

Testing XSLT

Tony Graham
Menteith Consulting Ltd
13 Kelly's Bay Beach
Skerries, Co Dublin
Ireland
info@MenteithConsulting.com
<http://www.menteithconsulting.com>

Version 1.0 – 5 December 2007
© 2007 Menteith Consulting Ltd

Menteith
C O N S U L T I N G

Testing XSLT

1

- Introductions
- XSLT profilers
- XSLT debuggers
- XSLT unit testing frameworks
- XSLV static validation tool
- FO testing frameworks
- Conclusion

Introductions

Who am I?

2

- Tony Graham of Menteith Consulting
- XML and XSL/XSLT consultant
- Based in Dublin, Ireland
- Doing XSLT since 1998
- Working with both XSLT 1.0 and XSLT 2.0

Who are You?

3

- What is your XSLT experience?
- What is your testing experience?
- What are you looking for?

Overview of XSLT Testing Tools

4

- Profilers
- Debuggers
- Unit testing frameworks
- XSLV Static Validator
- FO testing frameworks

Profilers

5

- Track activity of a running transformation
- Find "hotspots" where processor spending most time
- Typical scenario:
 - Profile
 - Review
 - Modify
 - Repeat

6 Debuggers

- Step through execution
- XML IDEs
- Emacs

7 Unit Testing Frameworks

- Run all or part of a transformation
- Provide "known" input
- Make assertions about expected result
- Black box testing: test complete transform *without* looking knowledge of implementation
- White box testing: test with knowledge of implementation

8 XSLV Static Validation Tool

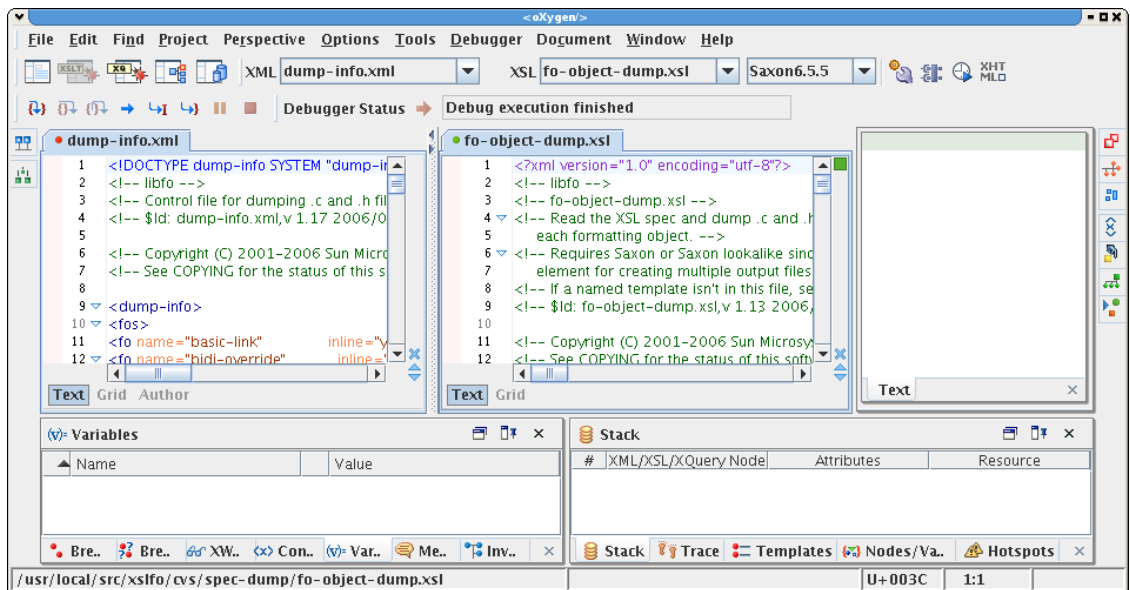
- Test that transform can convert from one schema to another
- Analyses stylesheet – does not run the transformation

9 FO Testing Frameworks

- Run XSL formatter on "known" input
- Check area tree or output matches expected result

