# Relax NG schema for XSL-FO

Alexander Peshkov
David Tolpin

RenderX

XMLEurope 2004
18 Apr - 21 Apr 2004
Amsterdam RAI Center, Netherlands

# Typical XSL-FO workflow

# Use of formal schemas

| Specification | Normative Schema | Unofficial Schemas |
|---|---|---|
| **XHTML 1.0** | DTD | Relax NG, W3C XML Schema |
| **Docbook 4.3** | DTD | Relax NG, W3C XML Schema |
| **SVG 1.1** | DTD | Relax NG, W3C XML Schema<br>Both schemas planned to become normative in 1.2 |
| **MathML 2.0** | DTD, W3C XML Schema | Relax NG |

# Formal schemas and validation tools

1. DTD;
2. XSLT (or Schematron);
3. Relax NG (or W3C XML Schema).

# Side-by-side comparison

| | DTD | XSLT | Relax NG |
|---|---|---|---|
| **Advantages** | Simple<br>Widespread<br>Declarative<br>No special tools | Powerful<br>Widespread<br>Namespaces<br>Strong error-reporting | Simple<br>Fast<br>Powerful<br>Declarative<br>Namespaces |
| **Drawbacks** | Poor<br>Slow<br>Bound with document<br>Namespace agnostic | Memory-hungry<br>Non-declarative | Weak error-reporting |
| **Tools (Java)** | Crimson, Xerces | XT, Saxon, Xalan | Jing, MSV |
| **RenderX XEP** | version 2.x | version 3.x | version 4.x ??? |

# DTD pros and cons

| Pros | Cons | Tools (Java) | RenderX XEP |
|---|---|---|---|
| • Simple;<br>• Widespread;<br>• No special tools required;<br>• Declarative. | • Semantically poor;<br>• Requires explicit document binding;<br>• No support for namespaces; | • Crimson<br>• Xerces | • version 2.x |

