XSpec Unit Testing for XSLT and Schematron

Alex Jitianu

alex_jitianu@oxygenxml.com

@AlexJitianu

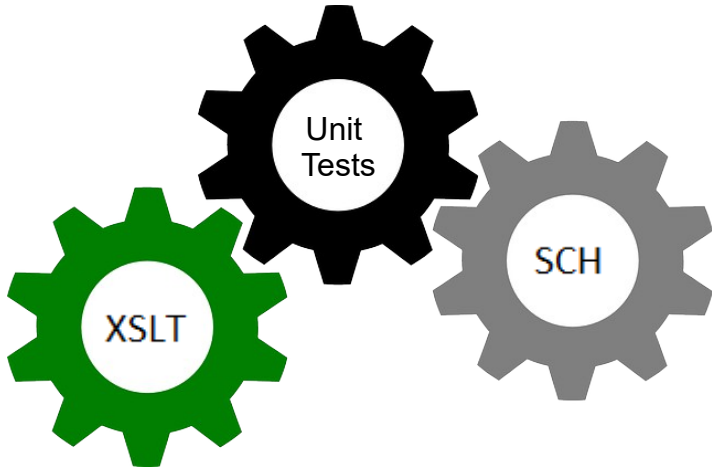
syncro soft

<oXygen/> XML Editor

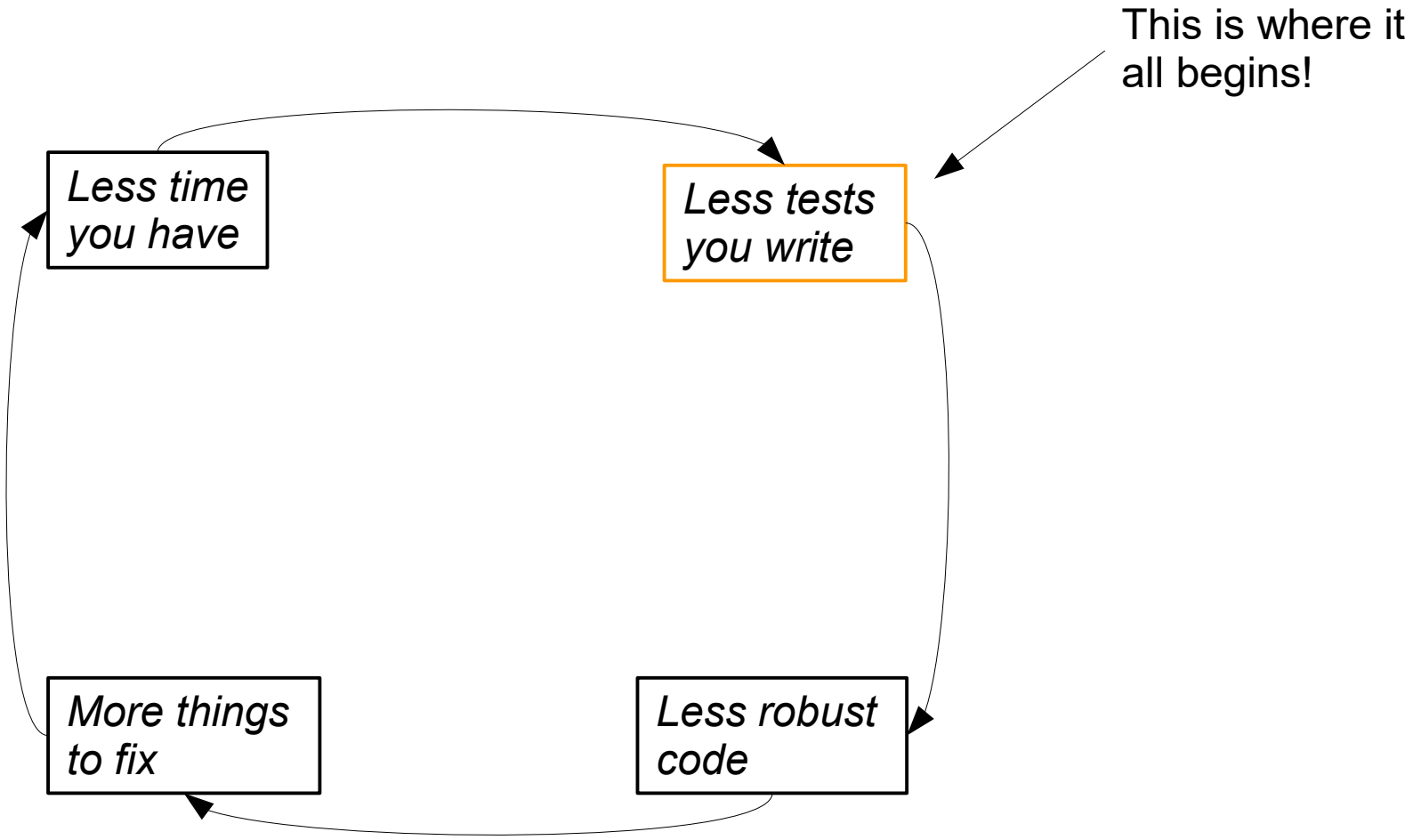
Agenda

- Unit testing and XSLT development
- XSpec framework
- Developing XSpec unit tests for XSLT in Oxygen
- Developing XSpec unit tests for Schematron

