Testing XSLT with XSpec

Jeni Tennison

Overview

- Background
- Syntax
- Implementation
- Experience
- Future

Background

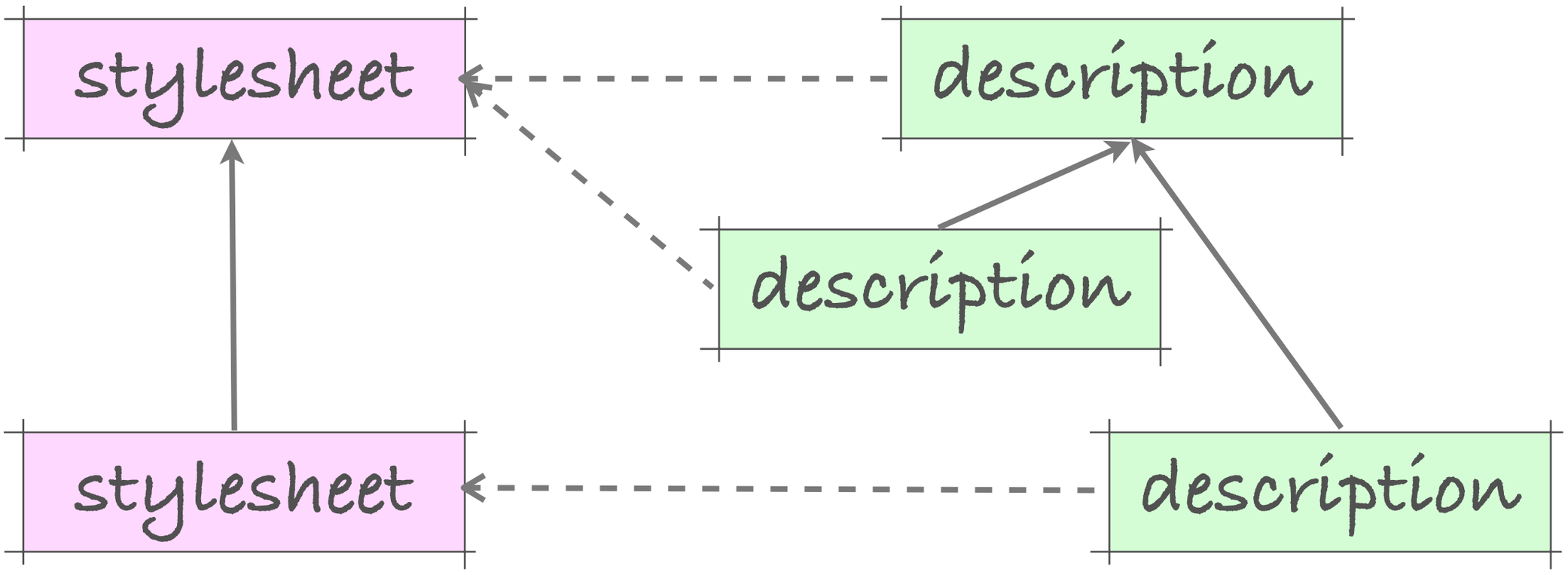
- Tests are important
  - especially for XSLT
- Agile methodologies
  - test-driven development
- "Tennison tests"
  - XSLT-based unit testing

Behaviour-Driven Dev.

- RSpec for Ruby (on Rails)
- Flexible level of testing
- Emphasis on description
  - tests as documentation
- Structured tests
  - reuse of 'setup' code

XSpec Documents

- Separate XSpec documents
  - associated with a stylesheet
  - modularised / importable



XSpec Scenarios

- Human-readable label
  - "when ..."
- XSLT hook
  - context node/mode/params
  - call to function/template
  - inherited from parent scenarios

Example Scenario

```
<x:scenario label="an empty paragraph">
   <x:context mode="preprocess">
      <w:p>
         <w:pPr>
            <w:ind w:left="0" w:firstLine="0" />
            <w:rPr>
               <w:b />
            </w:rPr>
         </w:pPr>
      </w:p>
   </x:context>
   ...
</x:scenario>
```

Example Nesting

```
<x:scenario label="when creating Amending elements">
  <x:context mode="LegAmending" />

  <x:scenario label="for an SI">
    <x:context>
      <tc>S. I. No. 189 of 1993, reg.  10 (2) </tc>
    </x:context>
    <x:expect>...</x:expect>

    <x:scenario label="with no spaces between the S. and the I.">
      <x:context>
        <tc>S.I. No. 52 of 2006, art. 8; art. 2(4)</tc>
      </x:context>
      ...
    </x:scenario>
  </x:scenario>
</x:scenario>
```

