The road to an XSLT/XQuery IDE

George Bina, Syncro Soft - oXygen® XML Editor

Overview

- Checking for errors and error reporting
- Navigation and refactoring
- Advanced content completion
- Running configurations
- Debugging and profiling
- Visual editing
- Documentation and unit testing
- Other features

Checking for errors and error reporting

Checking for errors and error reporting

General problems

- Are there any errors?
- Errors locations
- Errors descriptions

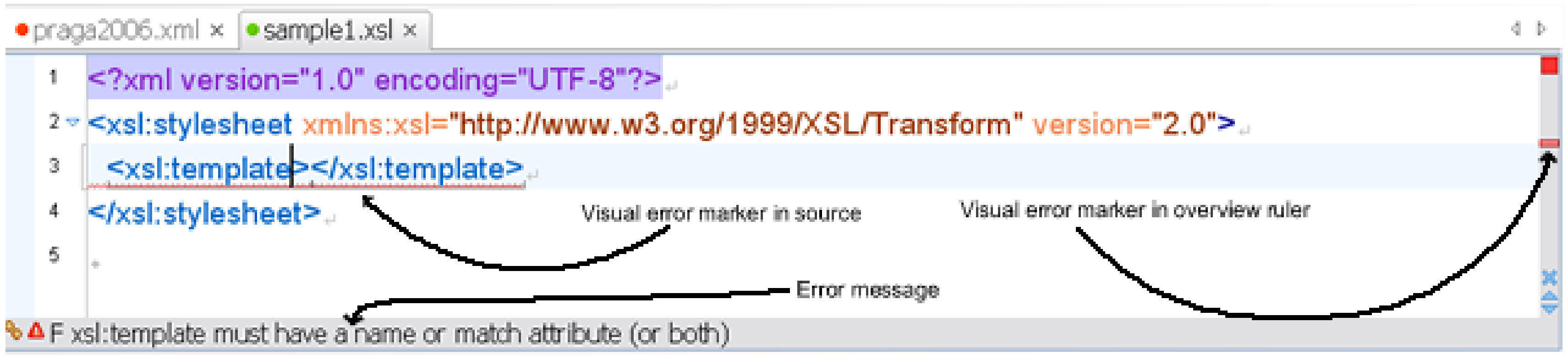
Validation

- Continuous validation
- Validation on demand
- Background validation

Reporting

- Table or list of errors
- Visual error markers

Error markers in oXygen



praga2006.xml × sample1.xsl ×

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
  <xsl:template></xsl:template>
</xsl:stylesheet>
```

F xsl:template must have a name or match attribute (or both)

Visual error marker in source
Visual error marker in overview ruler
Error message

Specific problems

- Syntax and/or structure checking
- More powerful error checking

Modules are not required to be themselves valid

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="*">
    <xsl:if test="$handleElements='true'">
      <xsl:apply-templates/>
    </xsl:if>
  </xsl:template>
</xsl:stylesheet>
```

Specific problems (Continued)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:variable name="handleElements"
    select="'true'"/>
  <xsl:include href="sample2module.xsl"/>
</xsl:stylesheet>
```

Main documents/module documents

Automatic detection of main/module files

- **The user has no control**

Let the user mark the files as main/module files

- **The user should take specific action**
- **Powerful actions**

Main documents - XML specific

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE test [
  <!ENTITY x "x">
  <!ENTITY module SYSTEM "module.xml">
  <!ELEMENT test (module)+>
  <!ELEMENT module ANY>
  ]>
<test>
  &module;
</test>

<?xml version="1.0" encoding="UTF-8"?>
<module>
  &x;
</module>
```

Navigation and refactoring

Navigation and refactoring

Navigation

- Go to definition
- Find references
- Follow includes/imports
- Outlining

Issues

- Handle invalid source
- Scoping
  - Current file
  - All the project
  - Start from a file
  - User defined working sets