10 Debuggers

- Step through execution of transformation



Profilers

Profiling XSLT is Inexact

11

- Template execution time depends on execution time of all templates it calls
- Depends on machine state
 - Faster when processor already in memory
- Depends on current node list
- Processor may be optimizing or doing lazy evaluation

XSLT Profiling in XML IDEs

12

- <oXygen/>, Stylus Studio, XMLSpy, etc.
- One of many features of an IDE

The screenshot shows the oXygen XML IDE interface. The main editor displays XSLT code for a search result page. Below the editor, two panels provide profiling data:

Invocation tree (Total time 2887 ms)

- 100.0% - 2887 ms - 0.01% - 0 ms 1 inv. template (match="/")
- 99.99% - 2887 ms - 0.02% - 0 ms 1 inv. apply-templates
 - 99.97% - 2886 ms - 0.05% - 1 ms 1 inv. apply-templates
 - 0.01% - 0 ms - 0.01% - 0 ms 1 inv. template (match="*")
 - 0.01% - 0 ms - 0.01% - 0 ms 1 inv. template (match="..w")
 - 0.01% - 0 ms - 0.01% - 0 ms 1 inv. template (match="..w")
 - 0.01% - 0 ms - 0.01% - 0 ms 1 inv. template (match="..w")
 - 0.01% - 0 ms - 0.01% - 0 ms 1 inv. template (match="..w")

Hotspots

Instruction	Time	Hits
210 Hotspots		
template (match="lir:e") (mode="compar	87 ms (3.03%)	24
hr	67 ms (2.33%)	8
template (name="encodeValue")	62 ms (2.15%)	70
for-each	54 ms (1.89%)	9
li	50 ms (1.76%)	24
value-of	48 ms (1.69%)	70
template (name="buildUrlNewFirstAnswer	46 ms (1.61%)	4
value-of	45 ms (1.57%)	48

13 xsltproc

- profile dumps profiling information
- repeat runs the transformation 20 times

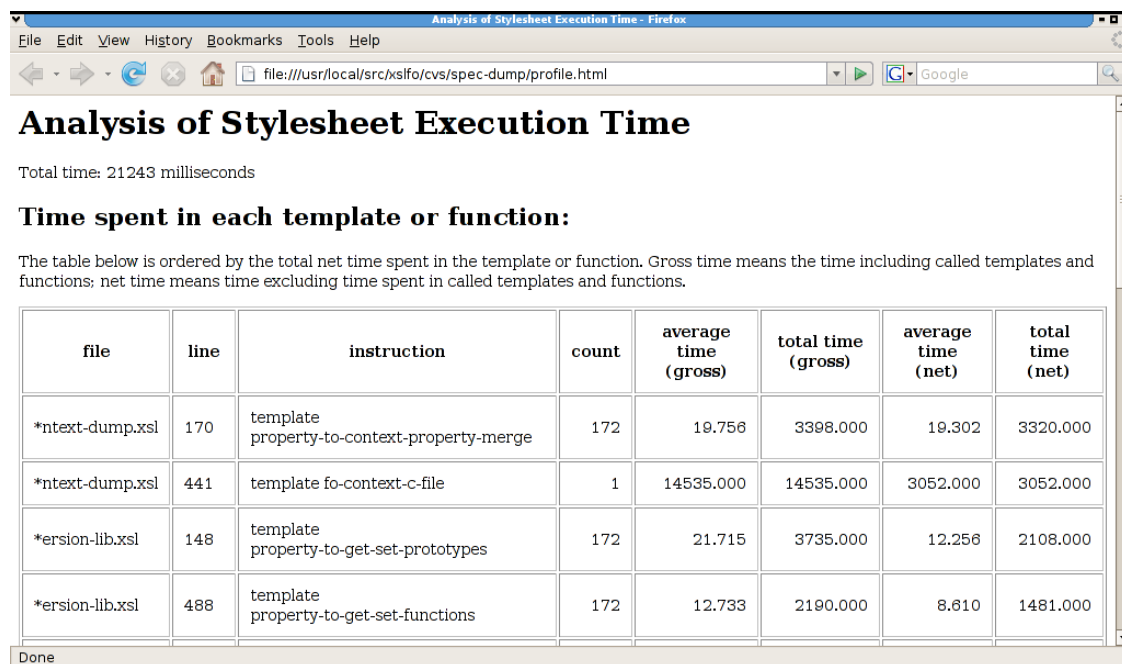
```

number  name                                     Calls Tot 100us Avg
   0    property-to-context-property-merge  172 426224 2478
   1    fo-context-c-file                    1 315848 315848
   2    property-to-slist-foreach-if         172 143712 835
   3    property-to-context-property-copy    172 107006 622