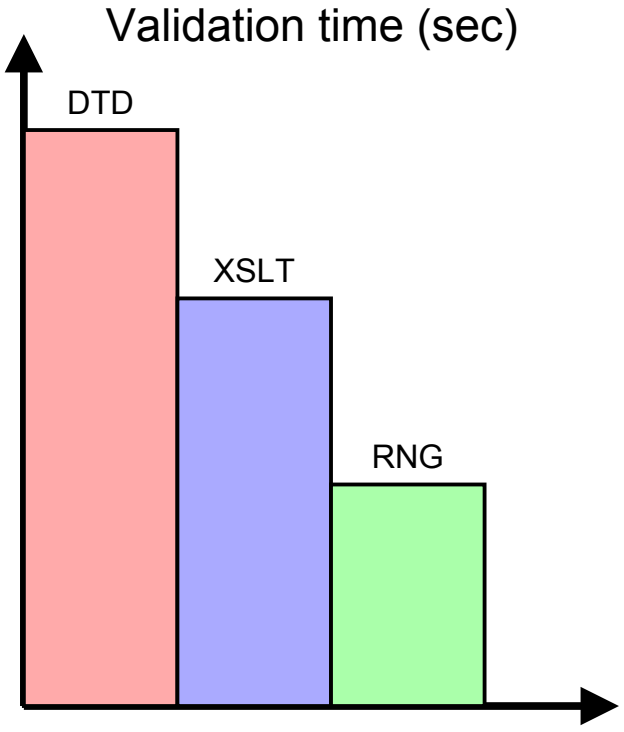
# XSLT pros and cons

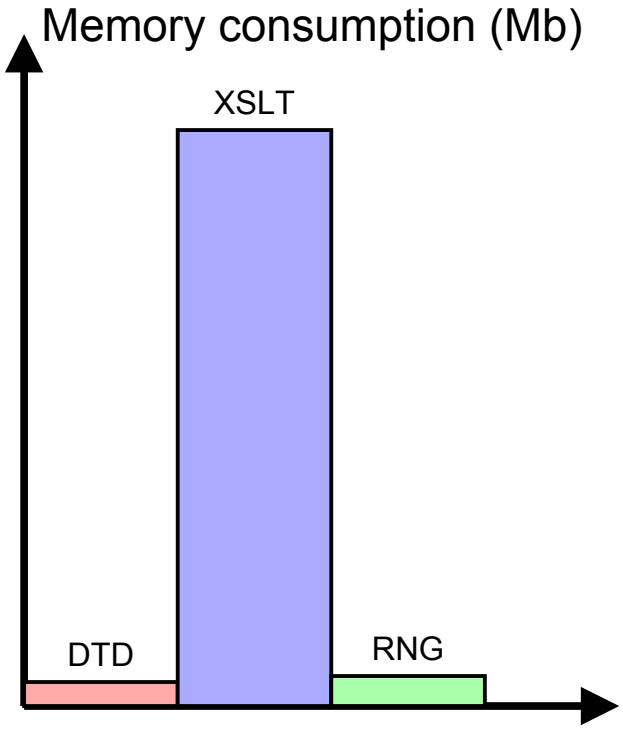
| Pros | Cons | Tools (Java) | RenderX XEP |
|---|---|---|---|
| • Powerful;<br>• Widespread;<br>• Namespaces supported;<br>• Strong error-reporting. | • Memory-hungry;<br>• Non-declarative. | • XT<br>• Saxon<br>• Xalan | • version 3.x |

# Relax NG pros and cons

| Pros | Cons | Tools (Java) | RenderX XEP |
|---|---|---|---|
| • Simple;<br>• Powerful;<br>• Efficient;<br>• Namespaces supported;<br>• Declarative. | • Weak error-reporting. | • Jing<br>• MSV | • version 4.x ? |

# Validation statistics



# Validation aspects

1. Content models for elements;
2. Constraints on attribute occurrence;
3. Constraints on attribute values.

# XSL-FO peculiarities

Content models of some elements depend on values of their attributes

**<fo:block-container absolute-position="auto">**

XSL FO is a real challenge for schema writers, because all three aspects are interdependent: content models for some elements depend on values of attributes. Those elements are fo:block-container and fo:float.

Normally positioned container. It does not use coordinates nor explicit height/width. It can contain outlines *

Behavior of the first one is strikingly different depending on the value of its "absolute-position" property. Absolutely positioned block-containers (with absolute-position="absolute" or absolute-position="fixed") resemble outline objects (floats): they are not allowed in the same contexts where outlines are not allowed, and cannot have outlines as descendants. Moreover, absolute and normal block-containers have quite different attribute sets. There is a clear need for separate element. In order to enforce XSL FO requirements we were forced to create a special pseudo element in our schema - "absolute-container".

** <fo:footnote> is one of the outline elements.*

**<fo:block-container absolute-position="absolute">**

XSL FO is a real challenge for schema writers, because all three aspects are interdependent: content models for some elements depend on values of attributes. Those elements are fo:block-container and fo:float.

Behavior of the first one is strikingly different depending on the value of its "absolute-position" prope tainers (with absolute-position="absolute" objects (floats): they are not allowed in the s ed, and cannot have outlines as descendar containers have quite different attribute sets. T order to enforce XSL FO requirements we we t in our schema - "absolute-container".

Absolutelly positioned container. It has fixed coordinates of the left top angle and a fixed size:
`left="80pt"`
`top="150pt"`
`height="250pt"`
`width="230pt"`
It cannot contain outlines.

A similar problem arises "float" property value, the element may assu nd before-float), having quite different sets of

Note that both problem solute-position" properties were directly por to the different elements. On the contrary, in XSL FO those properties are applied to the dedicated elements only, and their semantics is overloaded (e.g. fo:float float="none").

