

User Assistance with Schematron

George Bina

@georgebina
george@oxygenxml.com



What is Schematron

- An ISO Standard
 - ISO/IEC 19757 – DSDL (Document Schema Definition Language) Part 3: Rule-based validation
- A very simple schema language
 - less than 10 main elements, about 20 elements in total
- A different kind of schema
 - defines business rules, not the document structure
 - the error messages are specified inside the schema
- Invented by Rick Jelliffe

Related technologies

- **XPath**

Used by Schematron to match and assert
- **XSLT**

Can be used to extend XSLT-based Schematron implementations
- **SQF**

Provide quick-fixes to identified issues defined as small scripts annotating the Schematron assertions

User Assistance

Help users
create correct documents
as they write

Cost for solving a problem



Try to solve the problems at the authoring time!

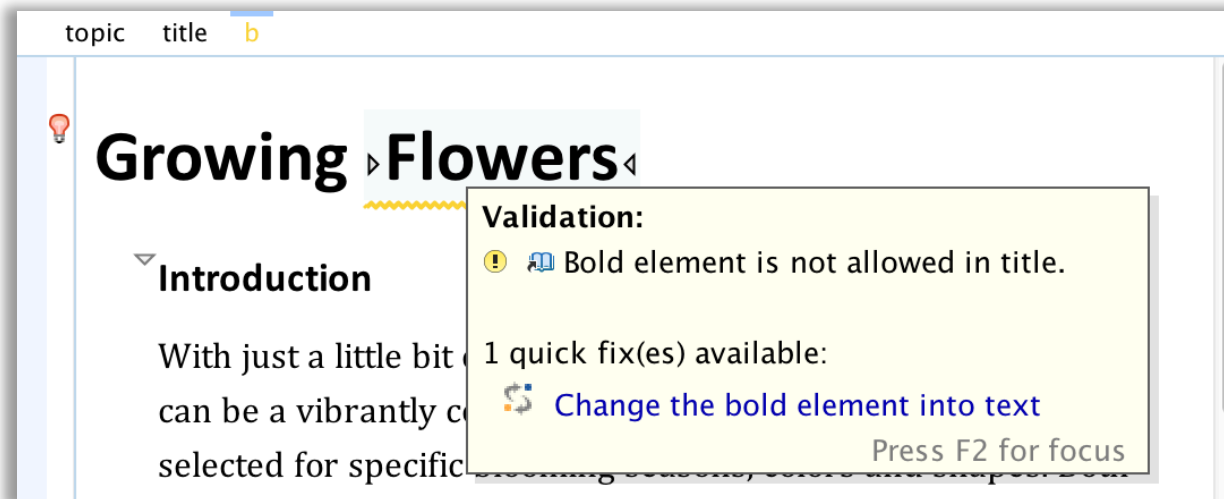
Assistance tools

Send messages to the user when some issues are detected in the edited document

- using different levels: information, warning, error, fatal
- provide links for more details using the @see attribute

