XML Navigation, Transformation

# Schematron

Lecture "XML in Communications"
Chapter 10

Dr.-Ing. Jesper Zedlitz
Research Group for Communication Systems
Dept. of Computer Science
Christian-Albrechts-University in Kiel

# Acknowledgement



- This chapter is based on:

  Roger L. Costello: XML Technologies Course

  http://www.xfront.com/files/tutorials.html
  

# Recommended Reading

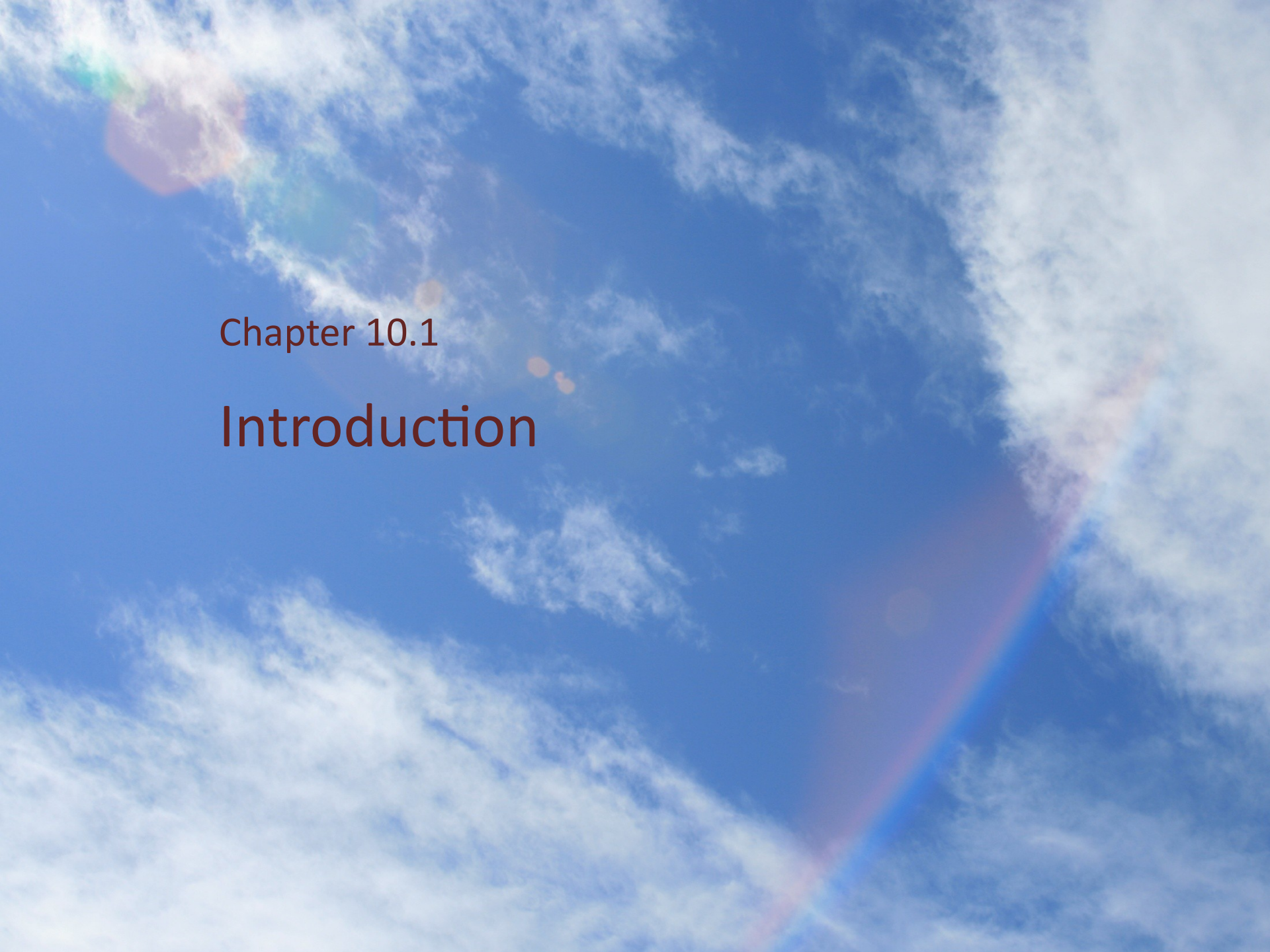


- ISO/IEC-Standard 19757-3:2006
  http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html
- http://www.schematron.com/

