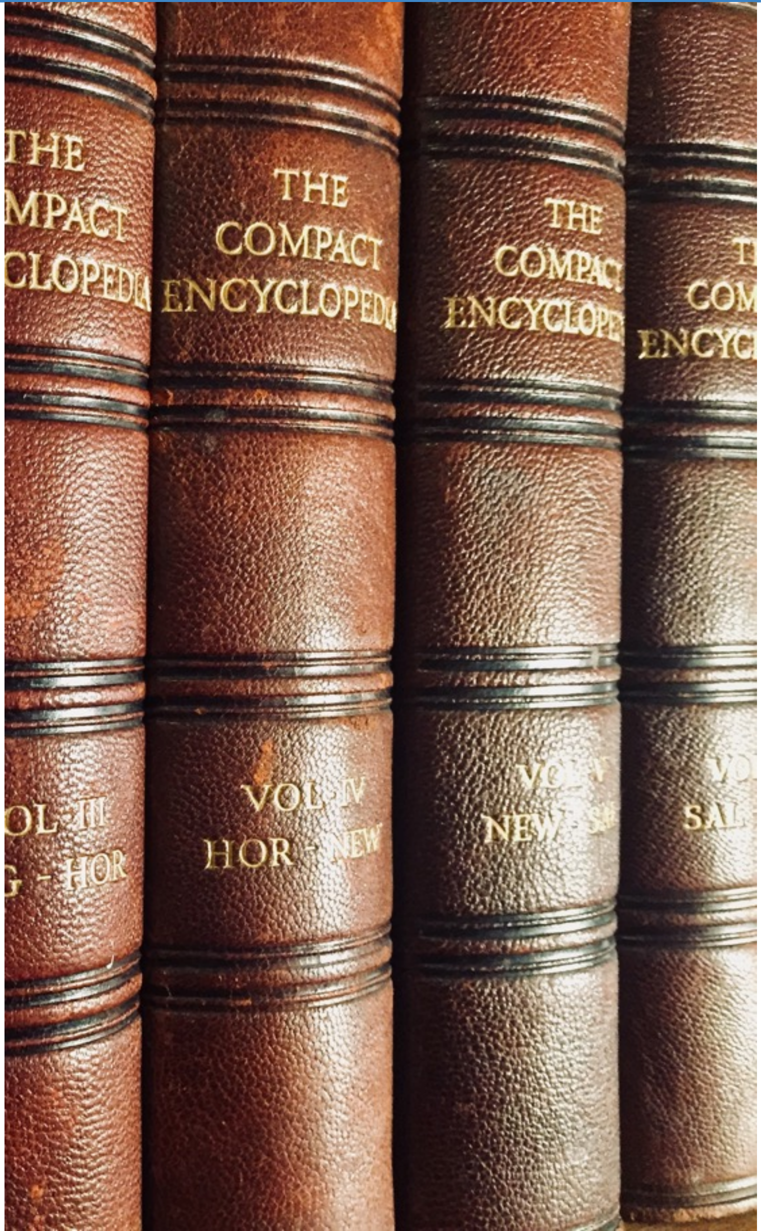
- Business rules with Schematron
- How to apply Schematron rules on DITA files
- Schematron rules for a documentation project
- Defining Schematron Quick Fix actions

What Are Business Rules?

- Rules that define or constrain some aspects of business and always resolve to true or false

- Intended to assert business structure or to control or influence the behavior of the business

- Helps the organization achieve its goals



Example of Business Rules for DITA

- **Titles** must be uppercase
- **Short descriptions** should not exceed 50 characters
- Avoid having **empty** DITA **elements**
- Avoid having a semi-colon at the end of a **list item**
- **Lists** should contain more than one item
- **Codeblocks** should have an @outputclass attribute set

Schematron

A natural language for making assertions in documents

schematron

A language for making assertions about patterns found in XML documents

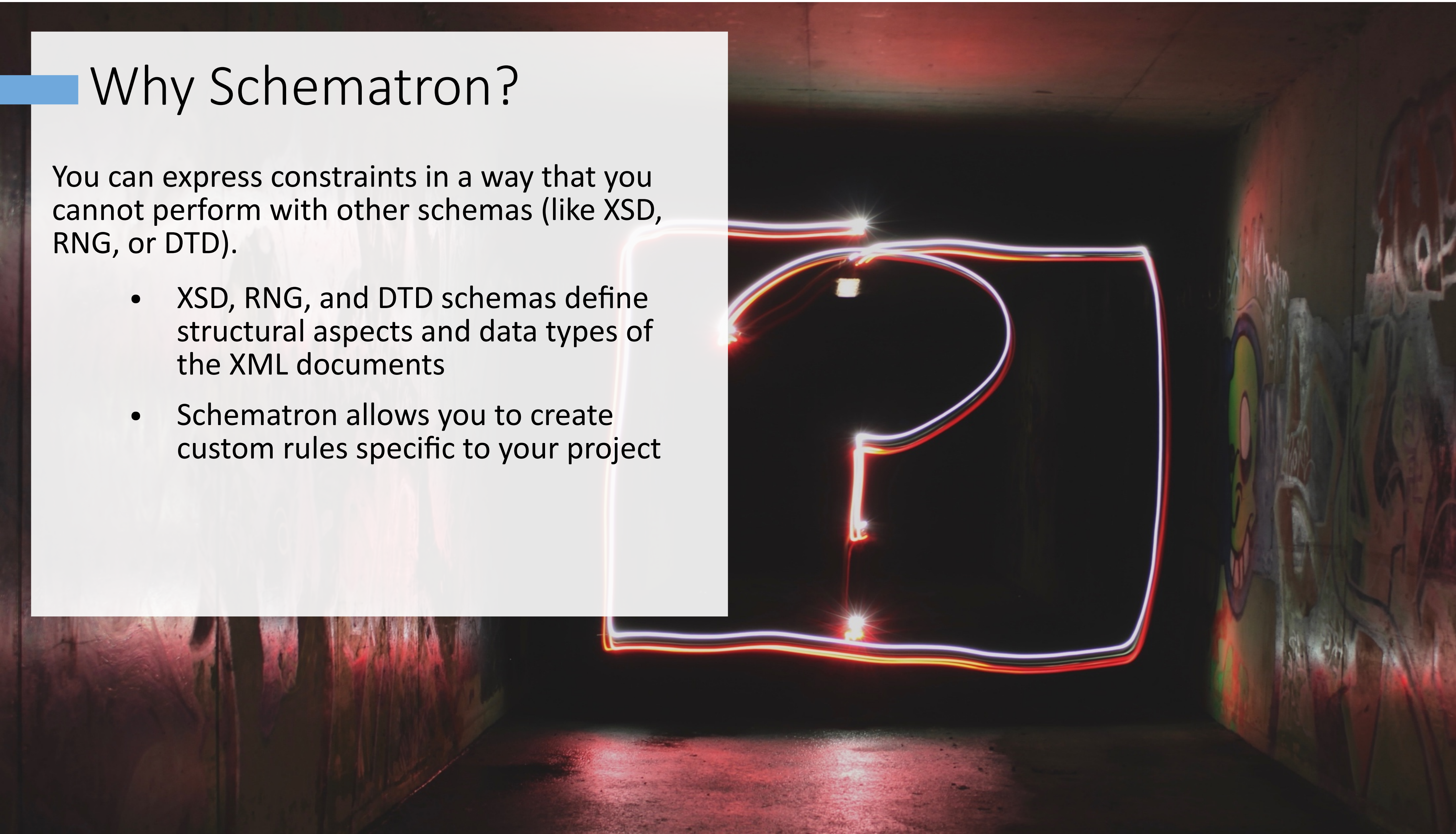
Schematron is an ISO Standard

Schematron is an ISO standard adopted by hundreds of major projects in the past decade