# XSL-FO peculiarities

Content models of some elements depend on values of their attributes

| <fo:float float="before"> | <fo:float float="start"> |
|---|---|
| Before float with float="before". It is drawn in a special page region.<br><br>XSL FO is a real challenge for schema writers, because all three aspects are interdependent: content models for some elements depend on values of attributes. Those elements are fo:block-container and fo:float.<br><br>Behavior of the first one is strikingly different depending on the value of its "absolute-position" property. Absolutely positioned block-containers (with absolute-position="absolute" or absolute-position="fixed") resemble outline objects (floats): they are not allowed in the same contexts where outlines are not allowed, and cannot have outlines as descendants. Moreover, absolute and normal block-containers have quite different attribute sets. There is a clear need for separate element. In order to enforce XSL FO requirements we were forced to create a special pseudo element in our schema - "absolute-container".<br><br>A similar problem arises with fo:float: depending on the value of the "float" property value, the element may assume two very different semantics (side-float and before-float), having quite different sets of constraints.<br><br>Note that both problems have the same origin - "float" and "absolute-position" properties were directly ported from CSS where they can be applied to the different | XSL FO is a real challenge for schema writers, because all three aspects are interdependent: content models for some elements depend on values of attributes. Those elements are fo:block-container and fo:float.<br><br>Side float with float="start".<br><br>Behavior of the first one is strikingly different depending on the value of its "absolute-position" property. Absolutely positioned block-containers (with absolute-position="absolute" or absolute-position="fixed") resemble outline objects (floats): they are not allowed in the same contexts where outlines are not allowed, and cannot have outlines as descendants. Moreover, absolute and normal block-containers have quite different attribute sets. There is a clear need for separate element. In order to enforce XSL FO requirements we were forced to create a special pseudo element in our schema - "absolute-container".<br><br>A similar problem arises with fo:float: depending on the value of the "float" property value, the element may assume two very different semantics (side-float and before-float), having quite different sets of constraints.<br><br>Note that both problems have the same origin - "float" and "absolute-position" properties were directly ported from CSS where they can be applied to the different elements. On the contrary, in XSL-FO those properties are applied to the dedicated elements only, and their semantics is overloaded (e.g. fo:float float="none"). |

# XSL-FO peculiarities

Content models of some elements are context sensitive

```
<fo:block>
  <fo:wrapper color="red">
    <fo:block>
     <fo:leader>#</fo:leader>
    </fo:block>
    <fo:inline>
      Text inside fo:inline.
    </fo:inline>
  </fo:wrapper>
  ...
</fo:block>
```

```
<fo:block-container>
  <fo:wrapper color="red">
    <fo:block>
     <fo:leader>#</fo:leader>
    </fo:block>
    <fo:inline>
      Text inside fo:inline.
    </fo:inline>
  </fo:wrapper>
  ...
</fo:block-container>
```

# XSL-FO peculiarities

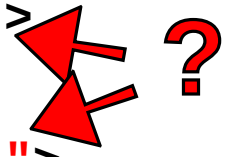
Attributes placement ambiguous

```
<fo:inline table-layout="fixed"
           orphans="4">
  Block with table inside inline.
  <fo:block>
    <fo:table
        table-layout="from-nearest-specified-value()">
      <fo:table-body>
    ...
  </fo:block>
</fo:inline>
```

# XSL-FO peculiarities

Attributes values uncertain

```
<fo:block font-size="abs(1em div 2 - 22pt)">
  Text inside fo:block.
  <fo:inline font-size="from-parent() + 4pt">
    Text inside inline.
    ...
  </fo:inline>
</fo:block>
```



# Error reporting - Relax NG vs. XSLT

```
...
 <fo:layout-master-set>
  <fo:simple-page-master master-name="main" margin="0in">
   <fo:region-before extent="0.5in"/>
   <fo:region-body margin="0.5in"/>
  </fo:simple-page-master>
 </fo:layout-master-set>
 <fo:page-sequence master-reference="main">
  <fo:flow flow-name="xsl-region-body">
   <fo:inline>
       This document has minor and serious errors.
   </fo:inline>
  </fo:flow>
 ...
```