# Overview



1. Introduction
2. Basic Features
3. Schematron Assertions
4. Inside Schematron



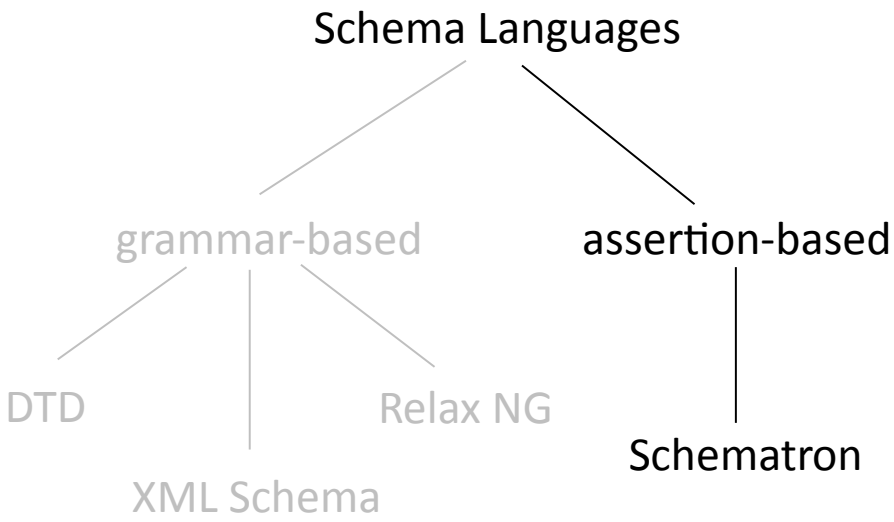
Chapter 10.1

# Introduction

# Introduction



- Schematron is assertion-based rather than grammar-based
  - grammar-based approaches take a closed approach: everything not explicitly allowed is treated as invalid
  - assertion-based approaches take an open approach: everything not explicitly disallowed is treated as valid



# Introduction



- Schematron is part of ISO/IEC's 19757 effort:
  Document Schema Definition Languages (DSDL)
  - "The main objective of DSDL is to bring together different validation-related tasks and expressions to form a single extensible framework that allows technologies to work in series or in parallel to produce a single or a set of validation results." (http://dsdl.org/)
  - Schematron is undergoing standarization as one part of this: "Rule-based validation – Schematron" – part 3
  - Namespace is http://purl.oclc.org/dsdl/schematron

# Introduction



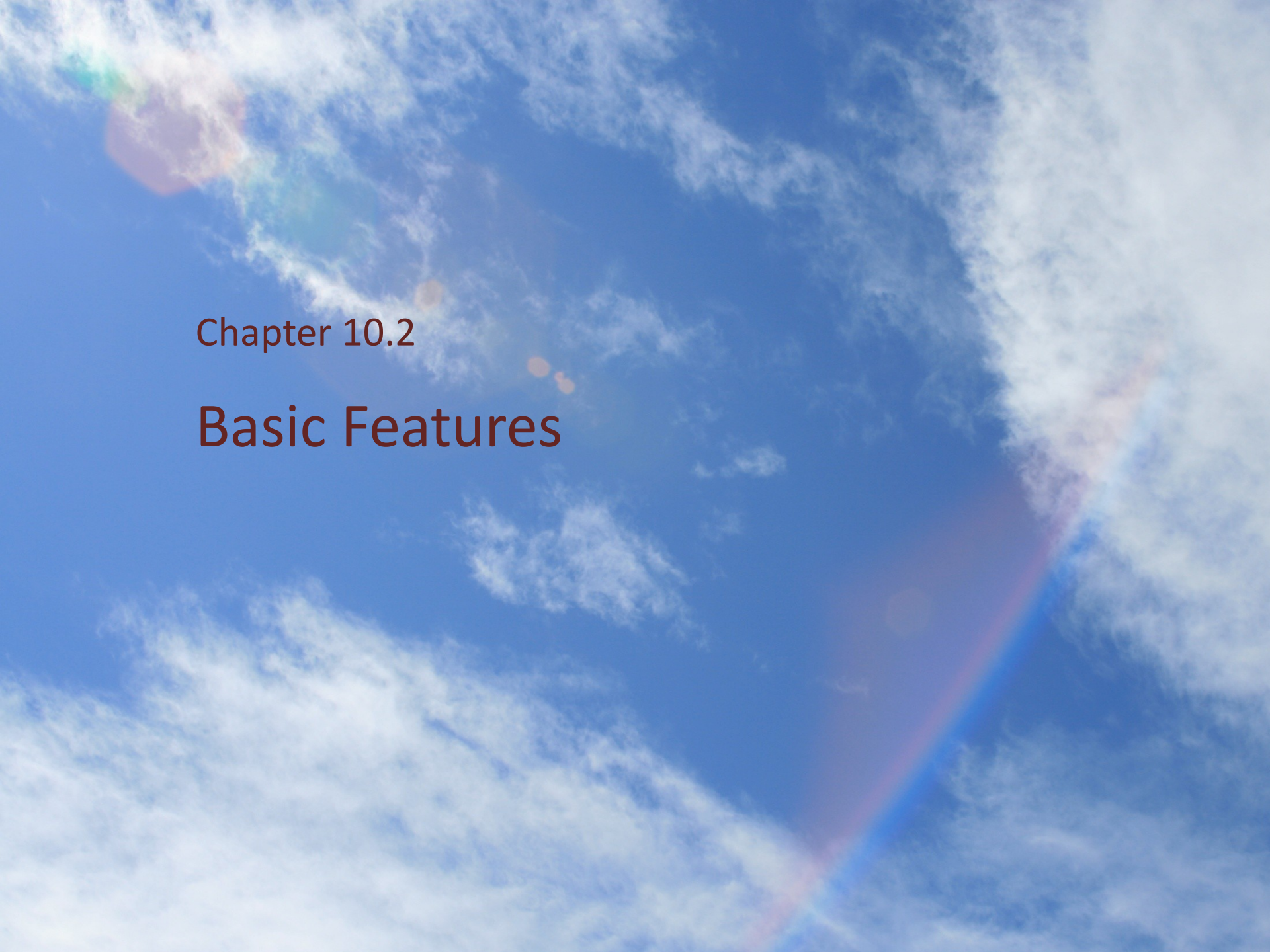
- Schematron not be used as the only validation method
  - use grammar-based method for structure and value constraints
  - use Schematron for constraints
    that can't be described in grammar-based methods, such as
    - constraints between multiple elements/attributes
    - validation across documents (using document function)