Schematron History

Schematron was invented by Rick Jelliffe

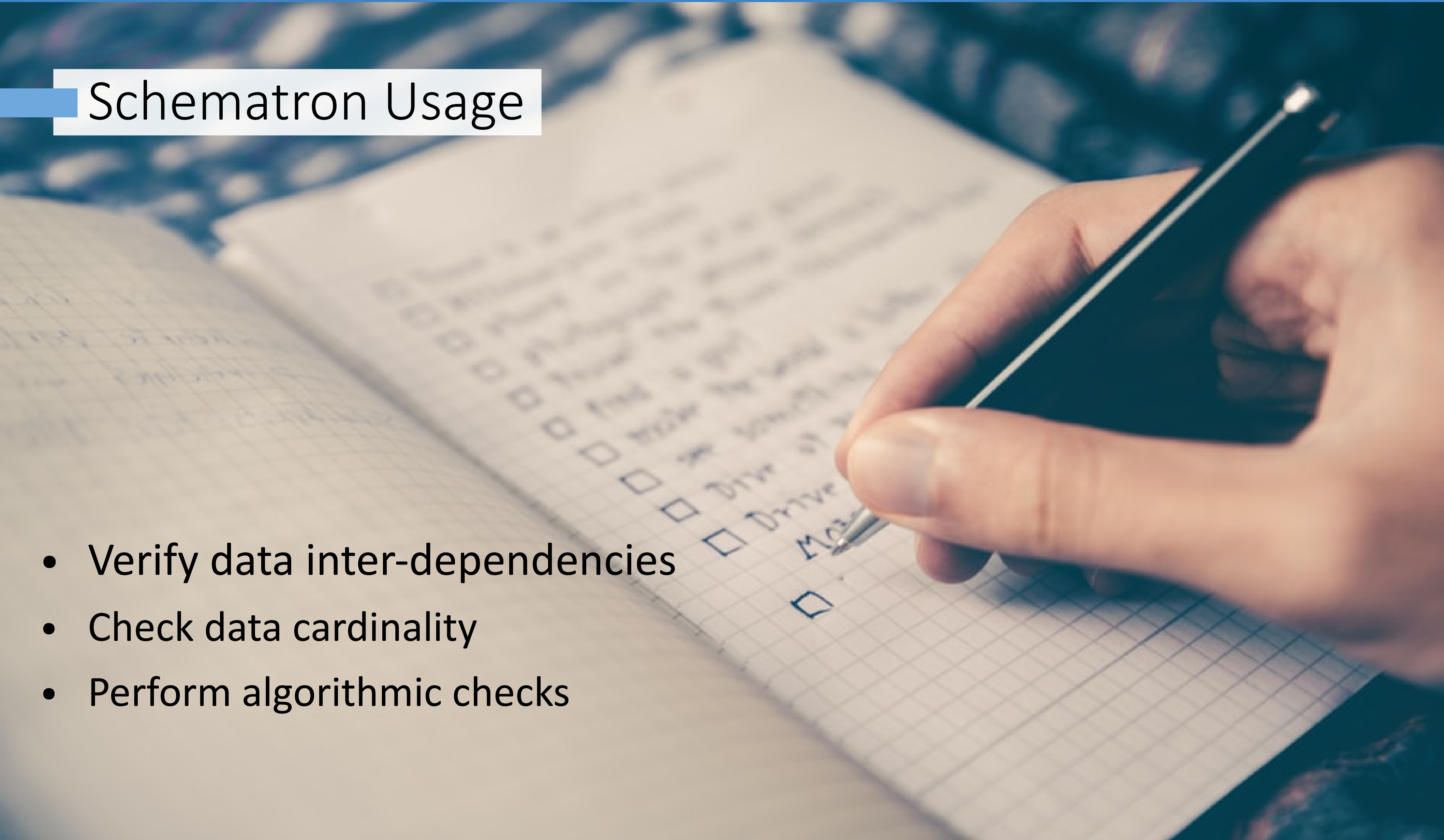
- Schematron 1.0 (1999), 1.3 (2000), 1.5 (2001)
- ISO Schematron (2006, 2010, 2016)



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

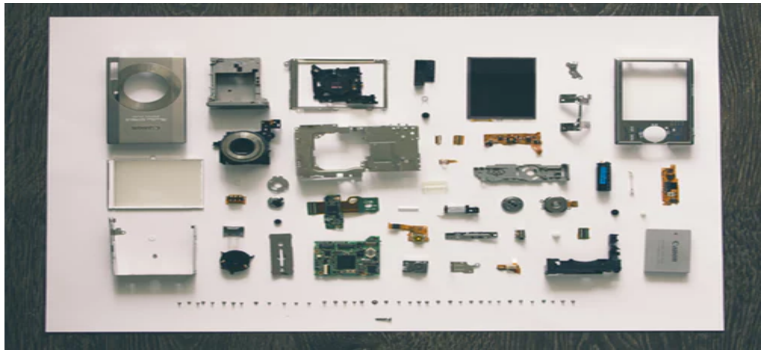


Schematron Usage

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

Used in Multiple Domains

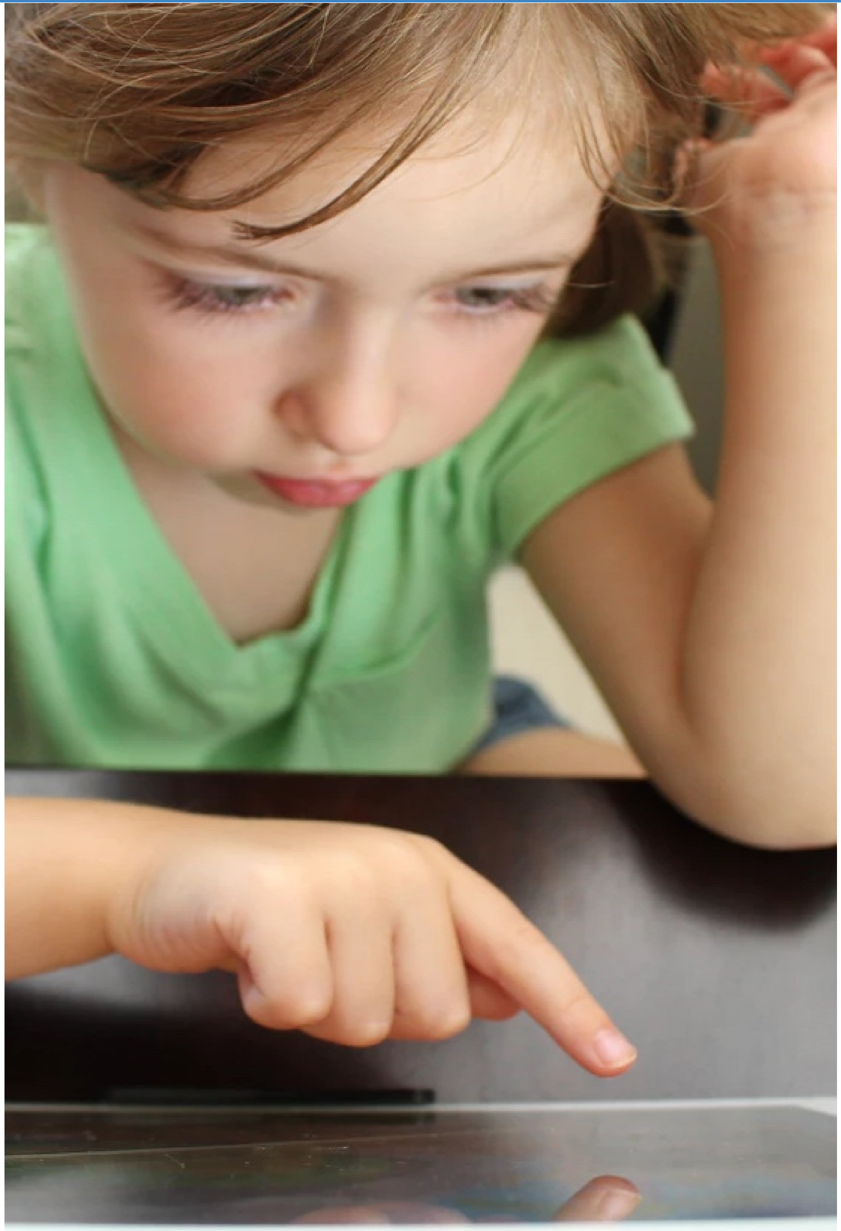
- Financial
- Insurance
- Government
- Technical publishing



Schematron is Very Simple

There are only 5 basic elements:

1. assert
2. report
3. rule
4. pattern
5. schema



<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>

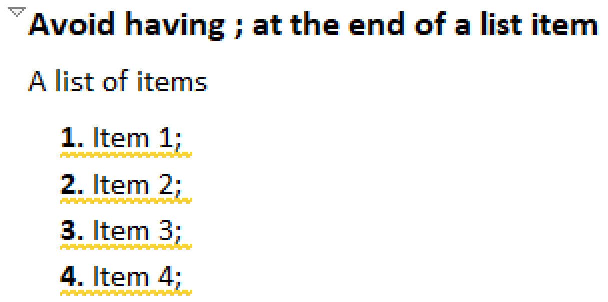
The top-level element of a Schematron schema.

```
<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>
```

Examples of Rules

Avoid Semi-Colons at the End of a List Item

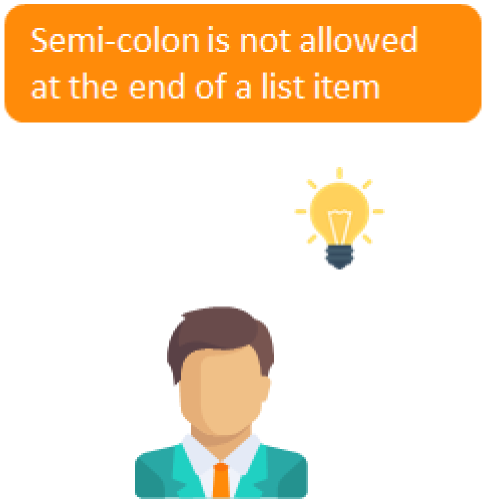
List items should not end with a semi-colon (;). If it is a complete sentence, then end it with a full stop (.), otherwise leave it without an ending character.