Refactoring

Semantic changes

Rename - most used and most useful

Changes in multiple locations

Diff before and after refactoring versions

Scoping

- Current file
- All the project
- Start from a file
- User defined working sets

The documents are wellformed

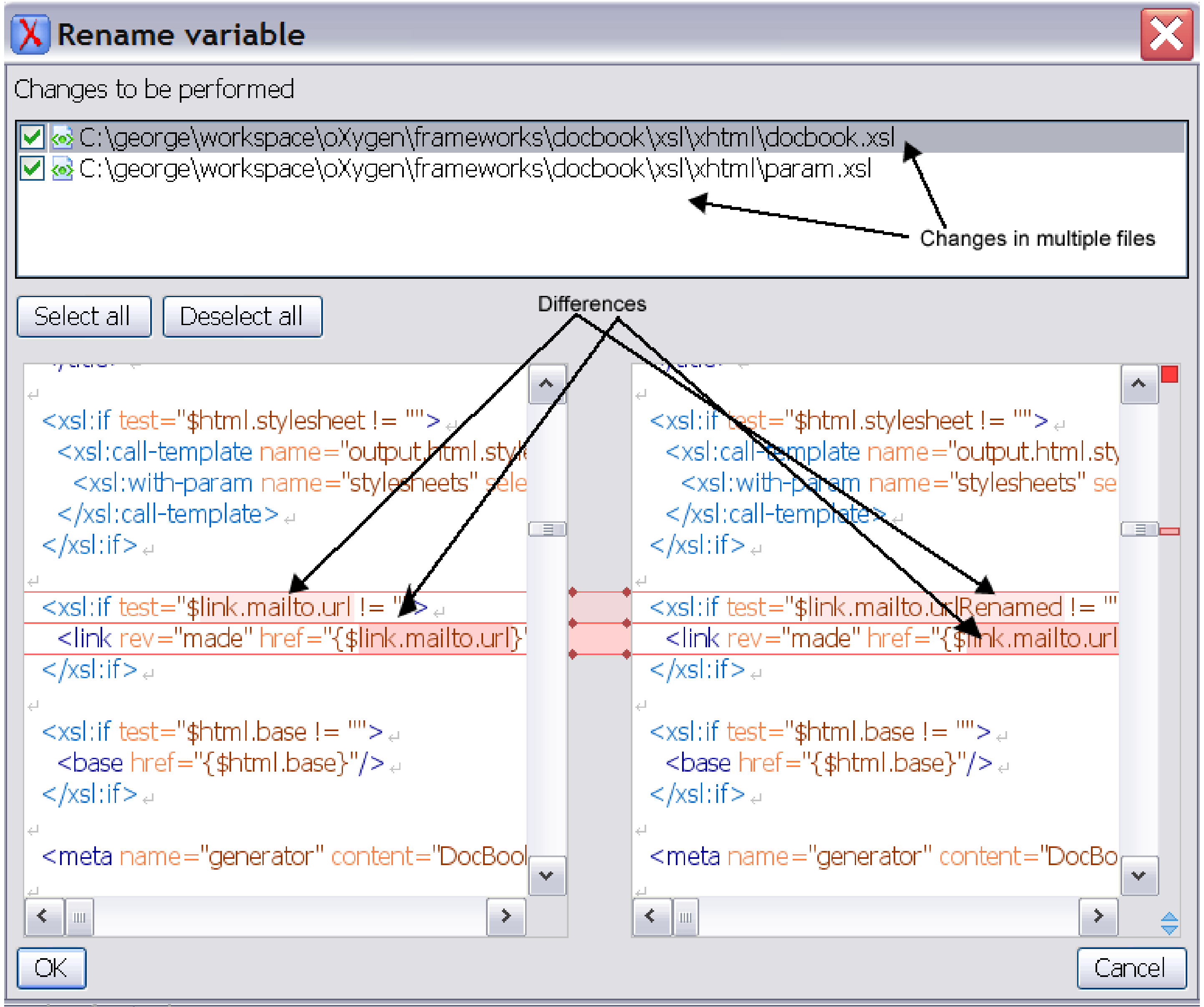
Rename

Applies to all named components

Determine the component name and type

Sample: rename a variable with oXygen

Rename (Continued)



C:\george\workspace\oXygen\frameworks\docbook\xsl\xhtml\docbook.xsl
C:\george\workspace\oXygen\frameworks\docbook\xsl\xhtml\param.xsl