```

14 Saxon

- Run with -TJ switch and save error output
- Process timing info with timing-profile.xml



Analysis of Stylesheet Execution Time - Firefox

file:///usr/local/src/xsifo/cvs/spec-dump/profile.html

Analysis of Stylesheet Execution Time

Total time: 21243 milliseconds

Time spent in each template or function:

The table below is ordered by the total net time spent in the template or function. Gross time means the time including called templates and functions; net time means time excluding time spent in called templates and functions.

file	line	instruction	count	average time (gross)	total time (gross)	average time (net)	total time (net)
*ntext-dump.xml	170	template property-to-context-property-merge	172	19.756	3398.000	19.302	3320.000
*ntext-dump.xml	441	template fo-context-c-file	1	14535.000	14535.000	3052.000	3052.000
*ersion-lib.xml	148	template property-to-get-set-prototypes	172	21.715	3735.000	12.256	2108.000
*ersion-lib.xml	488	template property-to-get-set-functions	172	12.733	2190.000	8.610	1481.000

Done

15 Profiling with Java (or C) Profiler

- Useful if understand internals of XSLT processor
- Last resort when stylesheet is a single template
- `java -Xrunhprof:cpu=samples`
 - Results in `java.prof.txt`
- Used by Michael Kay with Saxon
 - <http://www.biglist.com/lists/xsl-list/archives/200710/msg00192.html>

Why is my <hr/> a Hotspot?

16

- When profiler pauses processor to sample current state
- Sampling too seldom gives random results
- Profiler overhead skews result when sampling too frequently
- Solution: Run the transform multiple times

Instruction	Time	Hits
210 Hotspots		
template (match="iir:e") (mode="compar	87 ms (3.03%)	24
hr	67 ms (2.33%)	8
template (name="encodeValue")	62 ms (2.15%)	70
for-each	54 ms (1.89%)	9
li	50 ms (1.76%)	24
value-of	48 ms (1.69%)	70
template (name="buildUrNewFirstAnswer	46 ms (1.61%)	4
value-of	45 ms (1.57%)	48

Unit Testing XSLT

17

- Run all or part of a transformation
- Provide "known" input
- Make assertions about expected result
- Black box testing: test complete transform *without* looking knowledge of implementation
- White box testing: test with knowledge of implementation
- Catching `xsl:message`
- Errors not caught by unit tests

When To Use Unit Testing?

18

- Good for XML and HTML output
 - Easy to make assertions about structure of output
- Harder to verify text output
 - Messier to make assertions
 - Testing individual templates easier than testing complete output
- Still need to validate output
 - Spec may be incorrect
 - Tests may have bugs

19 How Effective is Unit Testing?

Removal Step	Lowest Rate	Modal Rate	Highest Rate
Informal design reviews	25%	35%	40%
Formal design inspections	45%	55%	65%
Informal code reviews	20%	25%	35%
Formal code inspections	45%	60%	70%
Modelling or prototyping	35%	65%	80%
Personal desk checking of code	20%	40%	60%
Unit test	15%	30%	50%
New function (component) test	20%	30%	35%
Integration test	25%	35%	40%
Regression test	15%	25%	30%
System test	25%	40%	55%
Low-volume beta test (<10 sites)	25%	35%	40%
High-volume beta test (>1,000 sites)	60%	75%	85%

Source: *Software Estimation: Demystifying the Black Art*, Steve McConnell

20 Unit Testing Frameworks for XSLT

- XSLTUnit
- Juxy
- Unit Testing XSLT
- tennison-tests
- <XmlUnit/>
- utf-x
- XTS