Avoid having ; at the end of a list item

A list of items

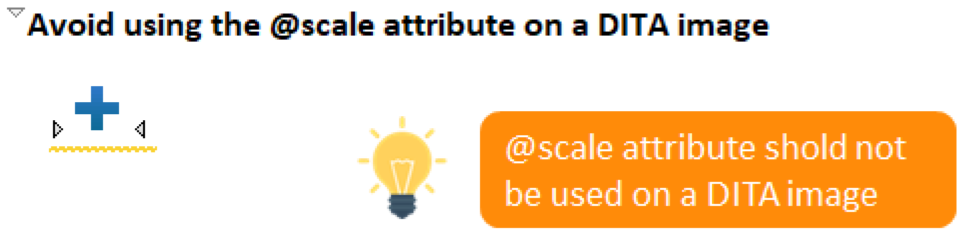
1. Item 1;
2. Item 2;
3. Item 3;
4. Item 4;



Avoid Using the @scale Attribute on a DITA Image

Dynamically scaled images are not properly displayed.

Scale the image with an image tool and keep it within the recommended width and height limits.

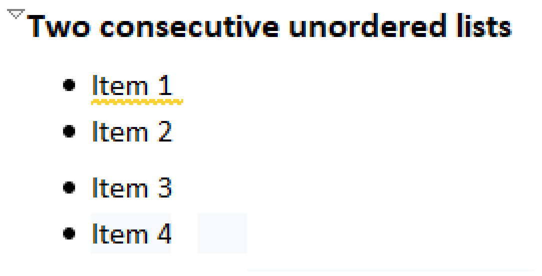


Avoid using the @scale attribute on a DITA image

@scale attribute shold not be used on a DITA image

Two Consecutive Lists

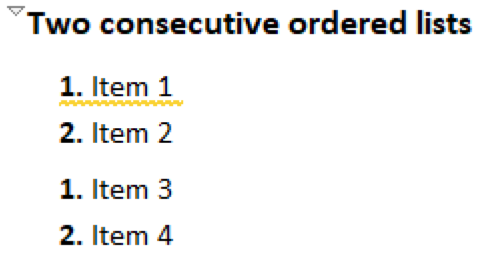
Here are two consecutive ordered lists. You can probably merge them into one or add a paragraph with a description between them.



Two consecutive unordered lists

- Item 1
- Item 2

- Item 3
- Item 4



Two consecutive ordered lists

1. Item 1
2. Item 2

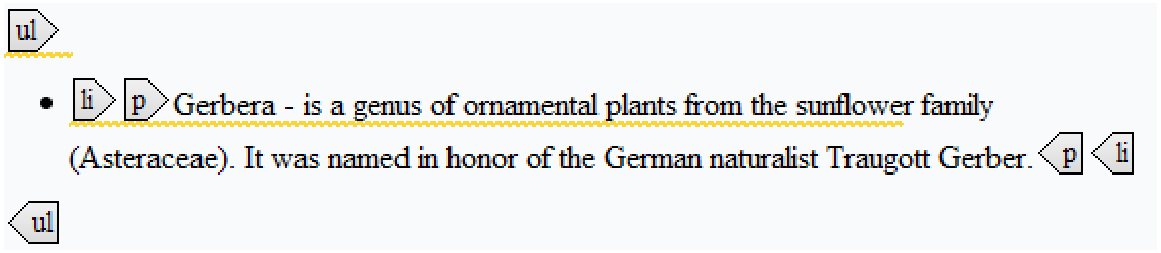
1. Item 3
2. Item 4

Avoid using two consecutive lists

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

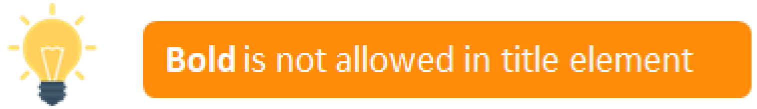
Check Number of List Items

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

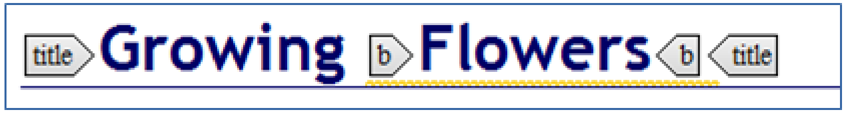
```
<rule context="ul">
   <assert test="count(li) > 1">
      A list must have more than one item</assert>
</rule>
```

Styling in Titles

Example rule that checks for styling in titles:



Bold is not allowed in title element



title Growing b Flowers b title

Styling in Titles

There should be no markup inside a title:

```
<rule context="title/*">
   <report test="name() != 'ph'">
     No elements other than ‘ph’ are allowed inside titles</report>
</rule>
```

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

Run Schematron Validation

- 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

Type of Problems

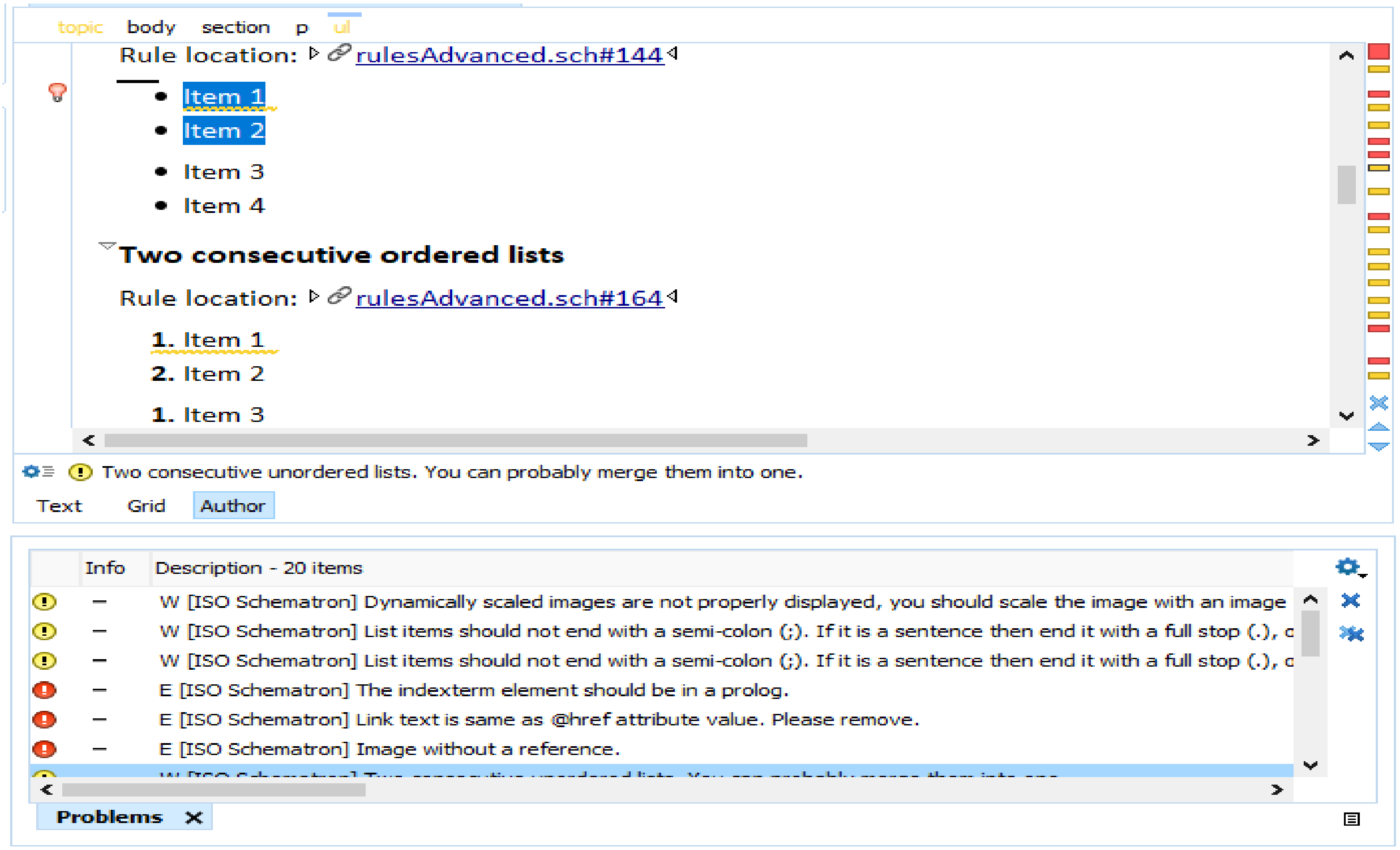
- Different type of problems
  - Code smell
  - Bug
  - Vulnerability
- Different severity
  - Fatal
  - Error
  - Warn
  - Info