XSpec Expectations

- Human-readable label
  - "it should ..."
- XPath test
- Sample output
  - against a portion of the result
  - use "..." to elide content

Example Expectations

```
<x:expect label="should produce an Amendment element">
  <Amendment Made="no">
    <Amended>...</Amended>
    <Amending>...</Amending>
  </Amendment>
</x:expect>

<x:expect label="should produce an Amended element whose value is
that of the third cell"
  test="/dir:Amendment/dir:Amended =
        'Regulations stated to be without prejudice to application
of the Act'" />

<x:expect label="should produce an Amending element whose value is
that of the fourth cell"
  test="/dir:Amendment/dir:Amending =
        'S.I. No. 551 of 2006, reg. 22(6)'" />
```

Shared Tests

- Expectations can be shared
  - reference with "like"

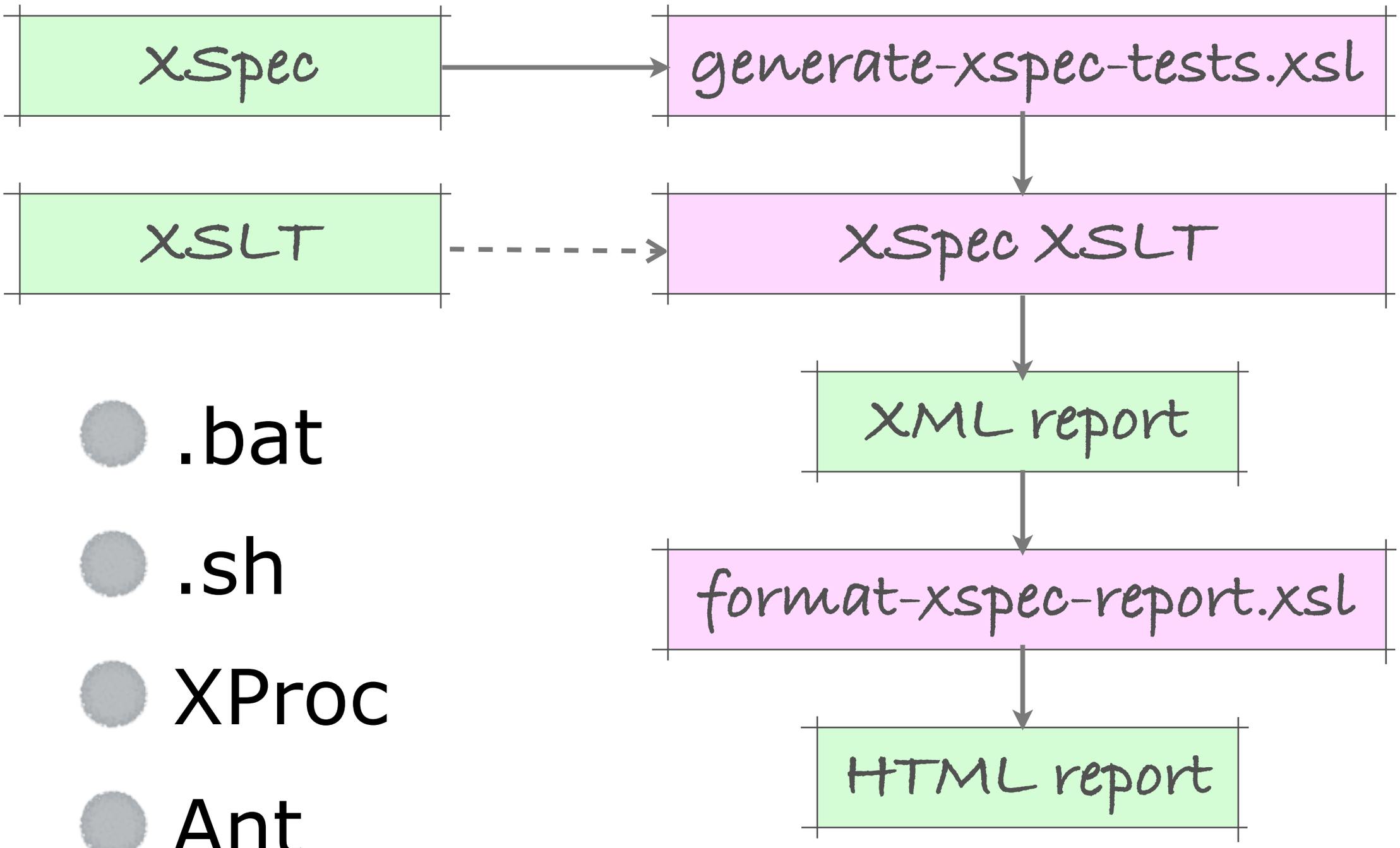
```
<x:scenario label="a consistent document" shared="yes">
  ...
</x:scenario>

<x:scenario label="filtering the current version">
  <x:context mode="currentVersion" />
  <x:like label="a consistent document" />
</x:scenario>

<x:scenario label="filtering the prospective version">
  <x:context mode="prospectiveVersion" />
  <x:like label="a consistent document" />
</x:scenario>
```