    → co-constraints

# Introduction



- Co-constraint: constraints that exist between data
  - element-to-element co-constraints
  - element-to-attribute co-constraints
  - attribute-to-attribute co-constraints
  - includes element/attribute existence checking
  - includes formula checking:
    validity checking may require performing an algorithm on the data,
    e.g. "% values in column 5 of table 'results' add up to 100%"
  - includes co-constraint checking "within" single XML document,
    or "across" multiple XML documents
    (intra- and inter-document co-constraints)



Chapter 10.2

# Basic Features

# Basic Features



- Schematron core language elements
  - Assertions
    - conditions to be tested such as existence and values of elements/attributes
    - `assert` and `report` elements
  - Rules
    - groups of assertions
    - selects the set of context nodes under which they are evaluated

# Basic Features

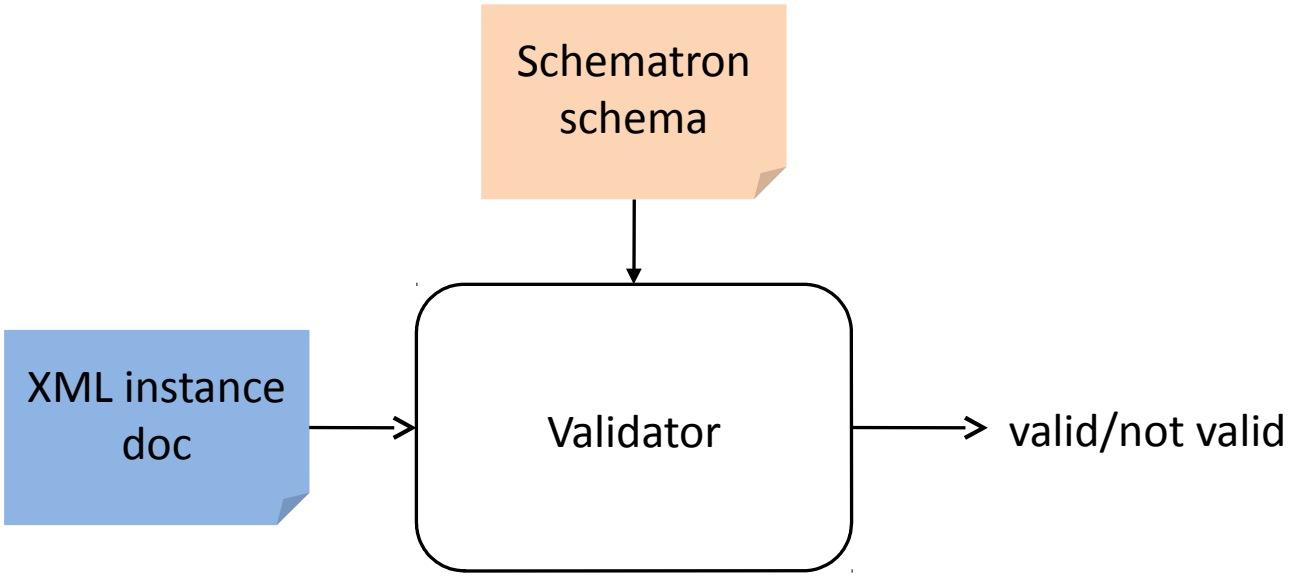


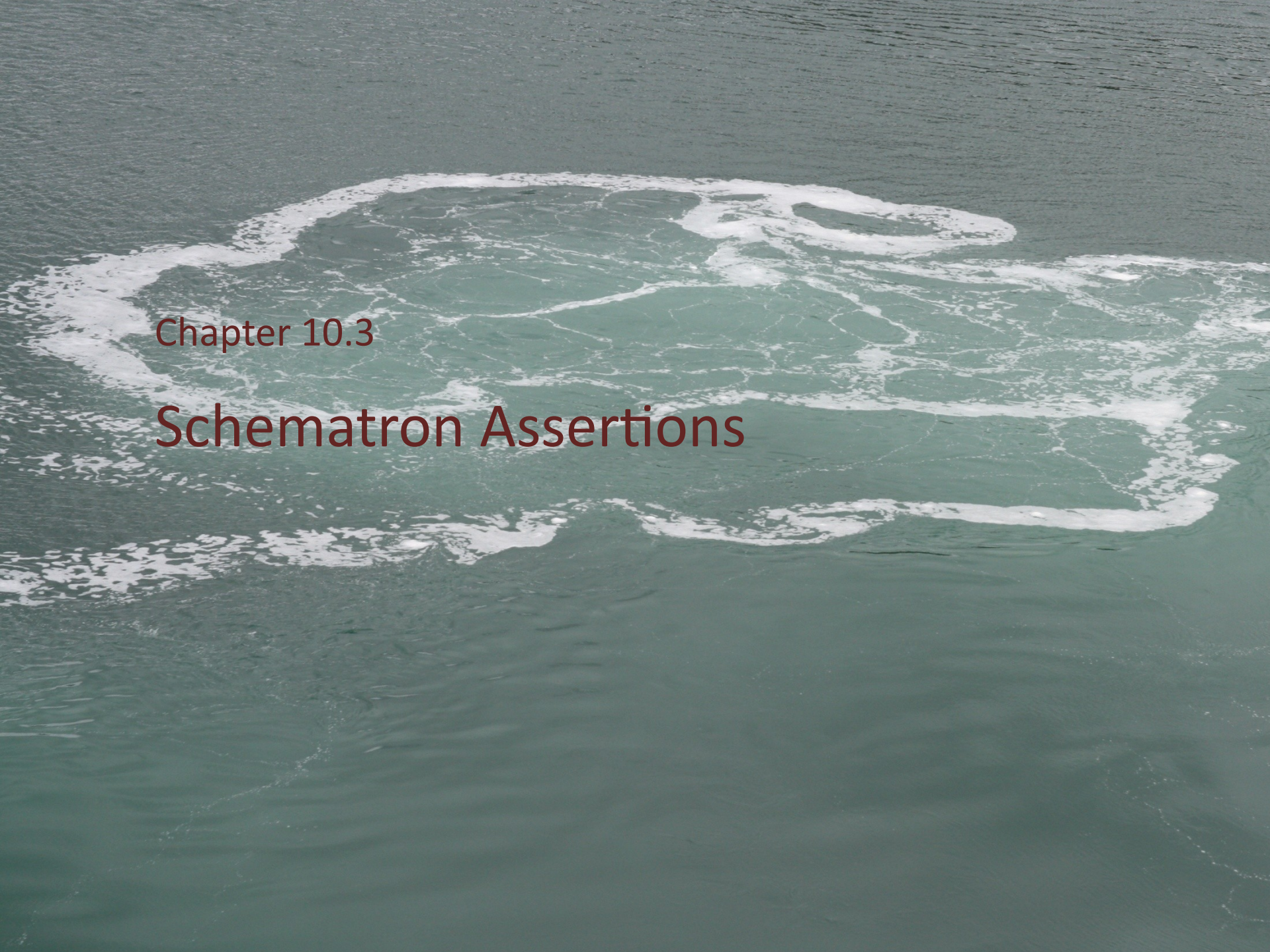
- Schematron language elements
  - Patterns
    - groups of rules with an id (used by phases, see below)
  - Phases
    - groups of patterns (specified by their id)
      that allow evaluating only the rules in those patterns

# Basic Features



- Schematron validation





Chapter 10.3

# Schematron Assertions

# Schematron Assertions



- Assertions use XPath expressions
  - can validate anything that can be expressed as a boolean XPath expression
- Assertion location
  - can be in a separate file, typically with a ".sch" extension
  - can be embedded within other schema files
- Validator
  - XSLT-based validators can easily be implemented