Schematron Messages Severity

Severity level can be set in the value of the @role attribute from an assert or report element (possible values: fatal, error, warn, info)

```
<rule context="linklist">
  <assert test="title" role="warn">
    The linklist should have a title</assert>
</rule>
```

Validation Result in Editing Framework



topic body section p ul

Rule location: rulesAdvanced.sch#144

- Item 1
- Item 2
- Item 3
- Item 4

Two consecutive ordered lists

Rule location: rulesAdvanced.sch#164

1. Item 1
2. Item 2

1. Item 3

Two consecutive unordered lists. You can probably merge them into one.

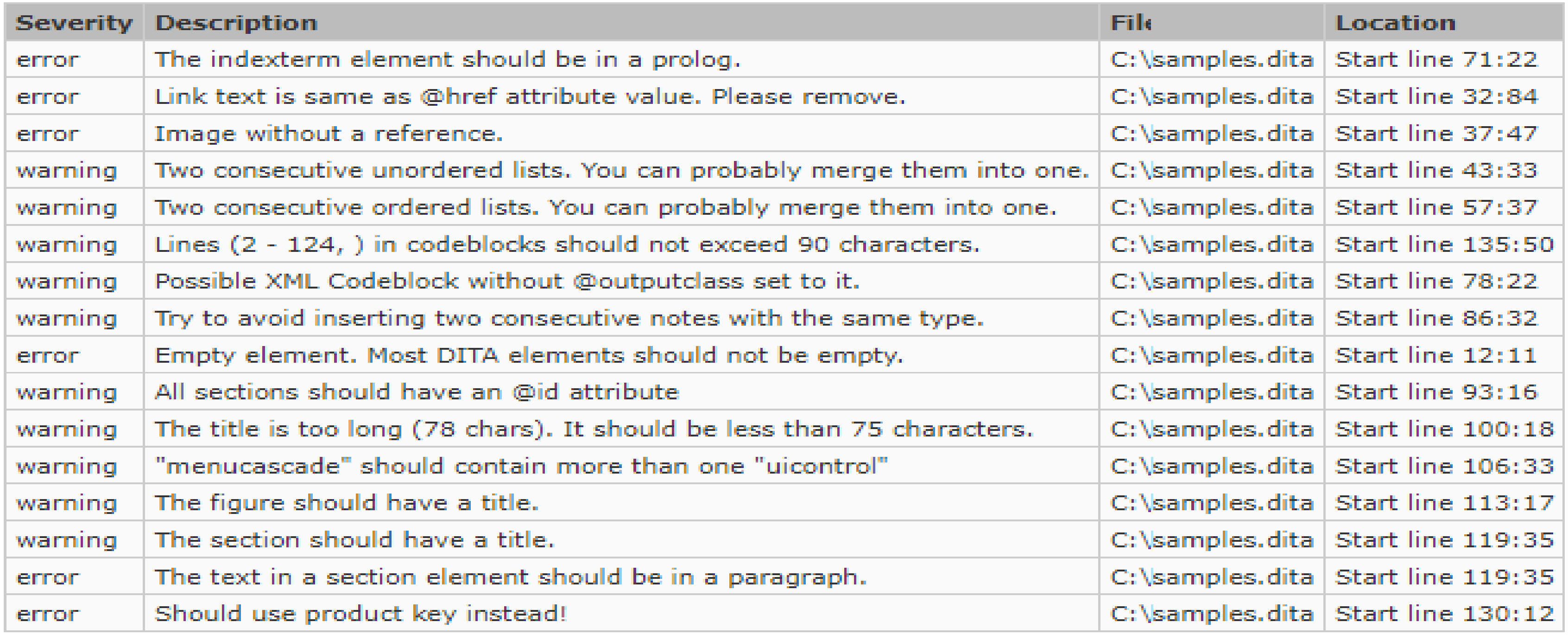
Text Grid Author

	Info	Description - 20 items
	–	W [ISO Schematron] Dynamically scaled images are not properly displayed, you should scale the image with an image
	–	W [ISO Schematron] List items should not end with a semi-colon (;). If it is a sentence then end it with a full stop (.), c
	–	W [ISO Schematron] List items should not end with a semi-colon (;). If it is a sentence then end it with a full stop (.), c
	–	E [ISO Schematron] The indexterm element should be in a prolog.
	–	E [ISO Schematron] Link text is same as @href attribute value. Please remove.
	–	E [ISO Schematron] Image without a reference.

Problems

Validation Result as HTML

Export validation result in an HTML file



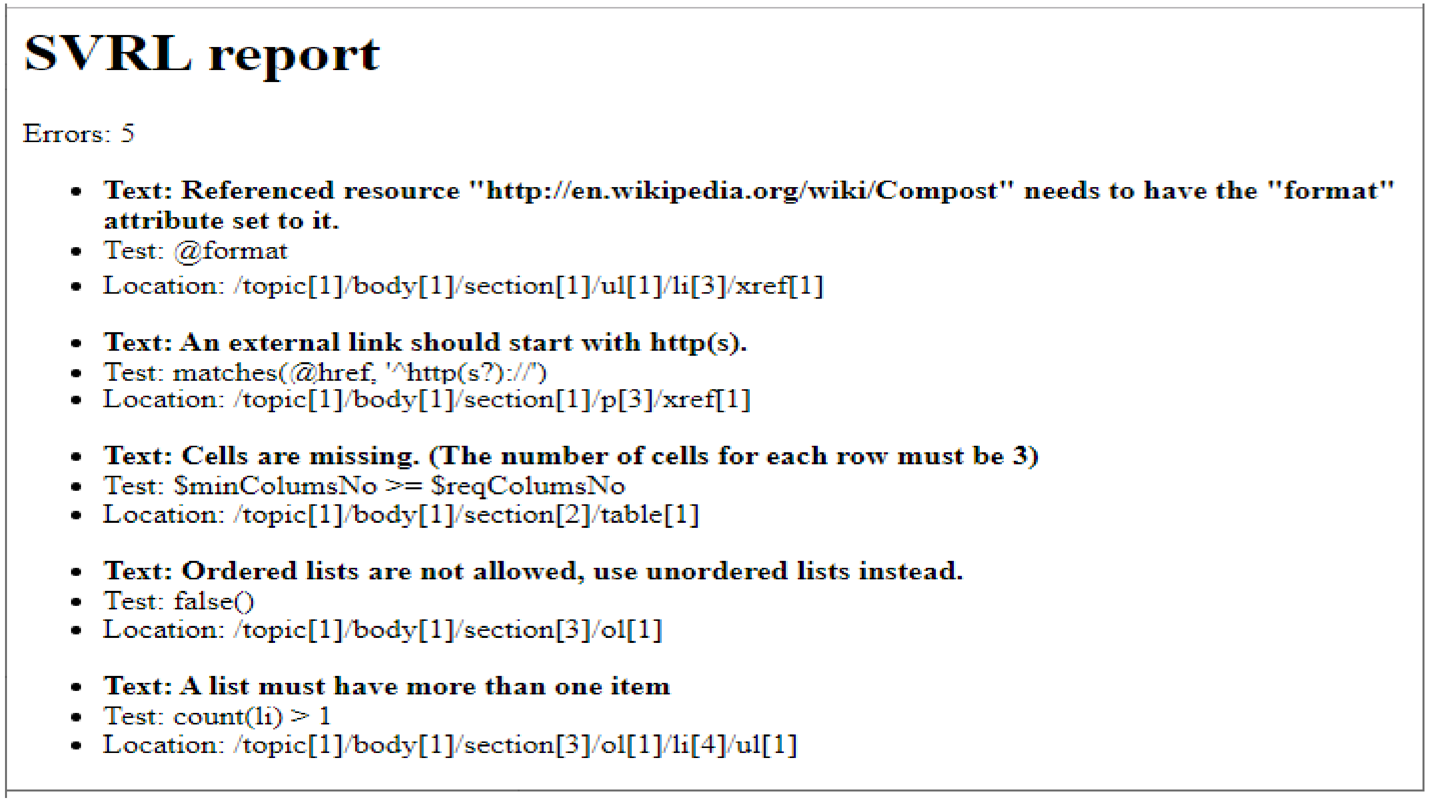
Severity	Description	File	Location
error	The indexterm element should be in a prolog.	C:\samples.dita	Start line 71:22
error	Link text is same as @href attribute value. Please remove.	C:\samples.dita	Start line 32:84
error	Image without a reference.	C:\samples.dita	Start line 37:47
warning	Two consecutive unordered lists. You can probably merge them into one.	C:\samples.dita	Start line 43:33
warning	Two consecutive ordered lists. You can probably merge them into one.	C:\samples.dita	Start line 57:37
warning	Lines (2 - 124,) in codeblocks should not exceed 90 characters.	C:\samples.dita	Start line 135:50
warning	Possible XML Codeblock without @outputclass set to it.	C:\samples.dita	Start line 78:22
warning	Try to avoid inserting two consecutive notes with the same type.	C:\samples.dita	Start line 86:32
error	Empty element. Most DITA elements should not be empty.	C:\samples.dita	Start line 12:11
warning	All sections should have an @id attribute	C:\samples.dita	Start line 93:16
warning	The title is too long (78 chars). It should be less than 75 characters.	C:\samples.dita	Start line 100:18
warning	"menucascade" should contain more than one "uicontrol"	C:\samples.dita	Start line 106:33
warning	The figure should have a title.	C:\samples.dita	Start line 113:17
warning	The section should have a title.	C:\samples.dita	Start line 119:35
error	The text in a section element should be in a paragraph.	C:\samples.dita	Start line 119:35
error	Should use product key instead!	C:\samples.dita	Start line 130:12