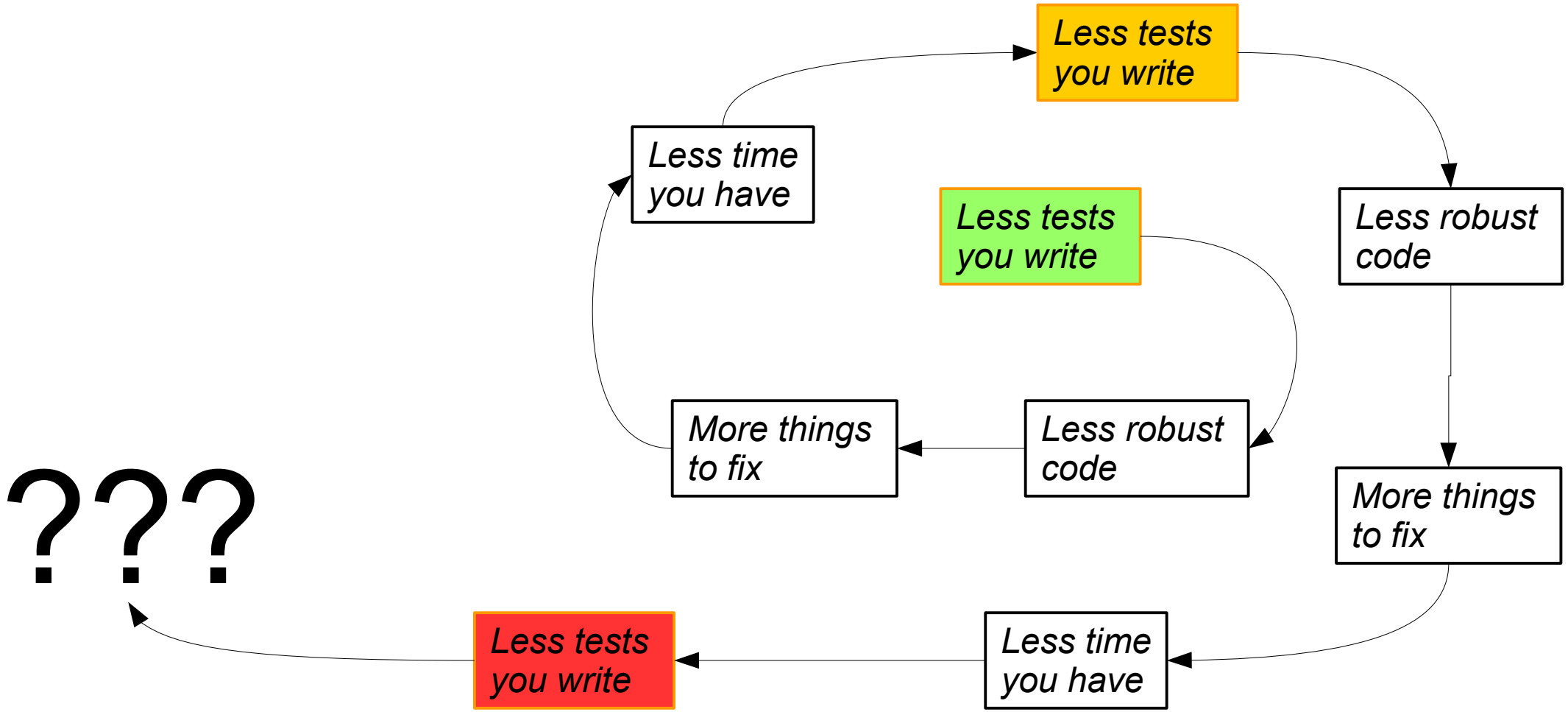
Why Unit Testing Matters