21 My Unit Testing Wish-List

- Assert a message was (or was not) emitted
- Assert a secondary result document was (or was not) emitted
- Works with keys
- Test stylesheet as a whole
- Summaries of multiple unit test files
- Maintainers respond to bug reports
- Test data and assertions in XML

XSLTunit

22

- Eric van der Vlist, Dyomedeia
- <http://xsltunit.org>
- “Stable, rustic, and mature”
- XSLT 1.0
- Input: XSLT stylesheet
- Output: XML indicating success or failure

XMLUnit Test

23

```
<xsl:template match="/">
  <xsltu:tests>
    <xsltu:test id="test-title">
      <xsl:call-template name="xsltu:assertEqual">
        <xsl:with-param name="id" select="'full-value'"/>
        <xsl:with-param name="nodes1">
          <xsl:apply-templates
            select="document('library.xml')/library/
              book[isbn='0836217462']/title"/>
        </xsl:with-param>
        <xsl:with-param name="nodes2">
          <h1>Being a Dog Is a Full-Time Job</h1>
        </xsl:with-param>
      </xsl:call-template>
    </xsltu:test>
    ...
  </xsltu:tests>
```

Juxy

24

- Pavel Sher, Tigris
- <http://juxy.tigris.org/>
- Library for unit testing XSLT stylesheets from Java
 - Standalone or integrates with Ant and JUnit
- Input: Java class
 - Test as String, DOM, or File
- Output: Depends on JUnit test runner used

25 Juxy Test

```

public class SampleTestCase extends JuxyTestCase
{
    public void testListTransformation() {
        newContext("stylesheet.xsl");
        context().setDocument("" +
            "<list>" +
            " <item>item 1</item>" +
            " <item>item 2</item>" +
            " <item>item 3</item>" +
            "</list>");
        Node result = applyTemplates();
        xpathAssert("text()", "item 1, item 2, item 3").eval(result);
    }
}

```

26 Unit Testing XSLT

- Jeni Tennison, Jeni Tennison Consulting
- <http://www.jenitennison.com/xslt/utilities/unit-testing>
- XSLT 2.0
- Input: Inside stylesheet or separate file
- Output: XML or HTML report

27 Unit Testing XSLT Test

```

<test:tests>
<test:test>
    <test:param name="number" select="2" />
    <test:expect select="4" />
</test:test>
</test:tests>
<xsl:function name="eg:square" as="xs:double">
<xsl:param name="number" as="xs:double" />
<xsl:sequence select="$number * $number" />
</xsl:function>

```

Unit Testing XSLT Report

28

TEST REPORT

Stylesheet: `usr/local/src/unit-testing/square.xsl`

Tested: 4 April 2007 at 23:48

Summary

Passed 1/2

Test ID	Title	Success/Total
square		1/2

Square

Test ID	Test Title	Success/Failure
square.1		Success
square.2		Failure

Tested XSLT

```
<xsl:function xmlns:xsl="http://www.w3.org/XML/1998/namespace"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:test="http://www.jenitennison.com/xslt/unit-test"
  xmlns:xsd="http://www.w3.org/2001/XMLSchemaAlias"
  xmlns:eg="http://example.com/"
  name="eg:square"
  as="xs:double">
  <xsl:param name="number"
    as="xs:double" />
  <xsl:sequence select="$number * $number" />
</xsl:function>
```

Square.2

Parameters

Name	Value
number	2

Results

Expected Result	Actual Result
5	4

tennison-tests

29

- Toby Weston
- <http://tennison-tests.sourceforge.net/>
- Unit Testing XSLT meets Ant
- Input and output as for Unit Testing XSLT
- Run multiple tests from Ant
- No summary of running multiple tests
- Needs namespace fixup in `generate-tests-utils.xsl`

tennison-tests in Ant Build File

30

```
<!-- Defines the 'tennison-tests' custom Ant task. -->
<taskdef name="xslttest"
  classname="com.jenitennison.xslt.unittest.XSLTTest"
  classpath="${basedir}/ant-xslttest-1.0.0.jar"/>
<!-- Executes all unit tests in 'test'. -->
<target name="test" depends="init">
<xslttest src="${basedir}/ant-xslttest-1.0.0/main/src/xslt"
  target="build/test"
  generate="true">
  <fileset dir="${basedir}/test">
<include name="*.xml" />
</fileset>
  <factory name="net.sf.saxon.TransformerFactoryImpl" />
</xslttest>
</target>
```