Provide automatic solutions to detected issues

- Using SQF



UA use cases using Schematron

- Integrated intelligent style guide
- Learn DITA from a Markdown perspective

Integrated intelligent style guide

From Schematron to style guide

Generic Schematron code

Follow a coding pattern that separates roles

- abstract patterns
- instantiations of abstract patterns

Focus on making easy to instantiate abstract patterns

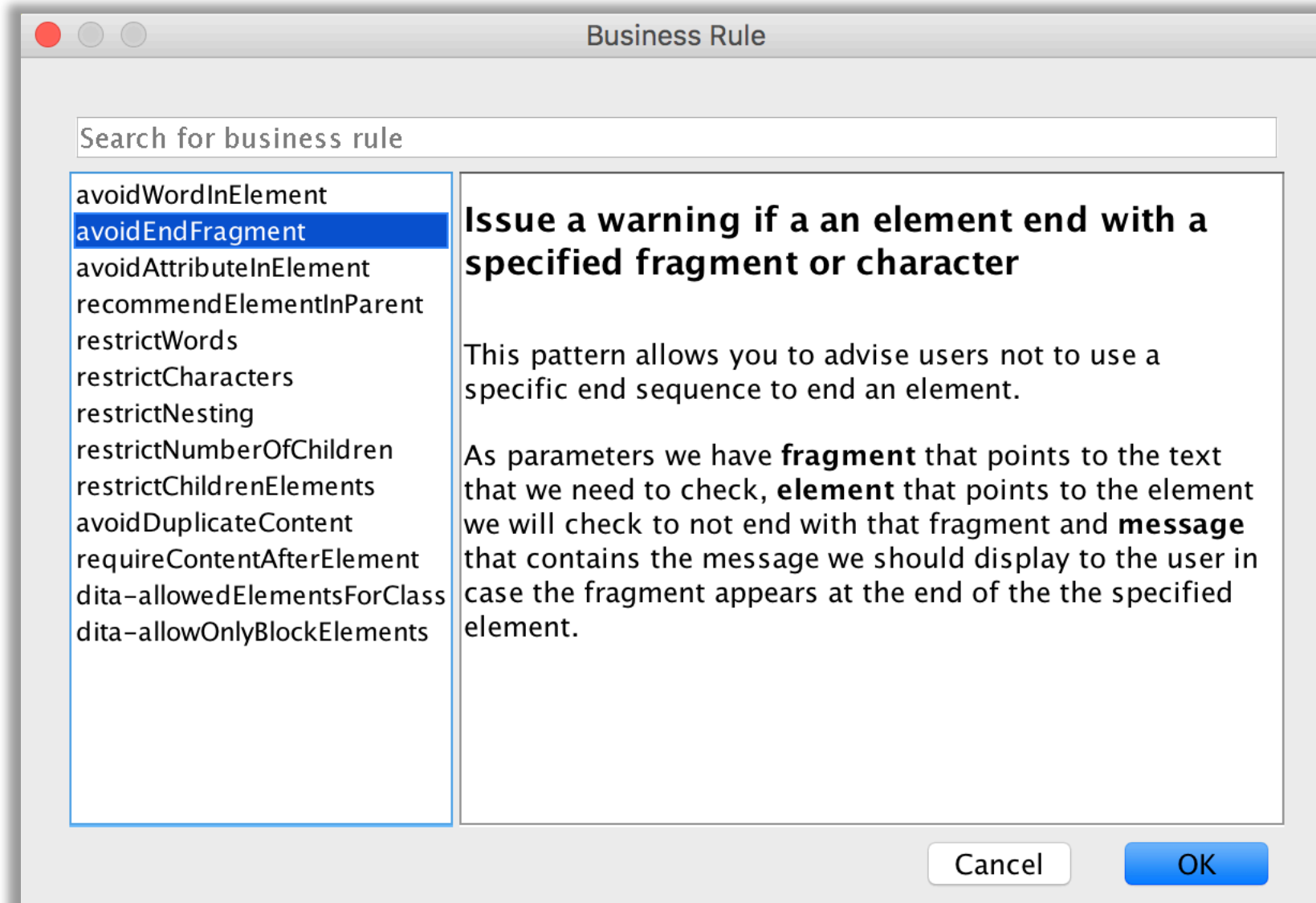
- embed the instantiations in a style guide using its language and generate the Schematron code
- annotate abstract patterns and provide support for looking up available abstract patterns

Sample rule


```
<pattern abstract="true" id="avoidEndFragment">
  <rule context="$element">
    <assert test="not(ends-with(normalize-space(.), '$fragment'))" role="warn"
      sqf:fix="avoidEndFragment_deleteFragment avoidEndFragment_replaceFragment">
      $message
    </assert>
  </rule>
</pattern>
```

```
<pattern is-a="avoidEndFragment"
  see="http://example.com/styleguide/webhelp/c_CreatingLists.html">
  <param name="element" value="li"/>
  <param name="fragment" value=";" />
  <param name="message" value="List items should not end with semicolon." />
</pattern>
```