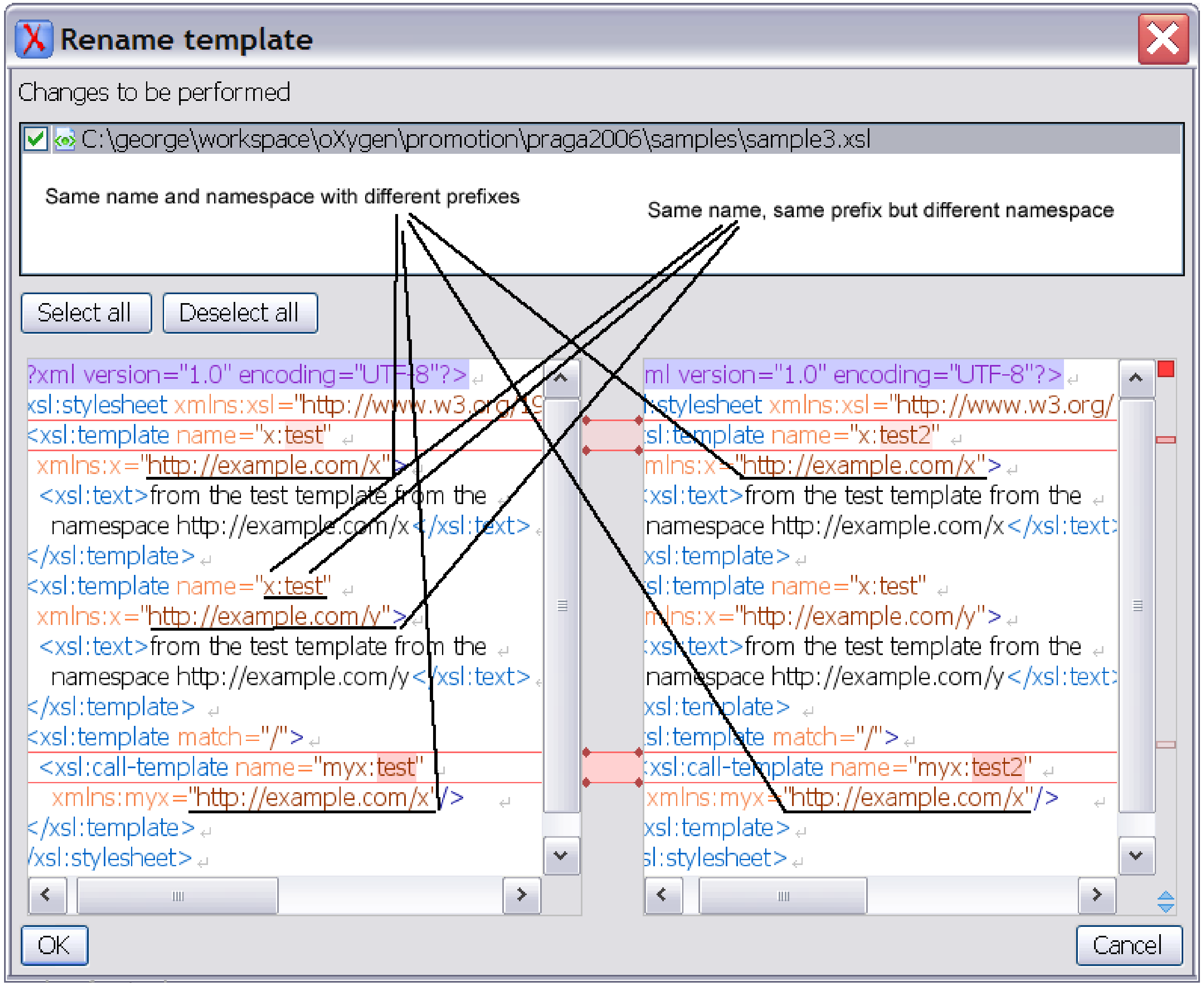
```
<xsl:if test="$html.stylesheet != ''">
  <xsl:call-template name="output.html.style
    <xsl:with-param name="stylesheets" sele
  </xsl:call-template>
</xsl:if>

<xsl:if test="$link.mailto.url != ''">
  <link rev="made" href="{$link.mailto.url}'
</xsl:if>

<xsl:if test="$html.base != ''">
  <base href="{$html.base}"/>
</xsl:if>

<meta name="generator" content="DocBool
```

```
<xsl:if test="$html.stylesheet != ''">
  <xsl:call-template name="output.html.sty
    <xsl:with-param name="stylesheets" se
  </xsl:call-template>
</xsl:if>

<xsl:if test="$link.mailto.urlRenamed != ''
  <link rev="made" href="{$link.mailto.url
</xsl:if>

<xsl:if test="$html.base != ''">
  <base href="{$html.base}"/>
</xsl:if>

<meta name="generator" content="DocBo
```

Rename (Continued)

Sample: Rename a template from a namespace

Rename (Continued)