Validation Result as SVRL

- 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>
```

Generate HTML from SVRL



SVRL report

Errors: 5

- **Text: Referenced resource "http://en.wikipedia.org/wiki/Compost" needs to have the "format" attribute set to it.**
- Test: @format
- Location: /topic[1]/body[1]/section[1]/ul[1]/li[3]/xref[1]

- **Text: An external link should start with http(s).**
- Test: matches(@href, '^http(s?)://')
- Location: /topic[1]/body[1]/section[1]/p[3]/xref[1]

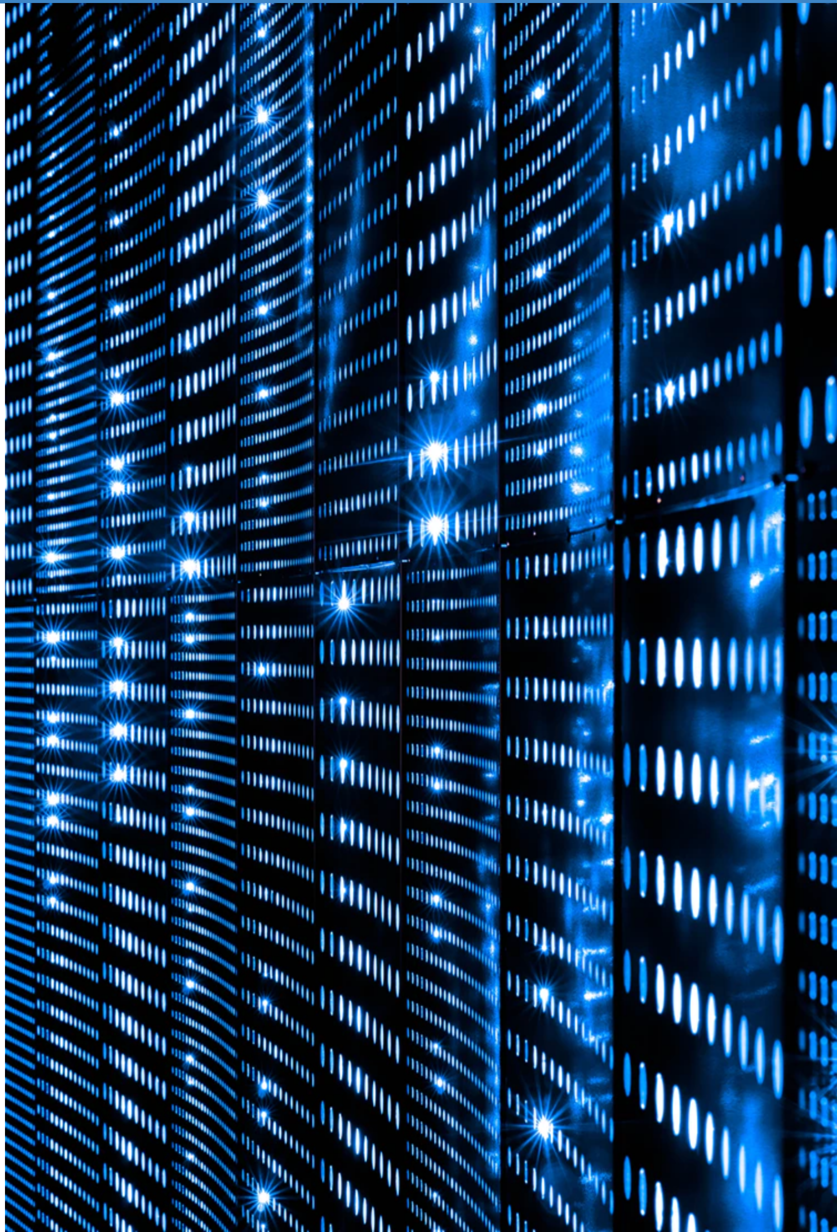
- **Text: Cells are missing. (The number of cells for each row must be 3)**
- Test: $minColumsNo >= $reqColumsNo
- Location: /topic[1]/body[1]/section[2]/table[1]

- **Text: Ordered lists are not allowed, use unordered lists instead.**
- Test: false()
- Location: /topic[1]/body[1]/section[3]/ol[1]

- **Text: A list must have more than one item**
- Test: count(li) > 1
- Location: /topic[1]/body[1]/section[3]/ol[1]/li[4]/ul[1]

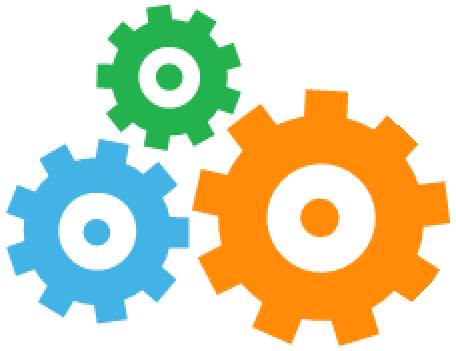
Integration server

- Use an integration server to run Schematron checks (such as Jenkins, Travis)
- Send a report to the user



Auto-Generate Schematron Rules

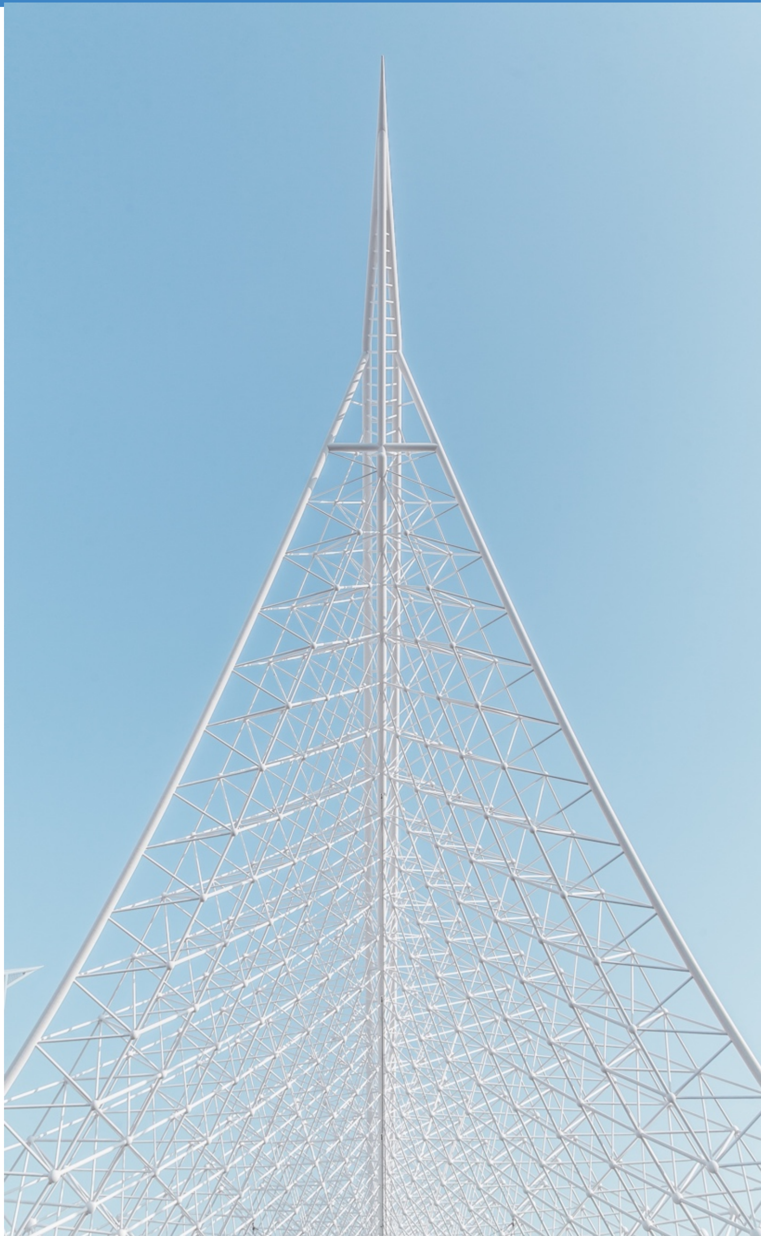
- A style guide can contain information to automate the style guide checks
- An automated rule is an instantiation of a generic rule



Intelligent Integrated Style Guide

- An implementation of an intelligent style guide
- Describes and enforces rules
- Open-source project is available on GitHub

github.com/oxygenxml/integrated-styleguide

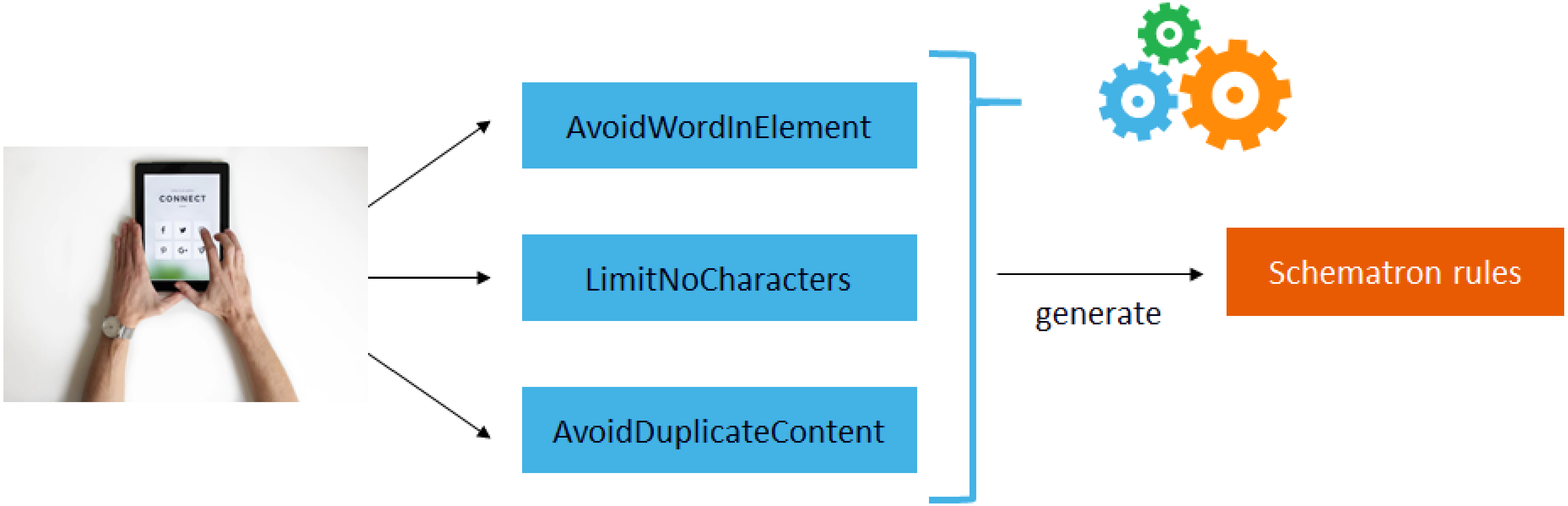


Library of Rules

- Generic rules can be created using a library of rules
- The user can choose the rule that they want to add:
  - Avoid a word in a certain element
  - Limit the number of characters
  - Avoid duplicate content

Abstract Patterns

Library of rules implemented using abstract patterns



Abstract Patterns

Allows to reuse patterns by making them generic