Insert rule in style guide



Sample rule inside styleguide

- Include closing punctuation on list items only if all items in the list are complete sentences. This maximizes reuse potential when the context of a list item is not known.
- You can nest block elements, such as paragraphs and notes in list items, but limit their use to keep the list as simple, reusable, and scannable as possible.
- Capitalize the first word in each list item, unless the word should never be capitalized such as a list of commands.
- When including a list in a collection file, give each list item a unique ID so that each can be referenced separately as needed. See  [Reusing Elements Through Collection Files](#)⁴ for more information on using collection files.

| Rule | avoidEndFragment |
|----------|---|
| element | li |
| fragment | ; |
| message | List items should not end with semicolon. |



Use to add a new rule

Real-time notification

- Information Types
- Block Elements
- Creating Lists**
- Inline Elements
- Maps and Bookmarks
- Linking Strategies
- Attributes
- Reuse Strategy
- Content management

- Do not insert a list in the middle of a sentence.
- Do not write list items to complete an introductory sentence.
- Include closing punctuation on list items only if all items in the list are complete sentences. This maximizes reuse potential when the context of a list item is not known.
- You can nest block elements, such as paragraphs and notes in list items, but limit their use to keep the list as simple, reusable, and scannable as possible.
- Capitalize the first word in each list item, unless the word should never be capitalized such as a list of commands.
- When including a list in a collection file, give each list item a unique ID so that each can be referenced separately as

Info: This is a good example.

- to demonstrate my point
- to show a warning;

Validation:

List items should not end with semicolon.

2 quick fix(es) available:

- The fragment ";" will be deleted.
- The fragment ";" will be replaced by another end fragment.

Press F2 for focus

Intelligent style guide project

Dynamic Information Model (DIM)

<https://github.com/oxygenxml/dim>

- open source (Apache 2.0 license)
- sample style guide in DITA
- “dim” framework to allow creating rules
- XSLT scripts to generate Schematron from style guide

Demo

DIM style guide in action!

Learn DITA from a Markdown perspective

What is Markdown?

“Markdown is a text-to-HTML conversion tool for web writers”

“The single biggest source of inspiration for Markdown’s syntax is the format of plain text email”

“Markdown is free software, available under the terms of a BSD-style open source license”

Source: <https://daringfireball.net/projects/markdown/> by [John Gruber](#)

What is Markdown?

> Markdown is a text-to-HTML conversion tool for web writers

> The single biggest source of inspiration for Markdown's
> syntax is the format of plain text email

> Markdown is free software, available under the terms of
> a BSD-style open source license

Source: [<https://daringfireball.net/projects/markdown/>](https://daringfireball.net/projects/markdown/)
by [\[John Gruber\]\(https://daringfireball.net/projects/markdown/\)](https://daringfireball.net/projects/markdown/)

Path from Markdown to DITA

- Recognize Markdown fragments in DITA topics
- Convert them automatically to DITA markup

Example:

| | | | |
|----------|--------|----------|-----------------|
| * item 1 | | • item 1 | |
| * item 2 | —————> | • item 2 | item 1 |
| * item 3 | | • item 3 | item 2 |
| | | | item 3 |
| | | | |

<https://github.com/oxygenxml/ditaMark>

Technologies

Schematron

Detect Markdown fragments in DITA and notify the user that the corresponding DITA markup should be used instead

SQF

Provide the option to replace automatically the Markdown fragment with the corresponding DITA markup