31 <XmlUnit/>

- Tim Bacon and Jeff Martin
- <http://xmlunit.sourceforge.net/>
- Java or .Net
- Input: Java/C# class
 - Test is String, DOM, or File
- Output: Depends on JUnit/NUnit test runner
 - E.g., XML to make HTML report when used with Ant

32 <XmlUnit/> Test

```
public void testXSLTransformation() throws Exception {
    String myInputXML = "...";
    File myStylesheetFile = new File("...");
    Transform myTransform = new Transform(myInputXML,
                                         myStylesheetFile);

    String myExpectedOutputXML = "...";
    Diff myDiff = new Diff(myExpectedOutputXML, myTransform);
    assertTrue("XSL transformation worked as expected " +
              myDiff, myDiff.similar());
}
```

33 Unit Testing Framework – XSLT

- Alex, Mikael, Richard, Yacek
- <http://utf-x.sourceforge.net/>
- Java
- Input: XML
- Output: Depends on JUnit test runner used

34 utf-x Test

```
<utfx:test>
  <utfx:name>stylesheet parameter test</utfx:name>
  <utfx:stylesheet-params>
    <utfx:with-param name="stylesheet-param1"
                    select="'UTF-X' " />
  </utfx:stylesheet-params>
  <utfx:assert-equal>
    <utfx:source>
      <print-param1 />
    </utfx:source>
    <utfx:expected>
      <stylesheet-param1>UTF-X</stylesheet-param1>
    </utfx:expected>
  </utfx:assert-equal>
</utfx:test>
```

XTS

35

- Florent Georges
- <http://www.fgeorges.org/xslt/xslt-unit/>
- XSLT 2.0
- XSLT & XQuery Unit Testing
- Input: XML file
 - Transformed into stylesheet that imports stylesheet being tested
- Output: XML and HTML report

xslt-unit Test

36

```

<t:tests>
<t:title>hello-world()</t:title>
<t:test>
  <t:expect select="'Hello, world!'" />
  <xsl:sequence select="hw:hello-world()" />
</t:test>
</t:tests>

```

"Black Box" Testing

37

- For this input... expect this output
- Need complete documents for input
- Make assertions about complete documents at output
- Unit test for `match="/"` template
- Could also use Schematron

"Black Box" Advantages

38

- Don't need to understand stylesheet
- Reusable even if stylesheet refactored

"Black Box" Disadvantages

39

- Need complete documents as input
- If input changes, need to revise *all* tests

Exercise 1

40

What are some useful tests for the specification in Exhibit 1?

41 "White Box" Testing

- For this template, with this context... expect this output
- Complete documents or fragments as input
 - XPath to select part of a document, or
 - Fragment provided as part of test
- Make assertions about complete result from fragment

42 "White Box" Advantages

- Input and output can be simpler
- Maybe revise fewer tests if input changes

43 "White Box" Disadvantages

- Result may be invalid even if all tests pass
- Best suited to testing named templates
- Fragile if:
 - Stylesheet refactored
 - Selecting templates by context and `match` attribute changes
- Keys may not work if input is fragment contained in test definition

44 Exercise 2

What are some useful tests for the specification in Exhibit 6?

45 "Clean" vs "Dirty" Tests

Clean test:

- Good input
- Test for correct output

Dirty test:

- Incorrect input
- Test for correct handling of error condition
 - Message
 - No output
 - Output based on input

46 `xs1:message` and Unit Tests

- Pure XSLT frameworks can't catch `xs1:message` output
- Can't differentiate between template giving error message and template accidentally giving no result
- Unit tests abort if `terminate="yes"`

Errors That Aren't Caught By Unit Tests

Exercise 3

47

What errors in the stylesheet in Exhibit 6 would not be caught by unit testing?