  - can also been implemented without using XSLT for better performance

# Schematron Assertions



- Formal stuff
  - A Schematron schema is an XML document.
  - The Schematron elements are in this namespace: http://purl.oclc.org/dsdl/schematron

```
<?xml version="1.0"?>
<sch:schema
xmlns:sch="http://purl.oclc.org/dsdl/schematron">
…
</sch:schema>
```

  - By convention, the file name of a Schematron schema has the suffix ".sch"

# Schematron Assertions



- Patterns
  - A Schematron schema is comprised of one or more pattern elements (with optional id):

```
<?xml version="1.0"?>
<sch:schema xmlns:sch=" http://purl.oclc.org/dsdl/schematron">
  <sch:pattern id="…"> … </sch:pattern>
  <sch:pattern id="…"> … </sch:pattern>
  <sch:pattern id="…"> … </sch:pattern>
</sch:schema>
```

  - The purpose of each pattern is to specify a relationship that must exist among information in the XML instance document.

# Schematron Assertions



- Rules
  - A pattern element is comprised of one or more rule elements:

```
<sch:pattern id="…">
    <sch:rule context="…"> … </sch:rule>
    <sch:rule context="…"> … </sch:rule>
    <sch:rule context="…"> … </sch:rule>
</sch:pattern>
```

  - The purpose of each rule is to specify a context node in the XML instance document, and assert a relationship relative to that context node.

# Schematron Assertions



- Assertions
  - A rule element is comprised of one or more assert elements:

```
<sch:rule context="…">
  <sch:assert test="boolean-XPath">
    … message …
  </sch:assert>
  <sch:assert test="boolean-XPath">
    … message …
  </sch:assert>
</sch:rule>
```

  - The purpose of an assert element is to state a relationship that must exist in the XML instance document.