XSLT

Used to support SQF actions which may imply complex changes to the document

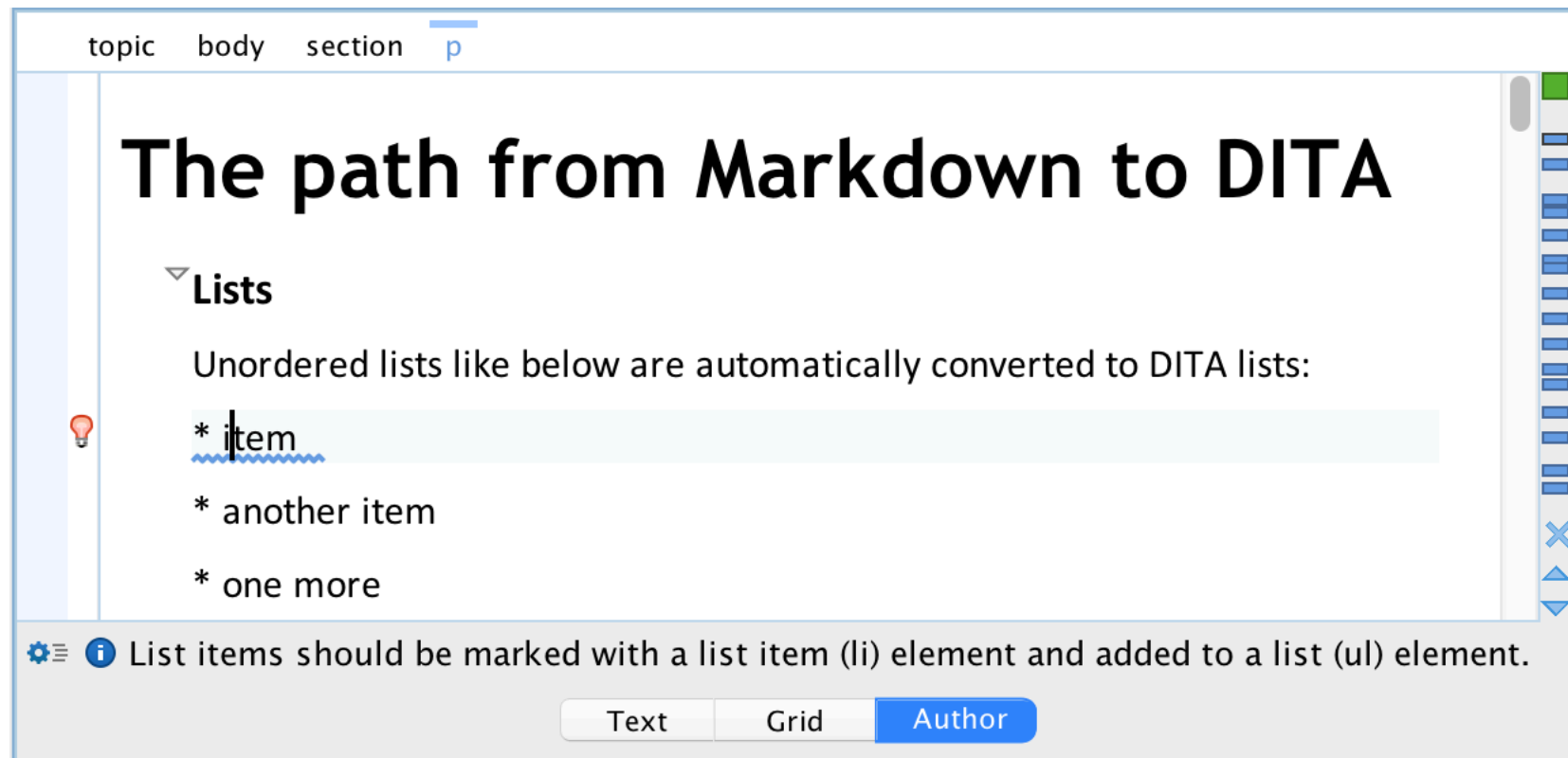
Recognized Markdown patterns

- Lists
 - * item or - item
- Quotes
 - > text
- Code blocks and inline code
 - ``` code and `inline code`
- Links
 - [link text](link URL) or <URL>
- Images
 - ![alternate text](URL) or ![alternate text](URL "title")
- Tables
 - | - | - | - | with or without a header
- Titles
 - # title or ## section

Demo

As you type Markdown to DITA conversions

Markdown lists to DITA

A screenshot of the Oxygen XML Editor interface. The top navigation bar shows "topic", "body", "section", and "p". The main content area displays a DITA document with a title "The path from Markdown to DITA" and a section titled "Lists". Below the section title, there is a paragraph: "Unordered lists like below are automatically converted to DITA lists:". This is followed by a list of three items: "* item", "* another item", and "* one more". The first item, "* item", is highlighted with a light blue background and has a red squiggly underline. A lightbulb icon is visible to the left of the list. At the bottom of the editor, there is a warning message: "List items should be marked with a list item (li) element and added to a list (ul) element." Below the warning message, there are three buttons: "Text", "Grid", and "Author".