Error Not Caught By Unit Tests

48

- Unnecessary //
- Empty #PCDATA
- Out of date or inappropriate comments
- Template that is never matched
- Typos in `xsl:strip-space` and `xsl:preserve-space`
- Error in `xsl:when` covered by `xsl:choose`

XSLV Static Validation Tool

49

- University of Aarhus, Denmark
- <http://www.brics.dk/XSLV/>
- Determine if output valid to output schema if input valid to input schema DTD, XML Schema, or Relax NG subset
- Makes approximations
 - Some false negatives
 - Never a false positive
- Limited use while developing stylesheet
- Some vocabularies don't have a schema, e.g., XSL FO

XSLT Coverage Tool

50

- Is there one?

FO Testing Frameworks

51

- Run XSL formatter on "known" input
- Check area tree or output matches expected result

52

xmlroff Testing Module

- Runs *any* command-line XSL formatter
- Rasterise PDF or PostScript output
- Compare against a reference
- Summary and individual reports

testsuite/expression/expression1
 Report created: Mon Apr 9 21:39:48 IST 2007

Test	FO multiple	XML inherited-property-value.fo	XSL	Result
	PDF expression.pdf	Log expression.log	Pages 1	None

Agreement:

Future support:

Spec problem:

Test problem:

Further info:

Note that "Update" updates the XML source for this report and **does not** regenerate this HTML page.

Page 1 expression1.00.png	Reference	Stereo
<p>inherited-property-value() The value of font-size property specified on ancestor page-sequence is 10pt. 1. No font-size specified. Should be 10pt. 2. font-size="20pt". Should be 20pt. 3. font-size="inherited-property-value()". Should be 10pt. 4. font-size="inherited-property-value(font-size)". Should be 10pt. 5. font-size="inherited-property-value('font-size')". Should be 10pt. 6. font-size="inherited-property-value(provisional-distance-between-starts) div 2". Should be 12pt.</p>	<p>inherited-property-value() The value of font-size property specified on ancestor page-sequence is 10pt. 1. No font-size specified. Should be 10pt. 2. font-size="20pt". Should be 20pt. 3. font-size="inherited-property-value()". Should be 10pt. 4. font-size="inherited-property-value(font-size)". Should be 10pt. 5. font-size="inherited-property-value('font-size')". Should be 10pt. 6. font-size="inherited-property-value(provisional-distance-between-starts) div 2". Should be 12pt.</p>	<p>inherited-property-value() The value of font-size property specified on ancestor page-sequence is 10pt. 1. No font-size specified. Should be 10pt. 2. font-size="20pt". Should be 20pt. 3. font-size="inherited-property-value()". Should be 10pt. 4. font-size="inherited-property-value(font-size)". Should be 10pt. 5. font-size="inherited-property-value('font-size')". Should be 10pt. 6. font-size="inherited-property-value(provisional-distance-between-starts) div 2". Should be 12pt.</p>

53

“Stereo” Comparison

- Visually compare result and reference
- Red channel from result
- Blue channel from reference

inherited-property-value()

The value of font-size property specified on ancestor page-sequence is 10pt.

1. No font-size specified. Should be 10pt.

2. font-size="20pt". Should be 20pt.

3. font-size="inherited-property-value()". Should be 10pt.

4. font-size="inherited-property-value(font-size)". Should be 10pt.

5. font-size="inherited-property-value('font-size')". Should be 10pt.

6. font-size="inherited-property-value(provisional-distance-between-starts) div 2". Should be 12pt.

XSL FO Test Definition

54

- Defined for XSL 1.0 Candidate Recommendation testing

```
<test id="expression1" fo="multiple"
xml="inherited-property-value.fo"
results="inherited-property-
value.pdf">Use inherited-property-value()
in expressions for font-size property,
including using function with property
other than font-size and using as part of
a larger expression.</test>
```

FOP

55

- <http://wiki.apache.org/xmlgraphics-fop/HowToCreateLayoutEngineTests>
- Integrates with FOP's JUnit tests
- Input: XML file containing:
 - Complete FO document
 - Assertions about resulting FOP area tree
- Output: Depends on JUnit test runner used