Rename template
Changes to be performed
C:\george\workspace\oXygen\promotion\praga2006\samples\sample3.xsl
Same name and namespace with different prefixes
Same name, same prefix but different namespace
Select all
Deselect all
OK
Cancel

```
?xml version="1.0" encoding="UTF-8"?>
xsl:stylesheet xmlns:xsl="http://www.w3.org/19
<xsl:template name="x:test"
 xmlns:x="http://example.com/x">
  <xsl:text>from the test template from the
   namespace http://example.com/x</xsl:text>
</xsl:template>
<xsl:template name="x:test"
 xmlns:x="http://example.com/y">
  <xsl:text>from the test template from the
   namespace http://example.com/y</xsl:text>
</xsl:template>
<xsl:template match="/">
  <xsl:call-template name="myx:test"
   xmlns:myx="http://example.com/x"/>
</xsl:template>
/xsl:stylesheet>
```

```
ml version="1.0" encoding="UTF-8"?>
:stylesheet xmlns:xsl="http://www.w3.org/
sl:template name="x:test2"
mlns:x="http://example.com/x">
:xsl:text>from the test template from the
namespace http://example.com/x</xsl:text:
xsl:template>
sl:template name="x:test"
mlns:x="http://example.com/y">
:xsl:text>from the test template from the
namespace http://example.com/y</xsl:text:
xsl:template>
sl:template match="/">
:xsl:call-template name="myx:test2"
xmlns:myx="http://example.com/x"/>
xsl:template>
sl:stylesheet>
```

Rename (Continued)

Other refactoring actions

Extract as component

Replace component with code

Examples

- **create template from selection**
- **create stylesheet from selection**
- **extract attributes as xsl:attributes**

Extract selection as template example

Same context

- **Same namespace context**
- **Same variables/parameters**

Initial code

```
<xsl:template match="/">
  <result>
    <xsl:variable name="elements" select="/*/*"/>

    <xsl:for-each select="$elements">
      <xsl:variable name="pos" select="position()"/>
      <xsl:value-of select="name()"/>
      <xsl:text>-</xsl:text>
      <xsl:value-of select="$pos"/>
    </xsl:for-each>

  </result>
</xsl:template>
```

After refactoring

```
<xsl:template match="/">
  <result>
    <xsl:variable name="elements" select="/*/*"/>

    <xsl:call-template name="printElements">
      <xsl:with-param name="elements"
        select="$elements"/>
    </xsl:call-template>

  </result>
</xsl:template>
```

After refactoring (continued)

```
<xsl:template name="printElements">

  <xsl:param name="elements"/>

  <xsl:for-each select="$elements">
    <xsl:variable name="pos" select="position()"/>
    <xsl:value-of select="name()"/>
    <xsl:text>-</xsl:text>
    <xsl:value-of select="$pos"/>
  </xsl:for-each>

</xsl:template>
</xsl:stylesheet>
```

Advanced content completion

Advanced content completion

Content completion

Documentation for proposals

Static proposals

- Instructions
- Keywords
- Built-in functions

Dynamic proposals

- User defined functions or templates
- Variables and parameters
- Output elements and attributes