topic body section p

The path from Markdown to DITA

▼ Lists

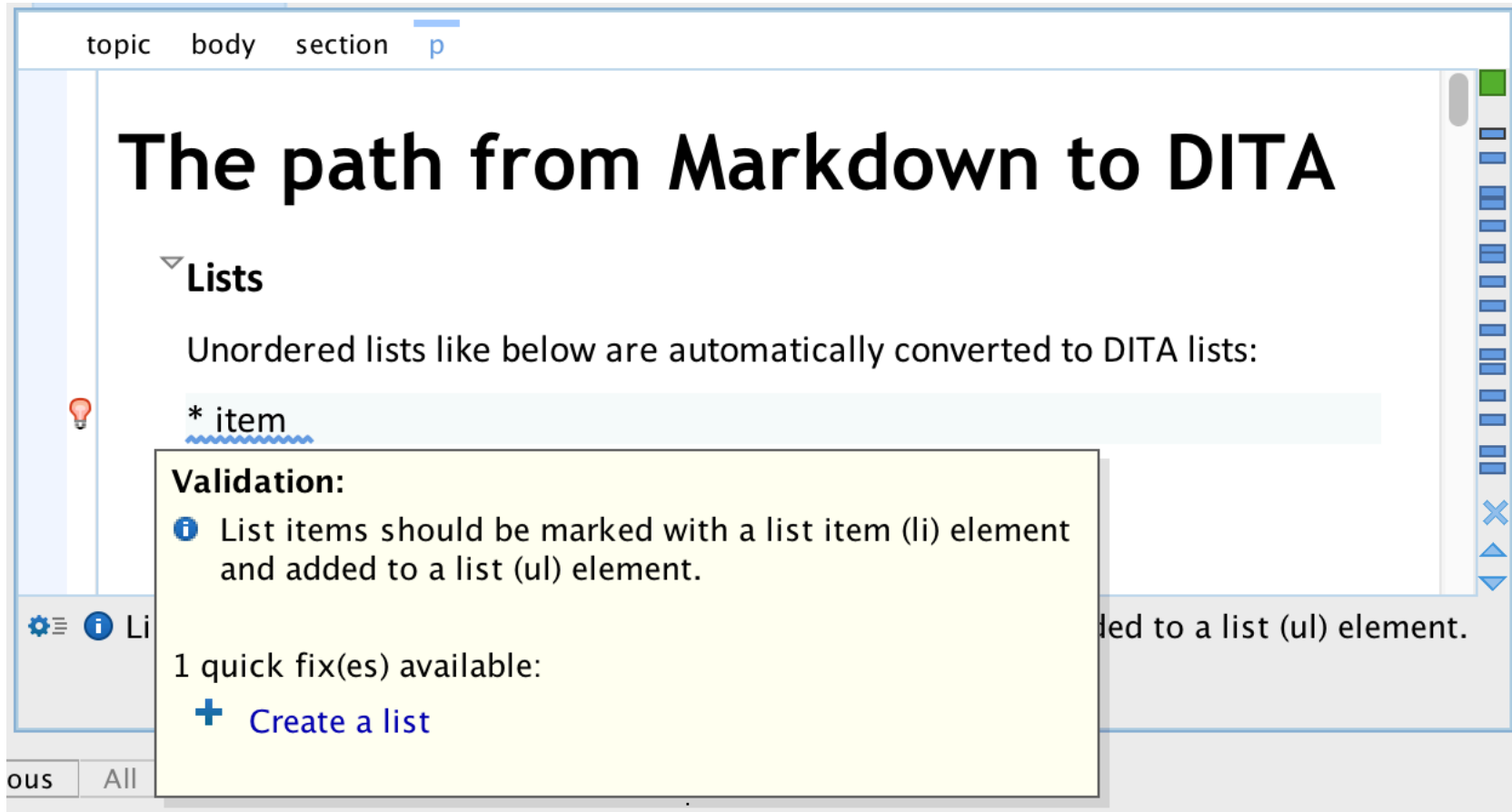
Unordered lists like below are automatically converted to DITA lists:

- * item
- * another item
- * one more

⚙️ ⓘ List items should be marked with a list item (li) element and added to a list (ul) element.

Text Grid Author

Markdown lists to DITA



The screenshot shows the Oxygen XML Editor interface. The main window displays a document with the title "The path from Markdown to DITA". Under the "Lists" section, there is a paragraph: "Unordered lists like below are automatically converted to DITA lists:". Below this, a list item is shown as "* item". A validation error is displayed over this list item, stating: "Validation: List items should be marked with a list item (li) element and added to a list (ul) element." Below the error message, it indicates "1 quick fix(es) available:" and provides a button labeled "+ Create a list". The editor's breadcrumb navigation at the top shows "topic > body > section > p".

topic body section p