Pending and Focus

- Pending scenarios
  - not yet implemented
- Focus scenarios
  - code you're working on

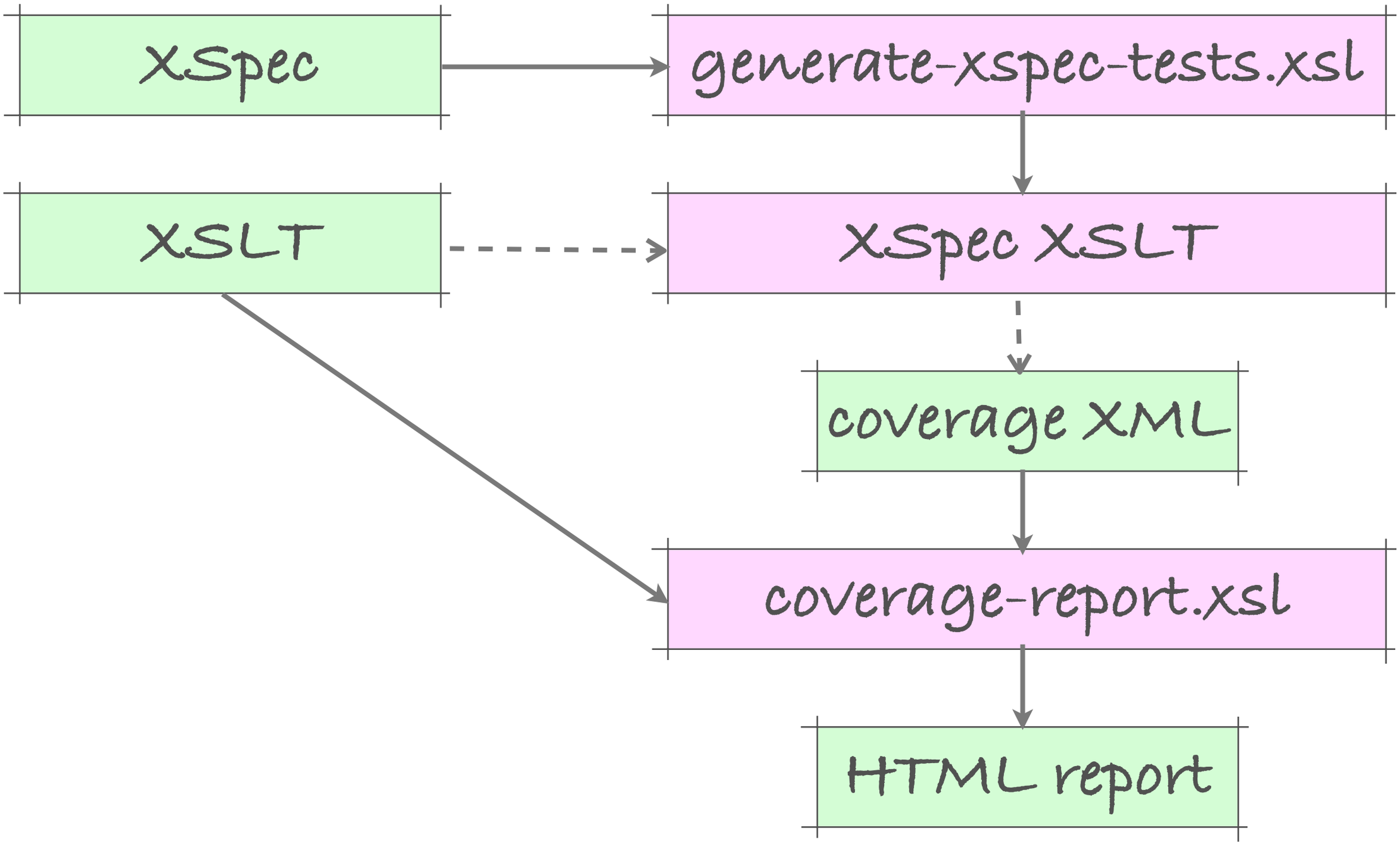
```
<x:scenario label="with no spaces between the S. and the I."
  focus="recognising the SI reference">
  ...
</x:scenario>
```