# Schematron Assertions



- Assertions

```
<sch:assert test="boolean-XPath">
  … message …
</sch:assert>
```

  - The value of the `test` attribute is an XPath expression.
  - A Schematron validator checks the XML instance document against the XPath expressions of every <`assert`> element within a rule.
  - The contents of the <`assert`> element is the text of message displayed to the user in case the assertion does not hold.

# Schematron Assertions



- Reports

```
<sch:report test="boolean-XPath">
  … message …
</sch:assert>
```

  - The value of the `test` attribute is an XPath expression.
  - A Schematron validator checks the XML instance document against the XPath expressions of every `<report>` element within a rule.
  - The contents of the `<report>` element is the text of message displayed to the user in case the assertion does hold.

# Schematron Assertions



- Example 1

```
<?xml version="1.0"?>
<Document classification="secret">
  <Para classification="unclassified">
    Schematron is a schema language for XML.
  </Para>
</Document>
```

  - Assume the following Security Classification Policy:
    - No <Para> element may have a classification value higher than the <Document>'s classification value.
    - Possible classification values, from highest to lowest: top-secret — secret — confidential — unclassified

# Schematron Assertions



- Example 1: first step

```
<sch:rule context="Para[@classification='top-secret']">
  <sch:assert test="/Document/@classification='top-secret'">
    If there is a Para labeled "top-secret",
    then the Document must be labeled top-secret.
  </sch:assert>
</sch:rule>
```

  - Read as: Within the context of a `<Para>` element whose classification value is 'top-secret', I assert that the classification value of the `<Document>` element must be 'top-secret'.

# Schematron Assertions



- Example 1: second step

```
<sch:rule context="Para[@classification='secret']">
  <sch:assert test="(/Document/@classification='top-secret') or
                    (/Document/@classification='secret')">
    If there is a Para labeled "secret", then the Document
    must be labeled either secret or top-secret.
  </sch:assert>
</sch:rule>
```

  - Read as: Within the context of a <Para> element whose classification value is 'secret', I assert that the classification value of the <Document> element must be either 'secret' or 'top-secret'.

# Schematron Assertions



- Example 1: third step

```
<sch:rule context="Para[@classification='confidential']">
  <sch:assert test="(/Document/@classification='top-secret') or
                    (/Document/@classification='secret') or
                    (/Document/@classification='confidential')">
    If there is a Para labeled "confidential" then the Document
    must be labeled either confidential, secret or top-secret.
  </sch:assert>
</sch:rule>
```

  - Read as: Within the context of a <Para> element whose classification value is 'confidential', I assert that the classification value of the <Document> element must be either 'top-secret', 'secret' or 'confidential'.

# Schematron Assertions



- Example 1: final solution

```
<sch:pattern id="SecurityClassificationPolicy">
  <sch:p>A Para's classification value cannot be more sensitive
         than the Document's classification value.</sch:p>
  <sch:rule context="Para[@classification='top-secret']">   …
</sch:rule>
  <sch:rule context="Para[@classification='secret']">   …
</sch:rule>
  <sch:rule context="Para[@classification='confidential']">    …
</sch:rule>
</sch:pattern>
```

  - Note that a pattern can be annotated using the <sch:p> element.
  - Note that a pattern element wraps the rule elements, and the pattern element is given an identifier.

# Schematron Assertions



- Example 2

```
<?xml version="1.0"?>
<election>
  <candidates>
    <candidate name="John Doe" percentage="51"/>
    <candidate name="Joe Blogs" percentage="49"/>
  </candidates>
</election>
```

# Schematron Assertions



- Example 2: validation schema

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron">
  <sch:pattern>
    <sch:rule context="candidates">
      <sch:assert test="sum(candidate/@percentage)='100'">
        The sum of the percentage attributes for the
        candidate elements must be 100.
      </sch:assert>
      <sch:report test="count(candidate) &lt; 2">
        There must be at least two candidate elements inside
        the candidates element.
      </sch:report>
    </sch:rule>
  </sch:pattern>
</sch:schema>
```

# Schematron Assertions



- Example 3

```
<?xml version="1.0" encoding="UTF-8"?>
<?oxygen SCHSchema="duplicateRole.sch"?>
<movies>
    <movie>
        <title>Elf</title>
        <actor name="Will Ferrell" role="Buddy"/>
        <actor name="James Caan" role="Walter"/>
        <actor name="Bob Newhart" role="Papa Elf"/>
        <actor name="Edward Asner" role="Santa"/>
        <actor name="Mary Steenburgen" role="Claus"/>
        <actor name="Zooey Deschanel" role="Jovie"/>
        <actor name="Mark Volkmann" role="Buddy"/>
        <actor name="Edward Asner" role="Mr. Grant"/>
    </movie>
</movies>
```

# Schematron Assertions



- Example 3: validation schema

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron">
  <sch:pattern>
    <sch:rule context="actor">
      <sch:report test="@role=preceding-sibling::actor/@role">
        Role assigned twice or more!
      </sch:report>
    </sch:rule>
  </sch:pattern>
</sch:schema>
```