The path from Markdown to DITA

▼ Lists

Unordered lists like below are automatically converted to DITA lists:

- * item

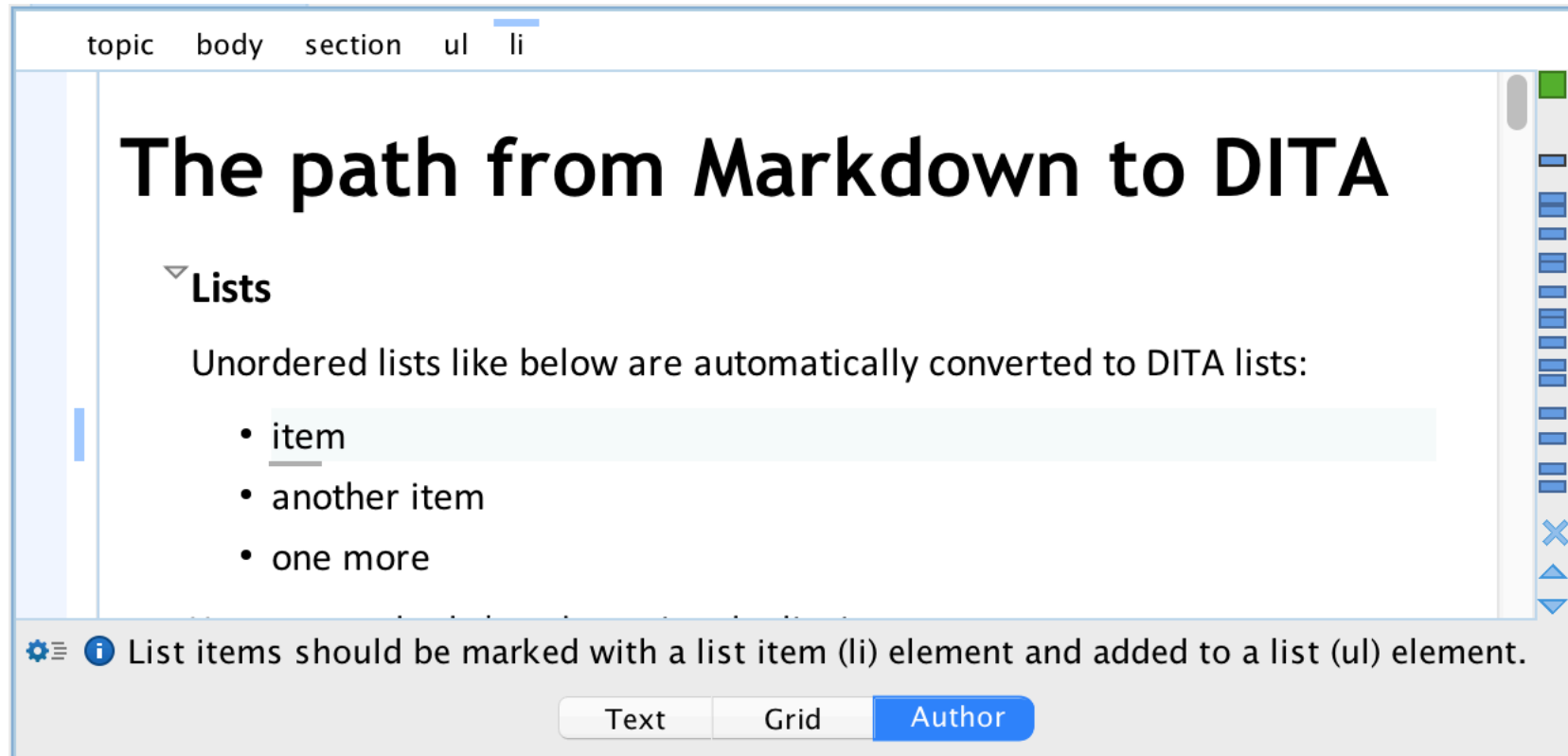
Validation:

- List items should be marked with a list item (li) element and added to a list (ul) element.

1 quick fix(es) available:

- + Create a list

Markdown lists to DITA

The screenshot shows the Oxygen XML Editor interface. At the top, there is a breadcrumb trail: "topic > body > section > ul > li". The main content area displays a DITA document with the following structure:

The path from Markdown to DITA

▼ **Lists**

Unordered lists like below are automatically converted to DITA lists:

- item
- another item
- one more

At the bottom of the editor, there is a warning message: "List items should be marked with a list item (li) element and added to a list (ul) element." Below the warning, there are three buttons: "Text", "Grid", and "Author".

Schematron Quick Fix

SQF

Layered on Schematron as annotation

SQF initiated by Nico Kutscherauer/data2type with contributions from Octavian Nadolu/oXygen XML Editor

www.w3.org/community/quickfix/

<https://github.com/schematron-quickfix/sqf>

<http://schematron-quickfix.github.io/sqf>

Example