```
<pattern id="LimitNoOfWords" 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>
```

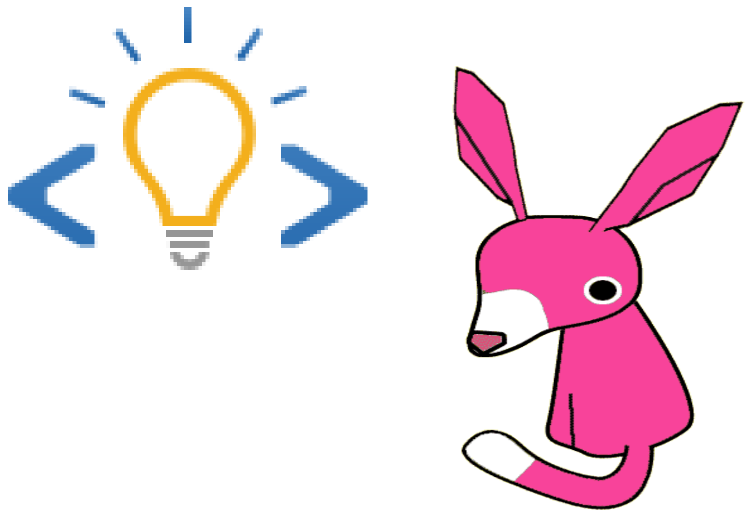
Pattern Instantiation

Refer an abstract pattern and specifies values for parameters