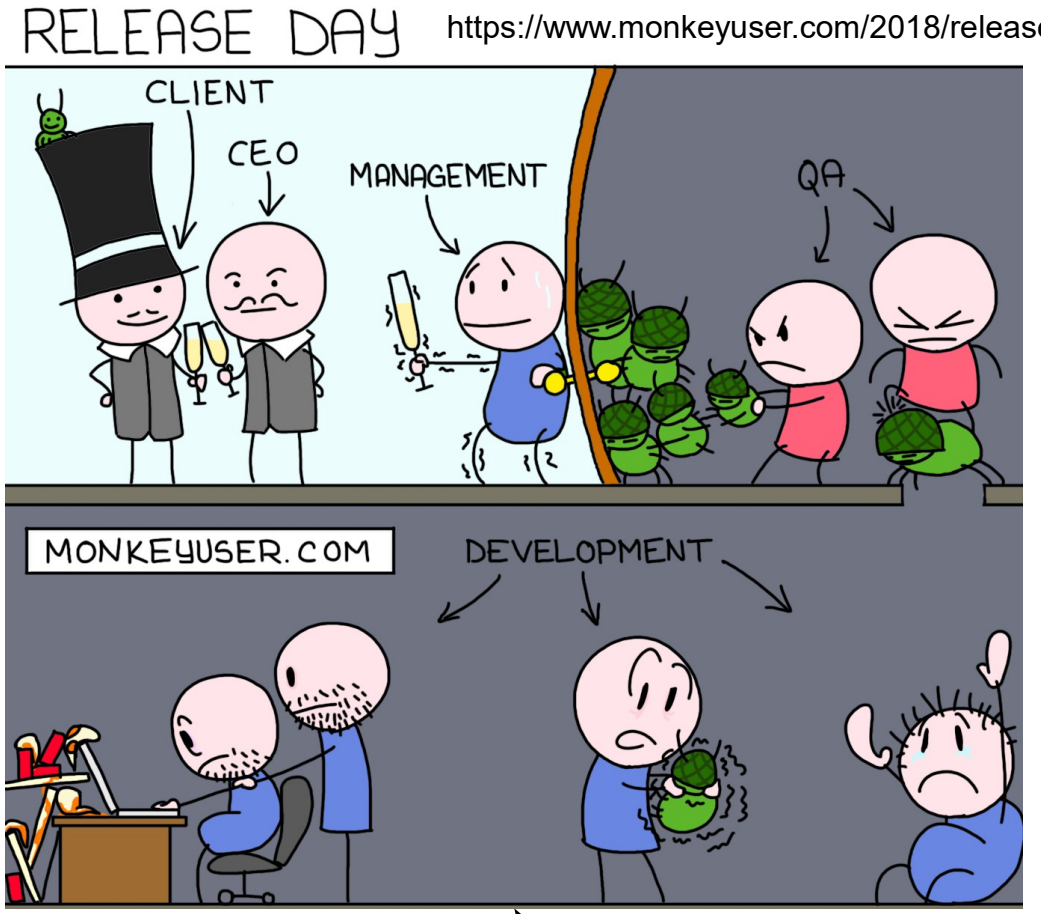
Why Unit Testing?