# Schematron Assertions



- Example 3: validation schema improved

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron">
  <sch:pattern id="single">
    <sch:rule context="actor">
      <sch:report test="@role=preceding-sibling::actor/@role"
      diagnostics="duplicateActorRole"/>
    </sch:rule>
  </sch:pattern>
  <sch:diagnostics>
    <sch:diagnostic id="duplicateActorRole">
      More than one actor plays the role
      <sch:value-of select="@role"/>. A duplicate is named
      <sch:value-of select="@name"/>.
    </sch:diagnostic>
  </sch:diagnostics>
</sch:schema>
```

# Schematron Assertions



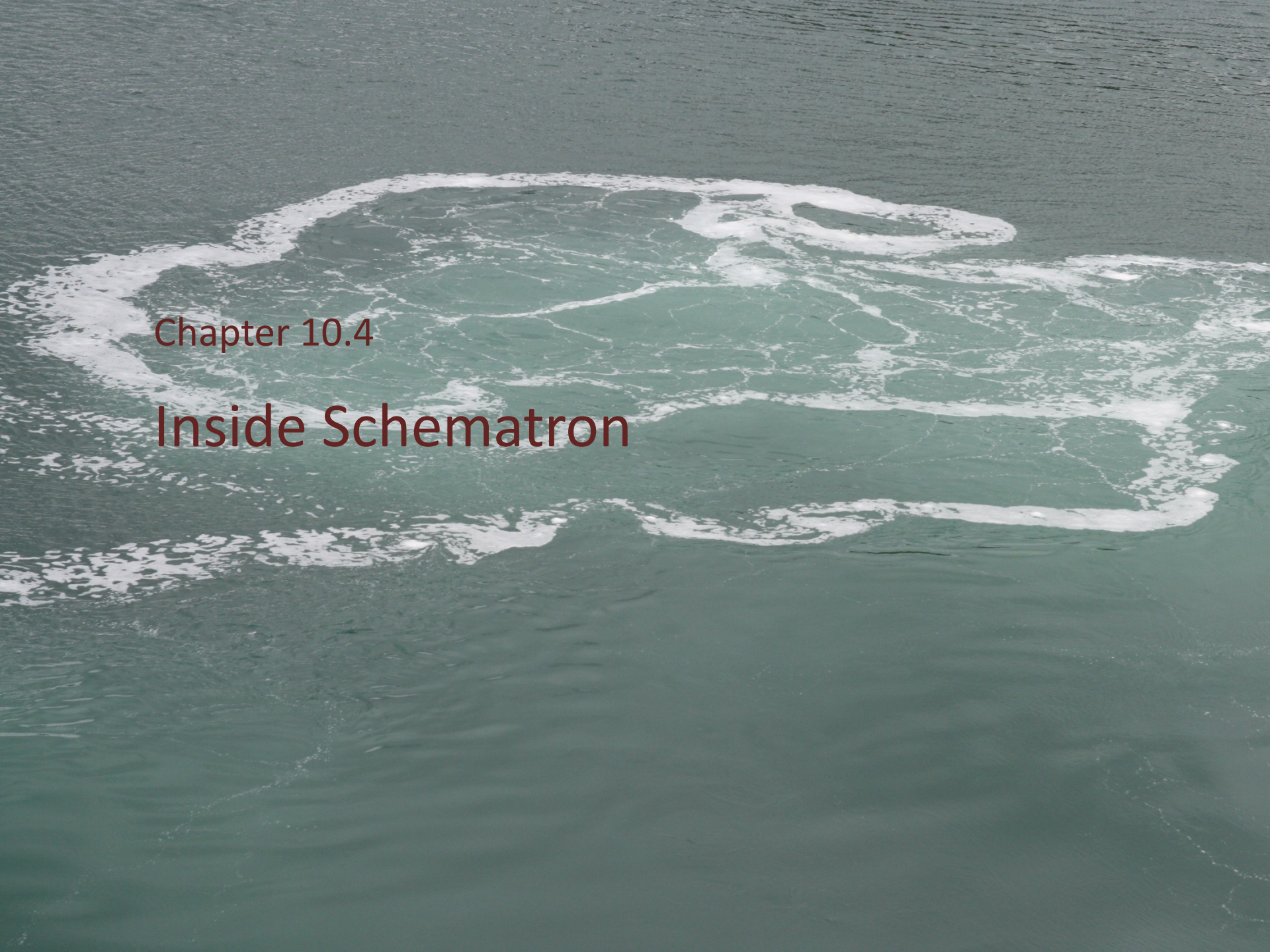
- Phases
  - Optional, named groups of patterns
    - Can evaluate only the rules of specific patterns instead of evaluating all rules in all patterns
    - by specifying a phase id
  - Options for specifying the phase to evaluate include
    - command-line option
    - selection in a GUI
    - parameter in API call

# Schematron Assertions



- Phases