```
<pattern is-a="LimitNoOfWords">
   <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>
```

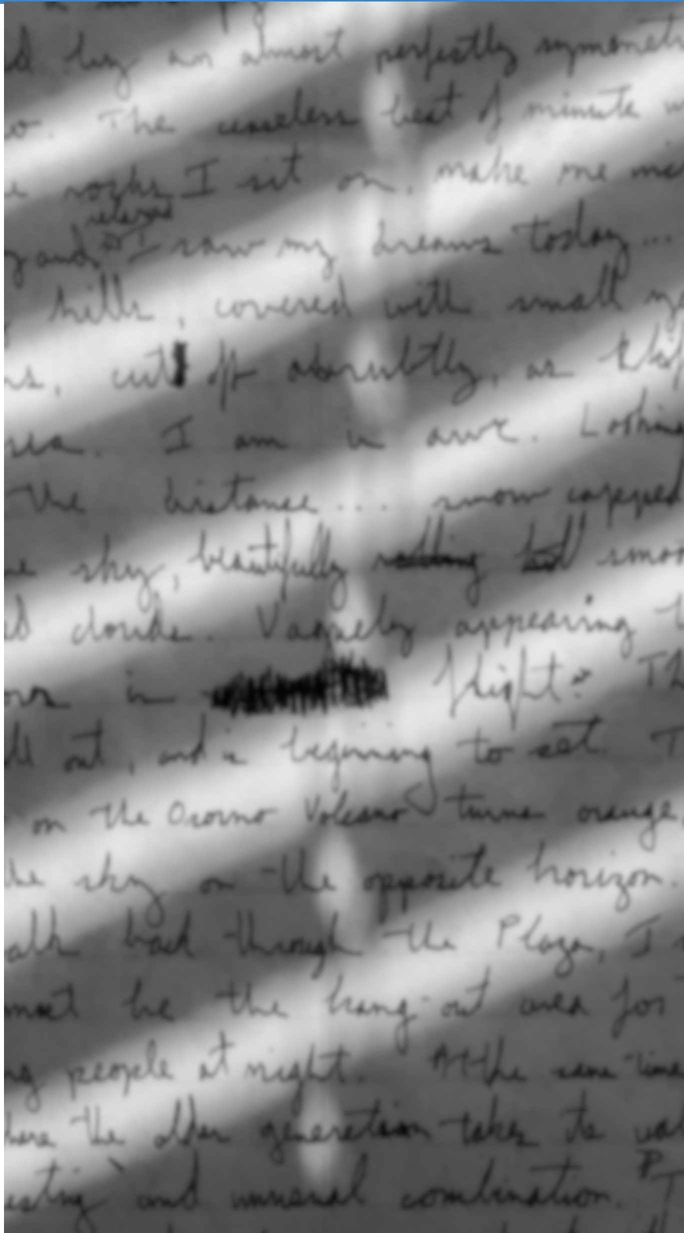
```
<pattern is-a="LimitNoOfWords">
   <param name="parentElement" value="title"/>
     <param name="minWords" value="1"/>
     <param name="maxWords" value="8"/>
     <param name="message" value="Keep titles between 1 and 8 words."/>
</pattern>
```

Schematron Quick Fixes



Schematron Quick Fixes

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



Type of 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 easily correct the problems.

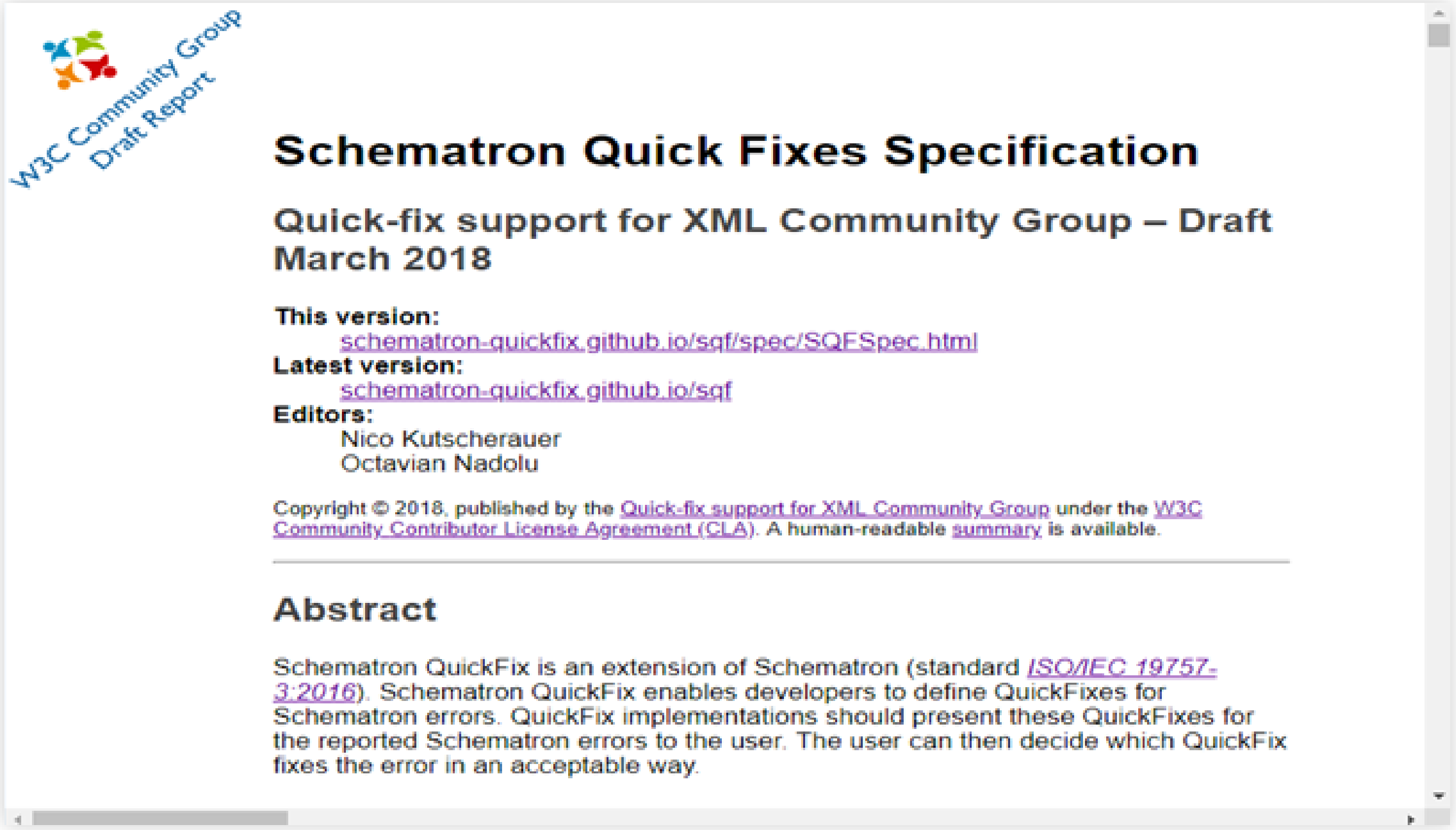
What Is SQF?

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

```
<rule context="html">
   <report test="//comment()" sqf:fix="removeComments">
         Comments are not allowed in document.</report>