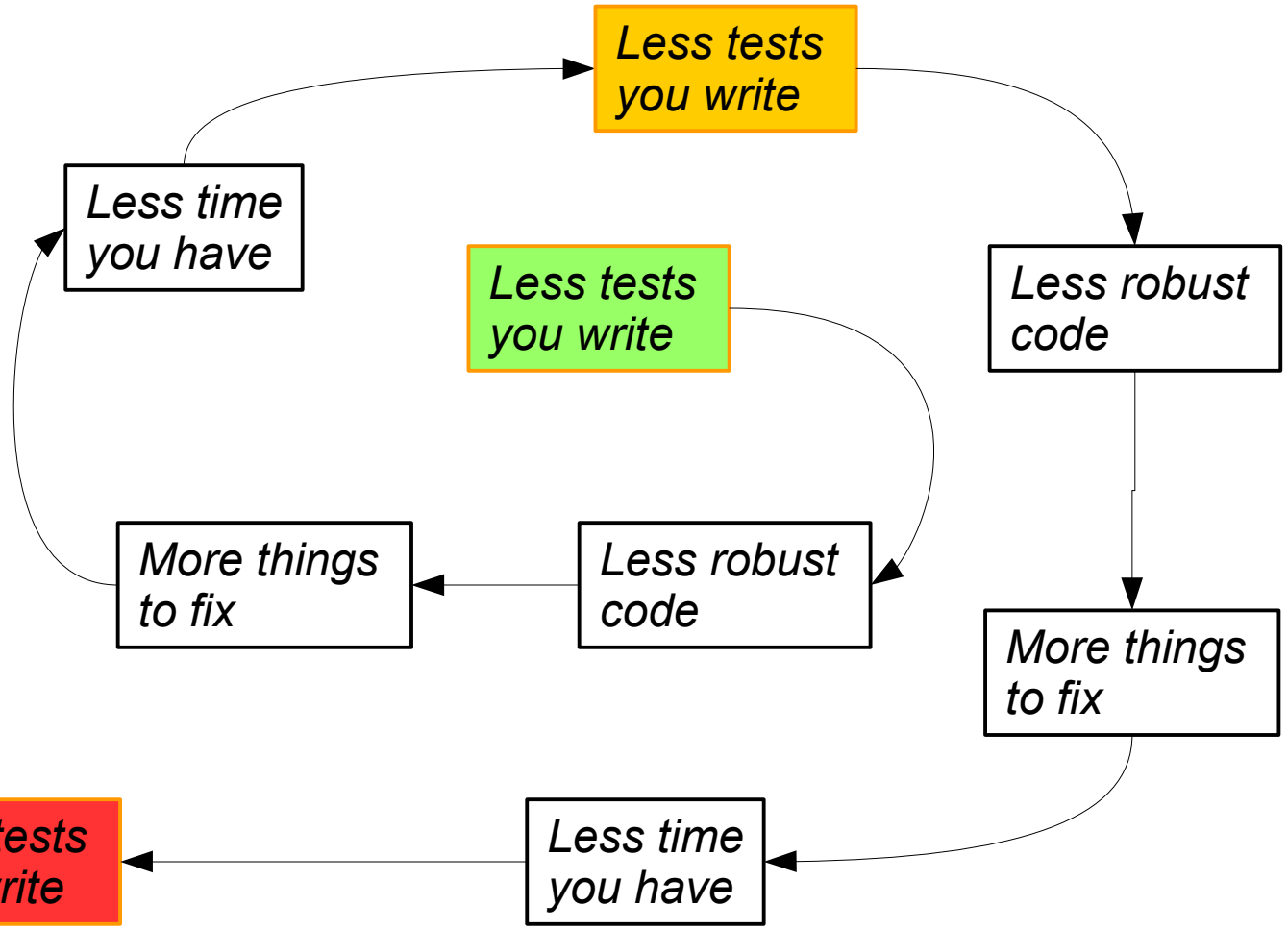
Why Unit Testing?



Why Unit Testing?



https://www.monkeyuser.com/2018/release-day/



Unit Tests

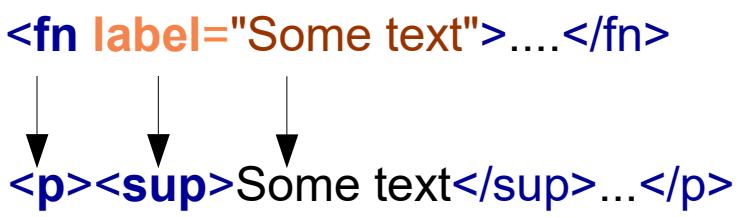
- More tests means less stress
- It verifies the behavior of a small part of the application
  - It is quite easy to implement
  - When it fails you know exactly where the fault is