```
<phase id="phase-id">
  <active pattern="pattern-id"/>
  …more active elements go here ...
</phase>
```



Chapter 10.4

# Inside Schematron

# Inside Schematron



- Processing order
  - Only the order of rule elements is significant.
    - For each context node, only the first matching rule within a pattern is used.
  - The order of other things is implementation dependent
    - order in which context nodes are validated
    - order in which phases are evaluated
    - order in which patterns within a phase are evaluated
    - order in which asserts and reports within a rule are tested

# Inside Schematron

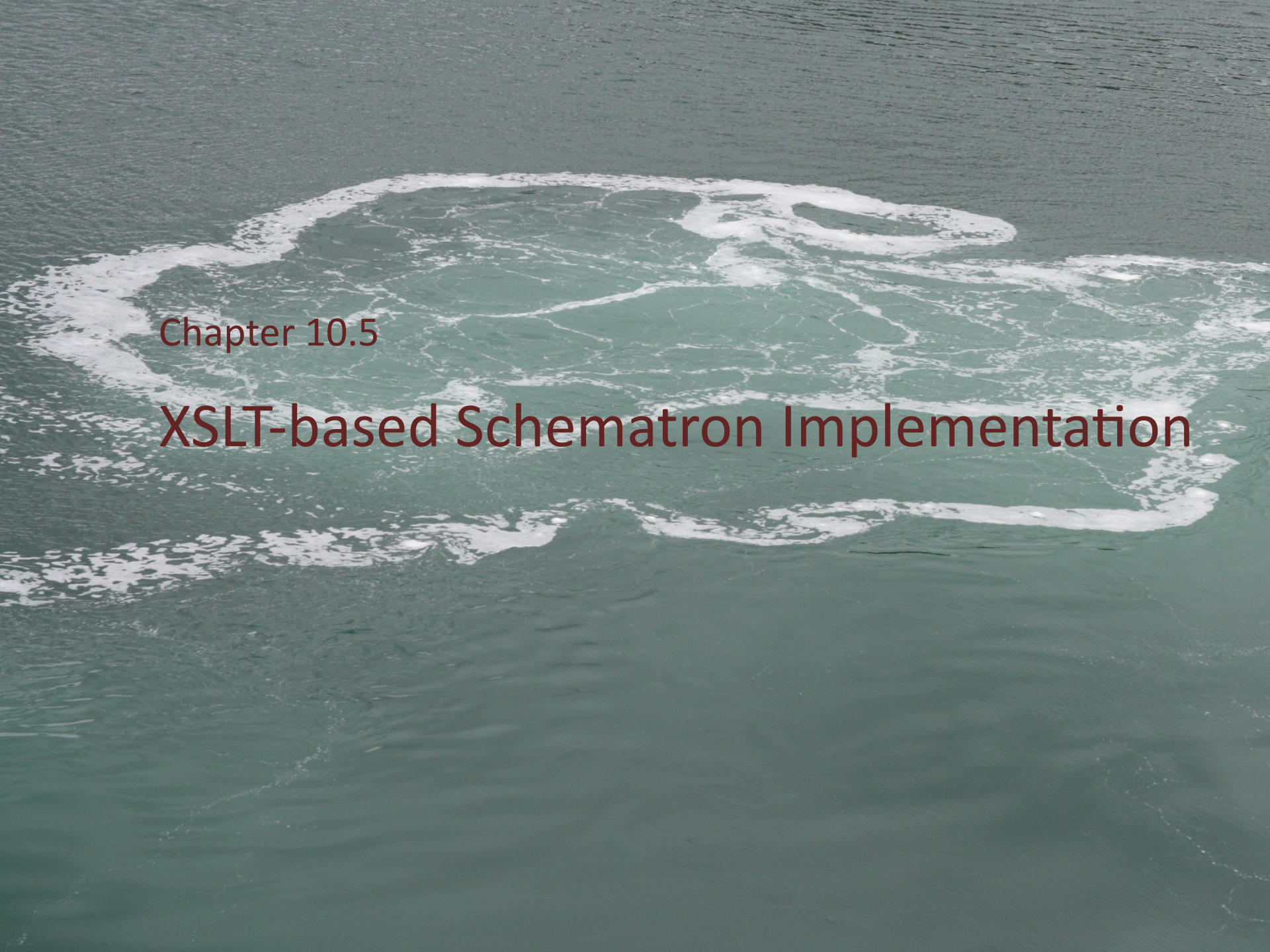


- Processing

For each context node in the document being validated
For each phase being evaluated
For each pattern in the phase
Find the first rule that matches the context node
For each assert and report in the rule
Perform the test
If an assert test fails or a report test passes
Output the message inside it
If diagnostics are enabled and there are associated diagnostics
Output the diagnostic messages

if phases aren't used then all patterns are evaluated

diagnostics provide information beyond messages in assert and report elements such as actual/expected values and hints to repair the document



Chapter 10.5

# XSLT-based Schematron Implementation

# XSLT-based Schematron Implementation



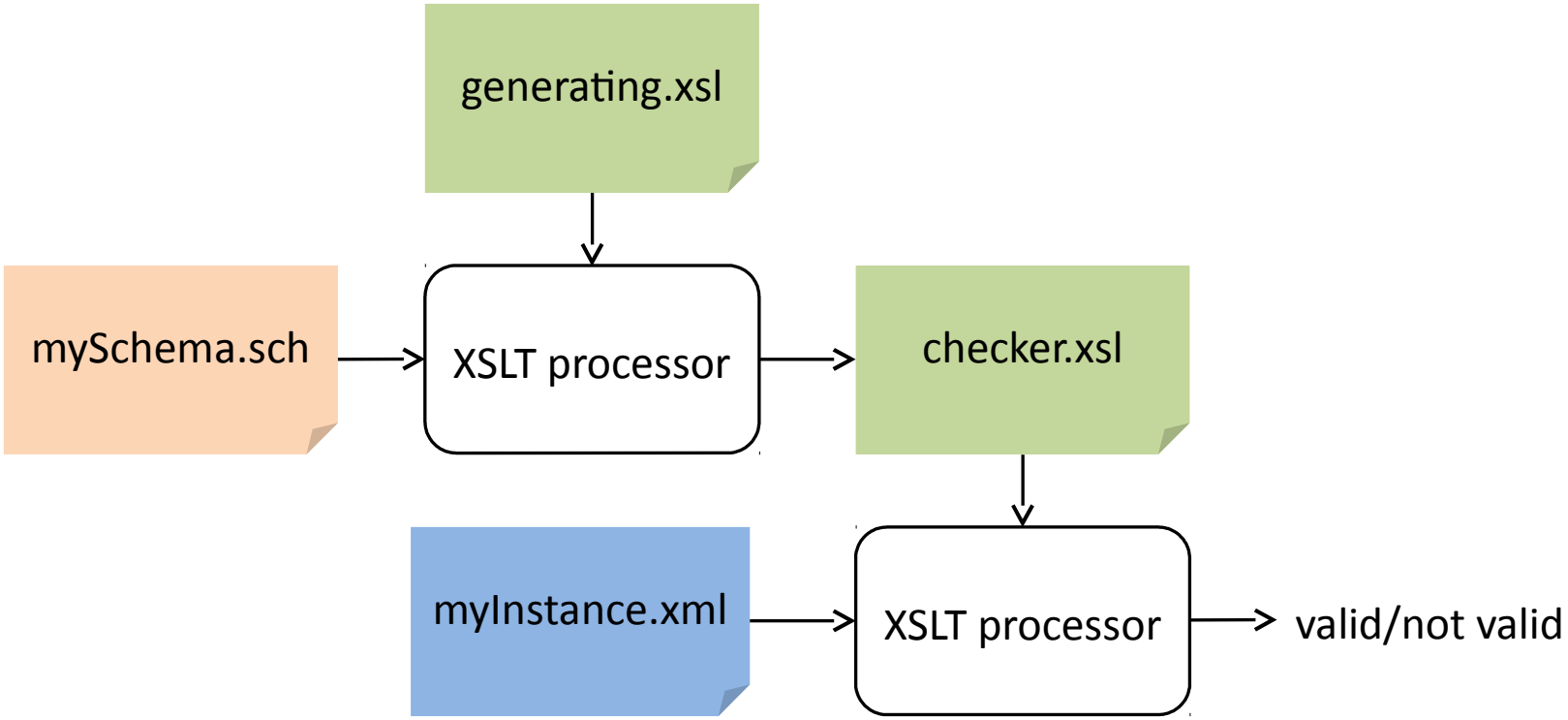
- Slides after:

  Bob DuCharme:
  Schematron 1.5: Looking Under the Hood.
  October 06, 2004
  http://www.xml.com/pub/a/2004/10/05/tr.html

# XSLT-based Schematron Implementation



- XSLT-based implementation



# XSLT-based Schematron Implementation



- Problem
  - "If a stylesheet must add `xsl:template` elements to the result tree, you need a way to distinguish the `xsl:template` elements that tell the generating stylesheet what to do from the `xsl:template` elements that the generating stylesheet adds to the generated stylesheet in the result tree."