      <sqf:fix id="removeComments" role="delete">
         <sqf:description><sqf:title>Remove all comments</sqf:title><sqf:description>
         <sqf:delete match="//comment()"/>
      </sqf:fix>
</rule>
```

Schematron Quick Fixes



Schematron Quick Fixes Specification

Quick-fix support for XML Community Group – Draft
March 2018

This version:
schematron-quickfix.github.io/sqf/spec/SQFSpec.html
Latest version:
schematron-quickfix.github.io/sqf
Editors:
Nico Kutscherauer
Octavian Nadolu

Copyright © 2018, published by the Quick-fix support for XML Community Group under the W3C Community Contributor License Agreement (CLA). A human-readable summary is available.

Abstract

Schematron QuickFix is an extension of Schematron (standard *ISO/IEC 19757-3:2016*). Schematron QuickFix enables developers to define QuickFixes for Schematron errors. QuickFix implementations should present these QuickFixes for the reported Schematron errors to the user. The user can then decide which QuickFix fixes the error in an acceptable way.

www.w3.org/community/quickfix
schematron-quickfix.github.io/sqf

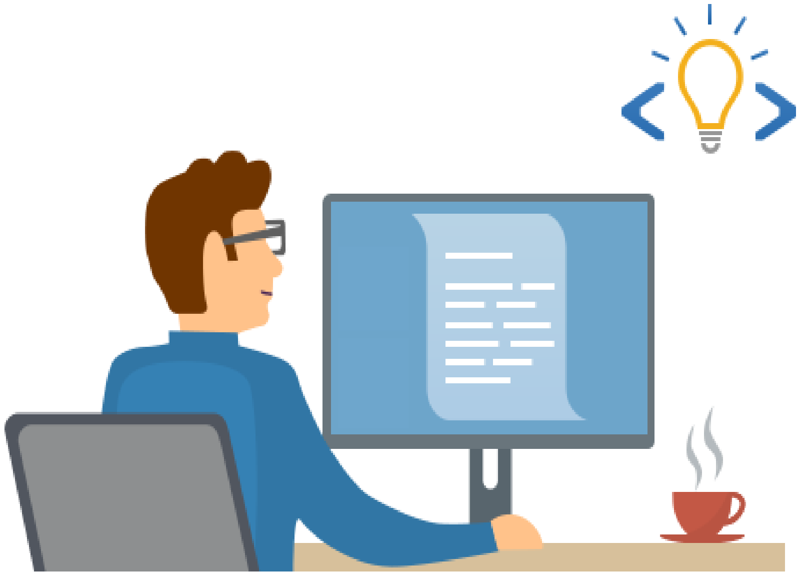
Use Cases

Generate Valid ID

All sections should have an ID

Add an ID to the current section

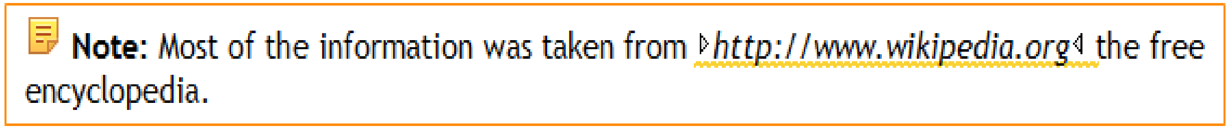
Add an ID to the all sections from document



Automatic Tagging

Link detected in text

Convert text link to xref

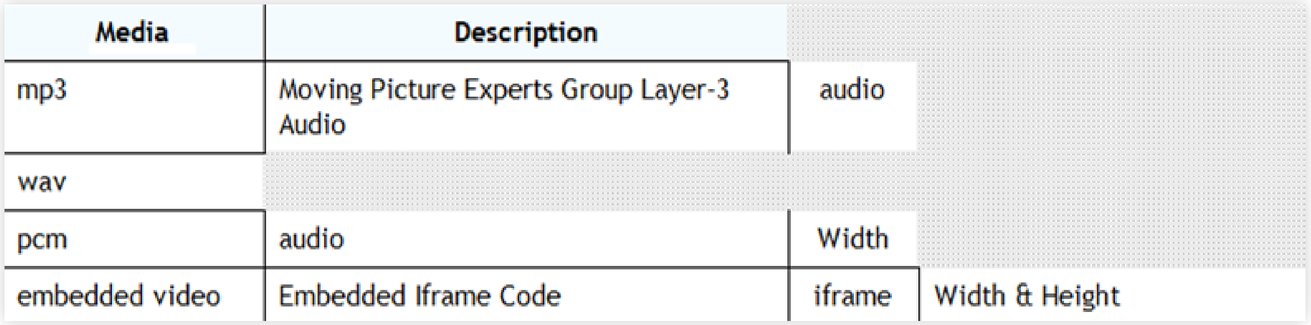


Note: Most of the information was taken from *http://www.wikipedia.org* the free encyclopedia.

Table Layout

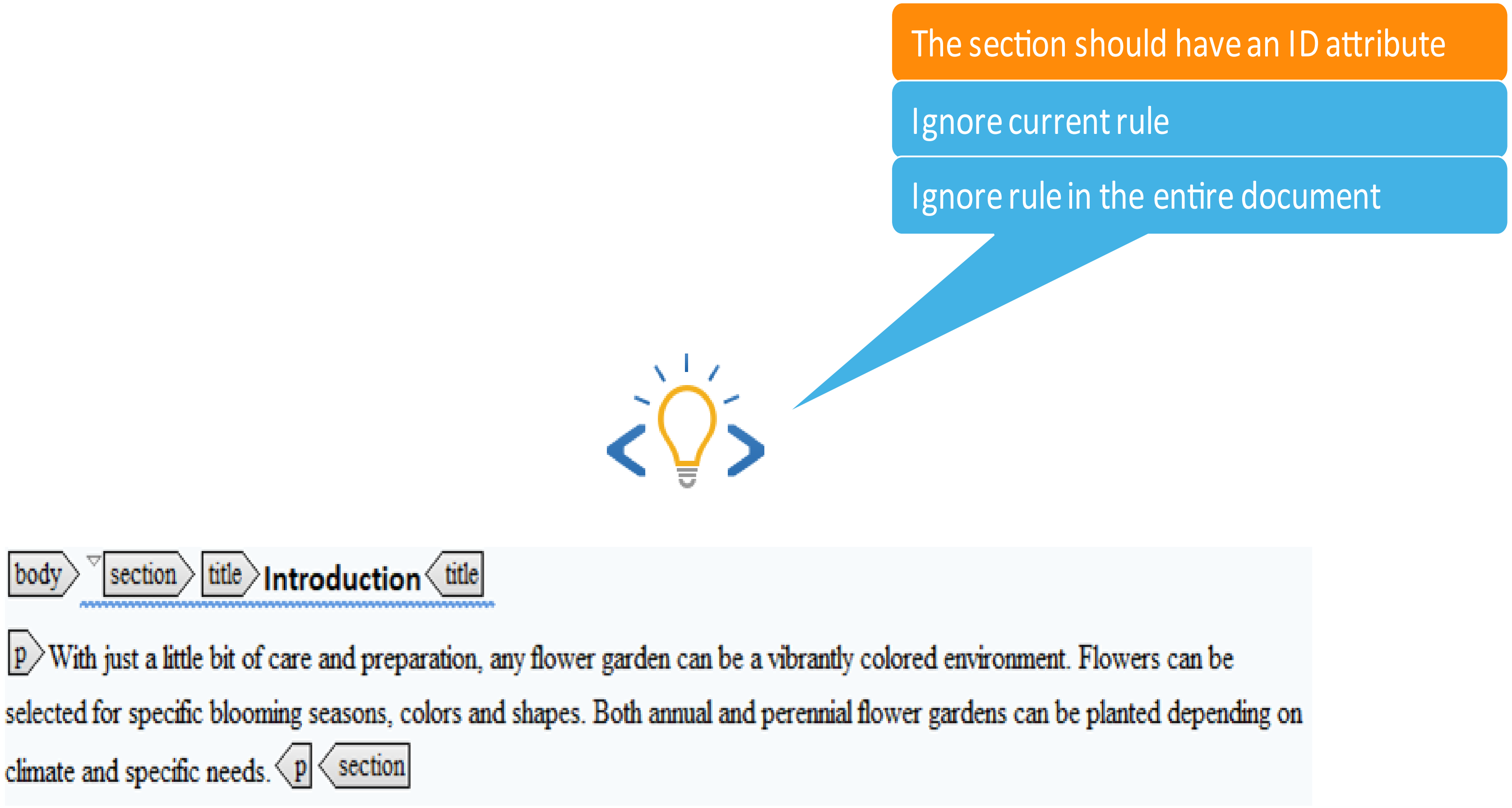
Cells are missing

Add enough empty cells to 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

Ignore Schematron Check

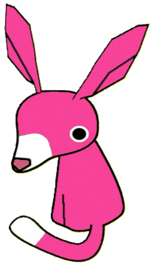

The section should have an ID attribute
Ignore current rule
Ignore rule in the entire document
body
section
title
title
p
p
section

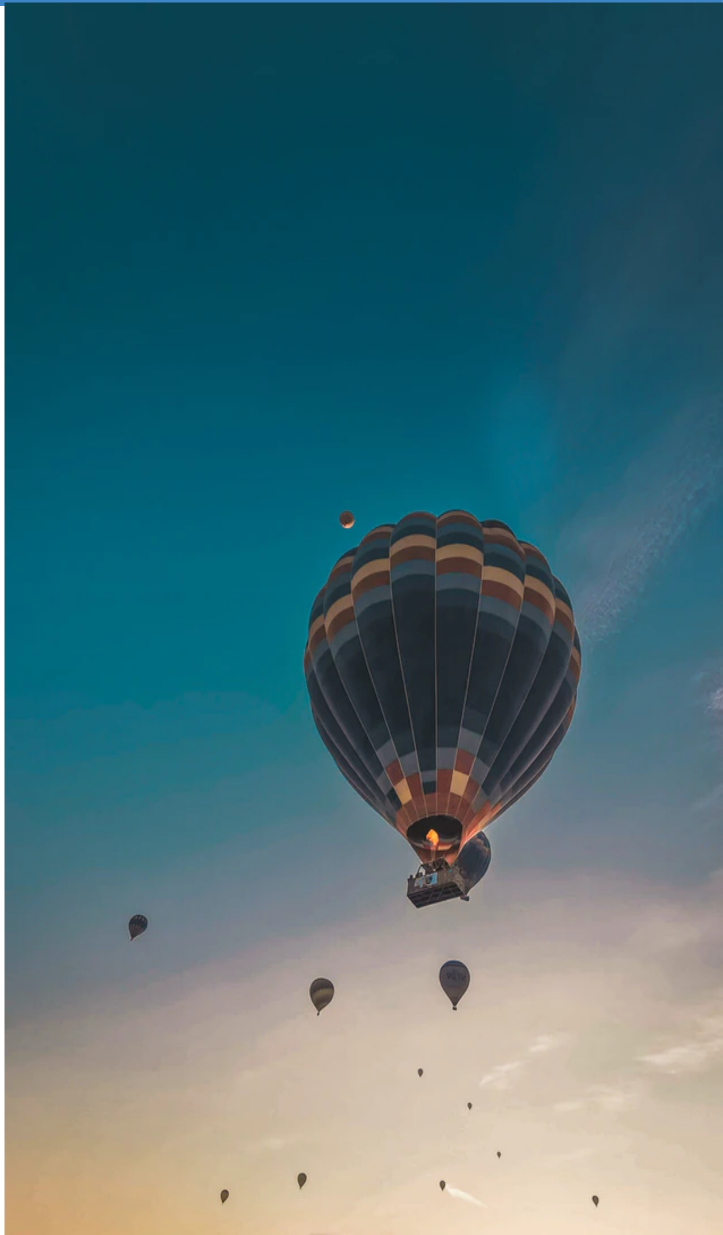
Introduction

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.

Conclusion

- Each project can have its own business rules
- Create business rules using Schematron
- Define SQF actions to correct the Schematron imposed rules
- Auto-generate rules from a library





Resources

- Schematron official site
- Schematron specification
- Schematron Quick Fix specification
- Examples of Schematron Rules
- Intelligent Integated Style Guide

Questions?

Octavian Nadolu
Software Architect at Syncro Soft

octavian.nadolu@oxygenxml.com
Twitter: @OctavianNadolu
LinkedIn: octaviannadolu