Test Driven Development

- You describe a behavior (a.k.a. write a test)
- You test if the code has that behavior
- If it doesn't, you develop the code until it does (a.k.a. the test passes)
- On the long run you'll have regression tests
  - New functionality will not break the existing one
  - You can refactor without fear

XSLT Test Driven Development

- focus on the expected behavior
  - "*when processing a **shortdesc** element, it should create a **p** element*"
  - "*when processing a **fn** element with a **@label** attribute, it should create a **p** element with a **sup** child holding the value of the **@label** attribute*"

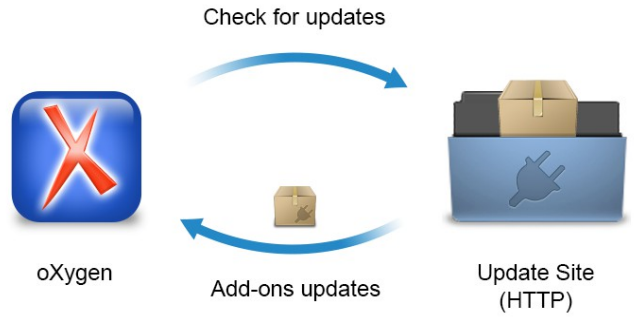


XSpec

- Unit test and BDD (Behaviour Driven Development) framework for XSLT and Schematron
  - https://github.com/xspec/xspec/
- Version 1.0.0 released
  - Schematron Unit Testing feature

XSpec framework integration

- bundled inside Oxygen
- an XSpec wizard
- support for working on XSpec scenarios
    - Validation
    - Content completion
- built-in transformation scenarios to execute XSpec scenarios
- XSpec helper plugin [Community update site]



XSLT TDD Demo

```
<book>
   <title>The Green Mile</title>
   <author>Stephen King</author>
   <price>18</price>
   <date>2013-10-30</date>
</book>
<book>
   <title>The Catcher in the Rye</title>
   <author>J. D. Salinger</author>
   <price>25</price>
   <date>2013-10-30</date>
</book>
```

XSLT TDD Demo

Behavior: *“Dates should be formated [D]/[M]/[Y]”*

Behavior: *“These … books have the total price of ...”*

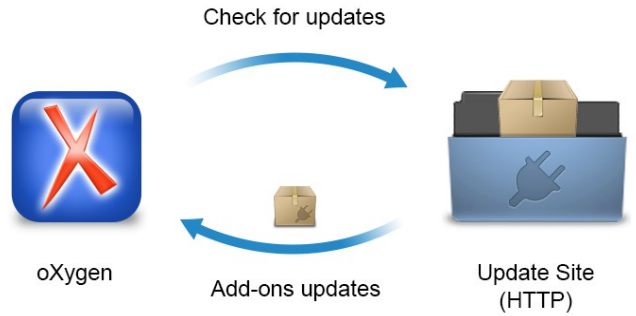
- Remember! Always start with a test!

XSpec framework scenario

```
<x:scenario label="Scenario for testing function total">
    <x:call function="f:total">
      <x:param name="books">
        <book>
          <title>The Green Mile</title>
          <author>Stephen King</author>
          <price>18</price>
        </book>
        <book>
          <title>The Catcher in the Rye</title>
          <author>J. D. Salinger</author>
          <price>25</price>
        </book>
      </x:param>
    </x:call>
    <x:expect label="The total price of the books">43</x:expect>
  </x:scenario>
```

XSpec framework integration

- bundled inside Oxygen
- an XSpec wizard
- support for working on XSpec scenarios
    - Validation
    - Content completion
- built-in transformation scenarios to execute XSpec scenarios
- XSpec helper plugin [Community update site]
    - Regression tests use case



Schematron

- A way to test XML documents
- A rule-based validation language
- Has business rules that DTD, XSD, RelaxNG can’t enforce
- You provide error messages that people can understand

Schematron

- Rule: *“The short description should be a single, concise paragraph containing one or two sentences of no more than 50 words.”*

```
<book>
  <title>Article title</title>
  <shortdesc>Short description.</shortdesc>
</book>
```

```
<sch:rule context="shortdesc">
  <sch:let name="words" value="count(tokenize(normalize-space(.), ' '))"/>
  <sch:assert test="$words &lt;= 50" role="warn" id=”shortdesc-too-short”>
    The short description should be a single.....
  </sch:assert>
</sch:rule>
```