```
<!-- Title - styling elements are not allowed in title. -->  
<sch:rule context="title/b">  
  <sch:report test="true()" sqf:fix="resolveBold" role="warn">  
    Bold element is not allowed in title.  
  </sch:report>
```

```
<!-- Quick fix that converts a bold element into text -->  
<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 select="node()"/>  
</sqf:fix>  
</sch:rule>
```

SQF

@id – an unique ID, used to refer the fix

title – represents the name of the quick fix

description – details about the fix

Operations:

- **add** – adds one or more nodes
- **delete** – deletes the matched nodes
- **replace** – replace a node by another one
- **stringReplace** – replace sub-strings of text nodes

You can use XSLT inside add and replace operations!

oXygen user guide

Many examples of Schematron rules + SQF:

<http://www.github.com/oxygenxml/userguide>

Transitioning from Schematron to an Integrated
Intelligent Style Guide

Rules

No image scaling ([image/@scale](#))

Avoid "oxygen" in index terms

No ";" at the end of list items

Index terms are allowed only in prolog

Topic IDs should match the topic filename

No link text equal with the referred URL

Images without a reference

Consecutive lists

Figures should have titles and should be placed within a paragraph

Definition lists should be inside paragraphs

Related links should be a linked list having a title

Long lines in code blocks (more than 90 characters)

Suggest specifying a language on some code blocks

Reports consecutive notes of the same type

Report empty elements

Report tables with more cells than the declared columns.

Sections should have IDs

Titles should be no more than 75 characters

A menu cascade should contain UI controls

Sections should have titles

No text directly inside a section

Thank you!

Questions?



george@oxygenxml.com



@georgebina



<http://www.oxygenxml.com>