XSLT Implementation



- .bat
- .sh
- XProc
- Ant

Coverage Testing



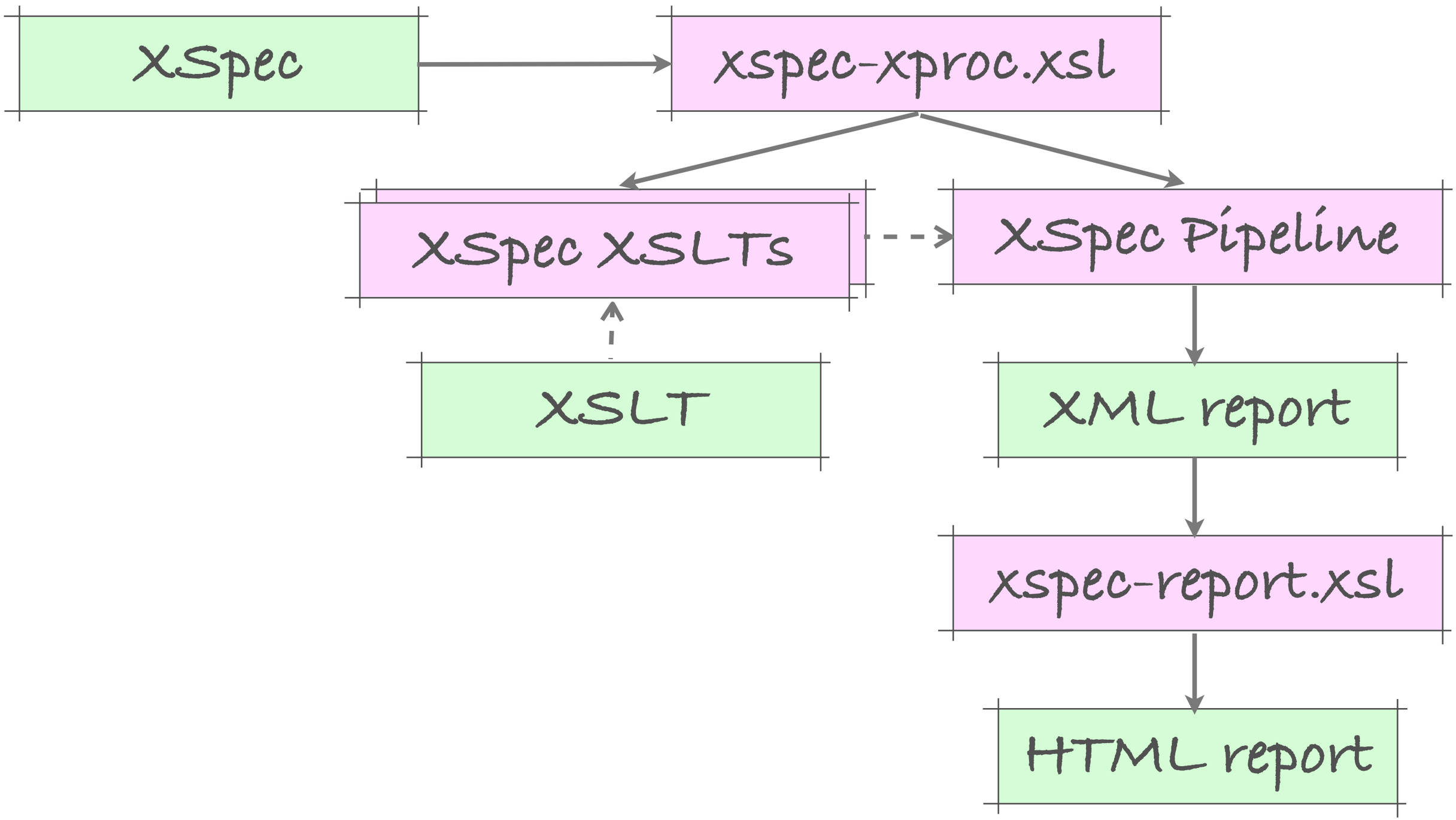
Experience

- Multiple projects
- Initial descriptions hard
- Ongoing changes
  - easy when adding new features
  - hard if tests haven't been focused
- Improves confidence

Limitations

- XSLT implementation
- Cannot
  - test with different global params
  - test generation of messages
  - test generation of multiple results
- Not supported in language

XProc Implementation?



Additional Features

Testing messages

```
<x:expect output="message">...</x:expect>
```

```
<x:expect output="error">...</x:expect>
```

Testing results

```
<x:expect output="#result">...</x:expect>
```

```
<x:expect output="filename.html">...</x:expect>
```

Summary

- Testing improves your code
- BDD good fit with XSLT
- Implementation works
  - but could be improved
  - XProc-based implementation

`http://code.google.com/p/xspec`

Questions?