  - Concept:
    - declare a dummy namespace
    - include an `xsl:namespace-alias` element in the generating stylesheet

# XSLT-based Schematron Implementation



- What is namespace aliasing?
  - The `<xsl:namespace-alias>` element is used to replace a namespace in the style sheet to a different namespace in the output.
  - It must be a child node of `<xsl:stylesheet>` or `<xsl:transform>`.

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:wxsl="http://www.w3schools.com/w3style.xsl">

<xsl:namespace-alias stylesheet-prefix="wxsl" result-prefix="xsl"/>

<xsl:template match="/">
  <wxsl:stylesheet>
    <xsl:apply-templates/>
  </wxsl:stylesheet>
</xsl:template>

</xsl:stylesheet>
```

# XSLT-based Schematron Implementation



- Core element

Look for assert elements in the source tree

```
<xsl:template match="sch:assert | assert">
   <xsl:if test="not(@test)">
      <xsl:message>Markup Error: no test attribute in <assert></xsl:message>
   </xsl:if>
   <axsl:choose>
      <axsl:when test="{@test}"/>
      <axsl:otherwise>
         <xsl:call-template name="process-assert">
            <xsl:with-param name="role" select="@role"/>
            <xsl:with-param name="id" select="@id"/>
            <xsl:with-param name="test" select="normalize-space(@test)" />
            <xsl:with-param name="icon" select="@icon"/>
            <xsl:with-param name="subject" select="@subject"/>
            <xsl:with-param name="diagnostics" select="@diagnostics"/>
         </xsl:call-template>
      </axsl:otherwise>
   </axsl:choose>
</xsl:template>
```

`axsl:choose` element goes into the result tree as `xsl:choose` element.

Do nothing.

Specific template to handle assert elements.