# Error reporting - Relax NG vs. XSLT

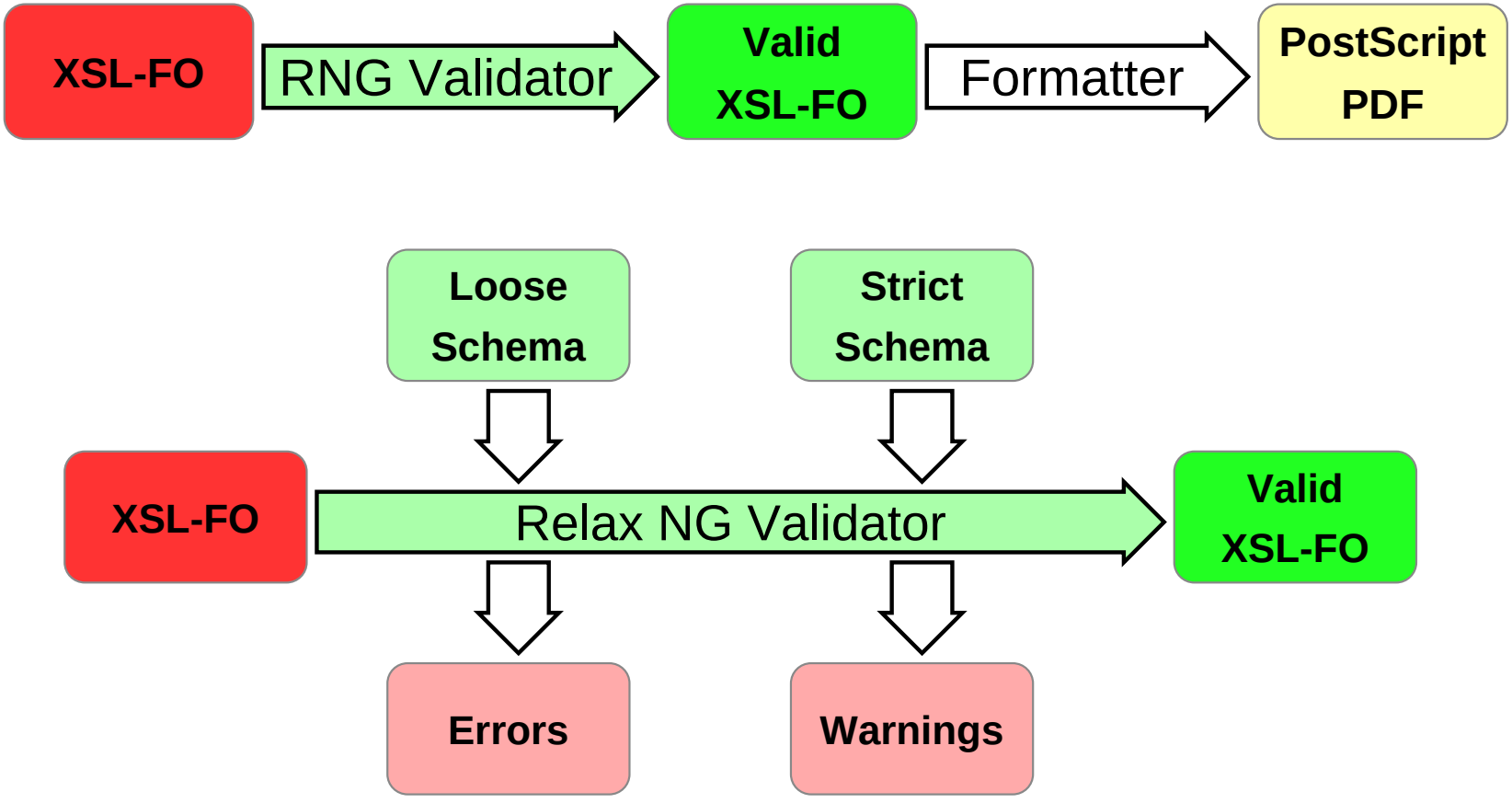
Jing validation log

```
severity.fo:5:38: error: required elements missing
severity.fo:6:36: error: element "region-body" from namespace "..."
                         not allowed in this context
severity.fo:11:15: error: element "inline" from namespace "..."
                          not allowed in this context
severity.fo:14:13: error: unfinished element
```

XSLT validation log

```
[warning] Incorrect order of region descriptors inside
          'fo:simple-page-master'.  Regions shall be ordered
          according to the following content model: fo:region-body,
          fo:region-before?, fo:region-after?, fo:region-start?,
          fo:region-end?
[error] Element 'fo:inline' cannot be a child of 'fo:flow'.
        Only block-level elements are permitted in this context.
```

# Error-reporting with multiple validations



# Relax NG tools

| Validators | Converters | XML editors | Other |
|---|---|---|---|
| • Jing<br>• MSV<br>• RNV<br>• Libxml2; | • Trang<br>• SUN RNG Converter | • Topologi Collaborative Markup Editor<br>• Oxygen from SyncRO Soft<br>• EMACS with Relax NG mode | • Relaxer<br>• RelaxNGCC<br>• Sun XML Instance Generator |

... and there is more...

# So why Relax NG schema for XSL-FO?

1. It is simple, powerful, efficient, namespace-aware, etc.
2. **It can be a common language for XSL-FO authors.**

# XSL-FO schemas and validation tools

- XSL-FO DTD;
- XSL-FO XSLT validator;
- Relax NG schema for XSL-FO.

**are publicly available at**

http://xep.xattic.com/