FOP Test

56

```
<testcase>
<info>
  <p>
    This test checks <something>.....
  </p>
</info>
<variables>
  <img>../../resources/images/bgimg300dpi.jpg</img>
</variables>
<fo>
  <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format"
xmlns:svg="http://www.w3.org/2000/svg">
    <fo:layout-master-set>
      <!-- etc. etc. -->
      <fo:block-container background-image="##img">
        <!-- etc. etc. -->
      </fo:block-container>
    </fo:root>
  </fo>
</testcase>
```

57

FOP Test

```
<checks>
  <eval expected="0 0 360000 360000"
xpath="/areaTree/pageSequence/pageViewport/@bounds" desc="page
size"/>
  <true xpath="/areaTree/pageSequence/pageViewport/page[1]"/>
  <true
xpath="not(/areaTree/pageSequence/pageViewport/page[2])"/>
  <eval expected="0 0 360000 360000"
xpath="/areaTree/pageSequence/pageViewport/page[1]/regionViewport/
@rect" desc="region body area"/>
</checks>
</testcase>
```

58

Conclusion

- Profilers
- Debuggers
- Unit testing frameworks
- XSLV Static Validator
- FO testing frameworks

Exhibit 1

Transformation Specification

- art becomes article containing front/article-meta.
- art/@fmt becomes article/@article-type, where:
 - art becomes article
 - bkr becomes book-review
 - ltr becomes correspondence
- art/ttl becomes article-meta/title-group/article-title.
- aut becomes contrib-group/author containing surname and given-names.
- bok becomes product.
- bok/ttl becomes source.
- pub becomes publisher.
- isb becomes isbn.
- pri becomes bold.
- bod becomes body
- par becomes p
- ita becomes i
- bak becomes back
- ref becomes cite

Exhibit 2

Source DTD – art.dtd

```

<!-- Item -->
<!ELEMENT art (ttl, aut*, bok?, bod, bak?) >
<!-- fmt Item format -->
<!ATTLIST art
  fmt (art | bkr | ltr) "art" >

<!-- Title -->
<!ELEMENT ttl (#PCDATA) >

<!-- Author -->
<!ELEMENT aut (fnm, snm) >

<!-- First name -->
<!ELEMENT fnm (#PCDATA) >

<!-- Surname -->
<!ELEMENT snm (#PCDATA) >

<!-- Book details -->
<!ELEMENT bok (ttl, aut, pub, isb, pri) >

<!-- Publisher -->
<!ELEMENT pub (#PCDATA) >

<!-- ISBN -->
<!ELEMENT isb (#PCDATA) >

<!-- Price -->
<!ELEMENT pri (#PCDATA) >

<!-- Body -->
<!ELEMENT bod (par+) >

<!-- Paragraph -->
<!ELEMENT par (#PCDATA | ita)* >

<!-- Italic -->
<!ELEMENT ita (#PCDATA) >

<!-- Back Matter -->
<!ELEMENT bak (ref+) >

<!-- Reference -->
<!ELEMENT ref (#PCDATA) >

```

Exhibit 3

Result DTD – article.dtd

```

<?xml encoding="UTF-8"?>
<!ELEMENT article      (front, body, back?)           >
<!ATTLIST article
      article-type
              (article | book-review | correspondence)
              #REQUIRED >

<!ELEMENT front        (article-meta)                 >
<!ELEMENT body         (p+)                          >
<!ELEMENT article-meta (contrib-group, title-group, product?) >
<!ELEMENT p           (#PCDATA | i)*                 >
<!ELEMENT i           (#PCDATA)                     >
<!ELEMENT contrib-group
              (author+)                               >
<!ELEMENT author      (name)                         >
<!ELEMENT title-group (article-title)                >
<!ELEMENT product     (source | name | publisher | isbn | bold)* >
<!ELEMENT article-title
              (#PCDATA)                              >
<!ELEMENT source      (#PCDATA)                     >
<!ELEMENT name        (surname, given-names)        >
<!ELEMENT publisher   (#PCDATA)                     >
<!ELEMENT isbn        (#PCDATA)                     >
<!ELEMENT bold        (#PCDATA)                     >
<!ELEMENT surname     (#PCDATA)                     >
<!ELEMENT given-names (#PCDATA)                     >
<!ELEMENT back        (cite+)                       >
<!ELEMENT cite        (#PCDATA)                     >