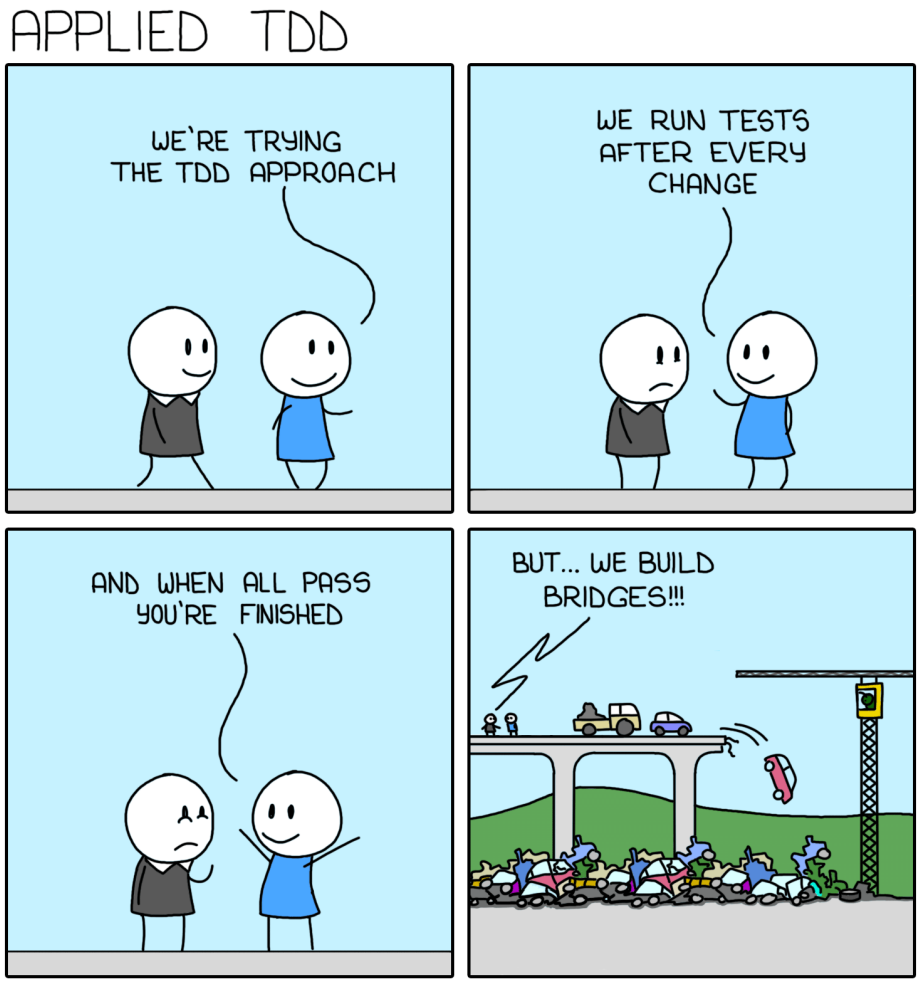
Schematron unit tests

- include XML that contains errors — which the Schematron should catch.
  - *<x:expect-assert>* verifies that <sch:assert> is thrown.
- include XML that is valid — which the Schematron should pass,
  - *<x:expect-not-assert>* verifies that <sch:assert> is **not** thrown.

```
<x:scenario label="Proper descriptionxxxx">
   <x:context>
      <books>
         <book>
            <shortdesc>A short description.</shortdesc>
         </book>
         <book>
            <shortdesc>A proper description with just enough words.</shortdesc>
         </book>
      </books>
   </x:context>

   <x:expect-assert     id="shortdesc-too-short" location="/books/book[1]/shortdesc[1]"/>
   <x:expect-not-assert id="shortdesc-too-short" location="/books/book[2]/shortdesc[1]"/>
</x:scenario>
```

TDD is Great!


APPLIED TDD
WE'RE TRYING THE TDD APPROACH
WE RUN TESTS AFTER EVERY CHANGE
AND WHEN ALL PASS YOU'RE FINISHED
BUT... WE BUILD BRIDGES!!!
https://www.monkeyuser.com/2018/applied-tdd/
MONKEYUSER.COM

THANK YOU!

Any questions?

Alex Jitianu

alex_jitianu@oxygenxml.com

@AlexJitianu