Abbreviations/Code templates

XPath content completion

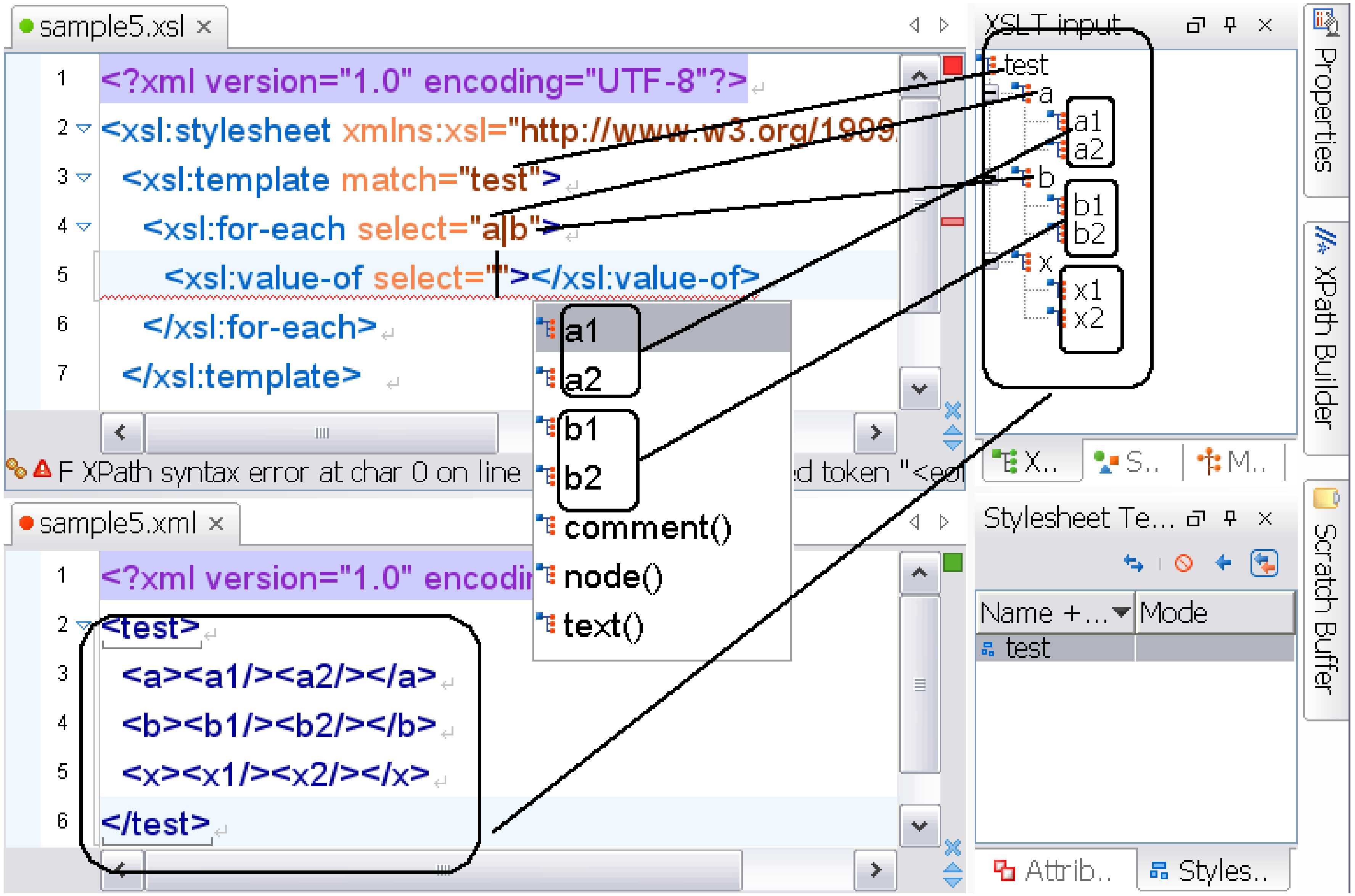
Static proposals

- **XPath functions**
- **Axes**

Dynamic proposals

- **Variables**
- **Parameters**
- **Name tests**

XPath proposals example



```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999
  <xsl:template match="test">
    <xsl:for-each select="a|b">
      <xsl:value-of select=""></xsl:value-of>
    </xsl:for-each>
  </xsl:template>
```

```
F XPath syntax error at char 0 on line          d token "<eo
```

```
<?xml version="1.0" encodi
<test>
  <a><a1/><a2/></a>
  <b><b1/><b2/></b>
  <x><x1/><x2/></x>
</test>
```

XPath proposals example (Continued)

Running configurations

Multiple scenarios/configurations

Reuse scenarios

Multiple processors/servers support

- **Match the configuration used in production**
- **Specific processor extensions**

FOP transformation support

Debugging and profiling

Debugging and profiling

Debugging

Basic support

- **Stepping and breakpoints**
- **Watch variables**
- **Stack and trace views**

More advanced debugging actions

- **Map from result to instructions**
- **Conditional breakpoints**
- **Change variables**

Issues

- **No common debugging interface**
- **Source different than the actual processor execution**

Profiling

Hot spots

Invocation tree

Issues

- **Similar issues as for debugging**
- **Difficult to compute the actual processor time**

  The effort may not be justified by the results

Visual editing

Drag and drop editing

Visual mappers

Documentation and unit testing

Documentation

- **Javadoc like reports**
- **Use documentation**
  - **When browsing the source**
  - **During content completion**
  - **When configuring running configurations**

Unit testing

- **Edit unit tests**
- **Run unit tests**

Other features

A lot of simple but useful actions

- **Transform an empty element in a non empty one**
- **Jump to the next editing position**
- **Smart indenting**
- **Element selection**
- **Content selection**
- **Indent on paste**
- **Toggle comment**
- **etc.**

Conclusion

Good support from tools

Missing features

- **Working with module documents**
- **Complete coverage of refactoring actions**
- **Support for documentation**
- **Support for unit testing**