```

Exhibit 4

Sample input – art.xml

```
<art fmt="bkr">
<ttl>Primed for Unicode</ttl>
<aut><fnm>John</fnm><snm>Smith</snm></aut>
<bok><ttl>Unicode: A Primer</ttl>
<aut><fnm>Tony</fnm><snm>Graham</snm></aut>
<pub>IDG Books Worldwide</pub>
<isb>0-7645-4625-2</isb>
<pri>$24.95</pri></bok>
<bod><par>A cast of thousands of interesting characters, but no
mystery. <ita>Two thumbs down</ita>.</par></bod>
</art>
```

Exhibit 5

Sample output – article.xml

```
<?xml version="1.0"?>
<article article-type="book-review">
<front>
<article-meta>
<contrib-group/>
<title-group>
<article-title>Primed for Unicode</article-title>
</title-group>
<product>
<source>Unicode: A Primer</source>
<name>
<surname>Graham</surname>
<given-names>Tony</given-names></name>
<publisher>IDG Books Worldwide</publisher>
<isbn>0-7645-4625-2</isbn>
<bold>$24.95</bold>
</product>
</article-meta>
</front>
<body>
<p>A cast of thousands of interesting characters, but no mystery. Two thumbs down.</p>
</body>
</article>
```

Exhibit 6

Stylesheet – art2article.xsl

```

<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">

    <xsl:output method="xml"/>

    <xsl:strip-space elements="itm aut body bak"/>

    <xsl:template match="art">
        <article>
            <xsl:attribute name="article-type">
                <xsl:choose>
                    <xsl:when test="@fmt = 'bkr'">book-review</xsl:when>
                    <xsl:when test="@fmt = 'let'">letter</xsl:when>
                    <xsl:otherwise>article</xsl:otherwise>
                </xsl:choose>
            </xsl:attribute>
            <front>
                <article-meta>
                    <contrib-group>
                        <xsl:for-each select="auth">
                            <contrib><xsl:apply-templates select="."/;</contrib>
                        </xsl:for-each>
                    </contrib-group>
                    <xsl:apply-templates select="ttl | //auth | //bok"/>
                </article-meta>
            </front>
            <xsl:apply-templates select="bod | bac"/>
        </article>
    </xsl:template>

    <xsl:template match="ttl">
        <title-group>
            <article-title><xsl:apply-templates/></article-title>
        </title-group>
    </xsl:template>

    <xsl:template match="bok/ttl">
        <source><xsl:apply-templates/></source>
    </xsl:template>

    <xsl:template match="bok">
        <product>
            <xsl:apply-templates/>
        </product>
    </xsl:template>

    <xsl:template match="isb">
        <issn><xsl:apply-templates/></issn>
    </xsl:template>

```

```

</xsl:template>

<xsl:template match="pub">
  <publisher><xsl:apply-templates/></publisher>
</xsl:template>

<xsl:template match="isb">
  <isbn><xsl:apply-templates/></isbn>
</xsl:template>

<xsl:template match="pri">
  <bold><xsl:apply-templates/></bold>
</xsl:template>

<!-- The client is crazy for doing this this way. -->
<xsl:template match="aut">
  <name>
    <surname><xsl:apply-templates select="snm"/></surname>
    <given-names><xsl:apply-templates select="fnm"/></given-names>
  </name>
</xsl:template>

<xsl:template match="//bod">
  <body><xsl:apply-templates/></body>
</xsl:template>

<xsl:template match="par">
  <p><xsl:value-of select="."/></p>
</xsl:template>

<!--
  This used to be needed, don't know why:
  <xsl:template match="blue">
    <p><xsl:value-of select="."/></p>
  </xsl:template>
-->

  <xsl:template match="ita">
    <i><xsl:apply-templates/></i>
  </xsl:template>

</xsl:stylesheet>

```

Exercise 1

"Black box" testing

What are some useful tests for the specification in Exhibit 1?

-
-
-
-

Exercise 2

"White box" testing

What are some useful tests for the stylesheet in Exhibit 6?

-
-
-
-

Exercise 3

Other Errors

What errors in the stylesheet in Exhibit 6 would not be caught by unit testing?

-
-
-
-
-
-
-
-