

CESNET technical report number 3/2006

Annotating XML Schemas with reStructuredText

Ladislav Lhotka

19.6.2006

Keywords: XML, RELAX NG, reStructuredText, XSLT

1 Abstract

This technical report describes a method for annotating XML schemas expressed in the RELAX NG language. The annotations use a natural text markup of reStructuredText (reST). The annotated schema is a valid RELAX NG XML document that can be transformed into a reST document via an XSLT stylesheet. Consequently, the schema can be easily presented as HTML or \LaTeX with added value of bidirectional hyperlinks between RELAX NG definitions and their references.

2 Introduction

Extensible Markup Language (XML) is increasingly used as a flexible format for representing various structured data in networking and other software applications, even if the data is intended to be parsed by machines. The concrete data model can be defined by means of a special language known as *XML schema*. The base XML 1.0 specification [XML] offers the DTD (Document Type Definition) language for this purpose. However, this language has a number of deficiencies:

- DTD itself is not an XML document
- DTD does not allow to use elements with identical names in different contexts
- Namespaces are not supported

The W3C consortium thus prepared a much more sophisticated language with a rather unfortunate name – *XML Schema* [XSch1], [XSch2] – as if it was supposed

to be *the* ultimate XML schema. As it turns out, it is a typical consortium-driven specification: very complex and in certain places even unclear and ambiguous. An interesting alternative with approximately the same expressive power is *RELAX NG*[RNG]. It is based on a sound mathematical basis of the tree automaton theory and in general is easier to use than W3C XML Schema.

Any XML schema is primarily used for validating XML documents: a validating XML parser is able to verify whether an XML document conforms to the given schema. However, an XML schema can also serve as *authoritative documentation* of the data model. This documentation role of an XML schema can be further improved by interspersing the schema with annotations. This can be realised either via XML comments analogical to comments in the source code of programs, or by using extra markup, for example special XML elements. RELAX NG specification [RNG] requires the parsers to ignore elements in foreign namespaces that are included in the schema document, so annotations can be quite naturally enclosed in elements whose namespace is different from that of RELAX NG schema itself (<http://relaxng.org/ns/structure/1.0>).

This technical report presents a method for annotating RELAX NG schemas and an XSLT stylesheet that allows to convert a RELAX NG schema augmented with annotations to common presentation forms such as HTML or \LaTeX . The annotations use the simple and effective markup of reStructuredText¹ (reST). The XSLT stylesheet mentioned above is then able to transform the annotated RELAX NG schema into a valid reST document that essentially combines RELAX NG mechanisms (pattern definitions and references) with reST features such as hyperlinks. The result closely resembles *literate programs*[Knu92].

This work was inspired by a similar XSLT stylesheet written by Zdeněk Wagner [Wag05]. However, the present approach is slightly more general, thanks to the flexibility of reST as opposed to HTML, which is the result of Wagner's stylesheet. Also, my stylesheet uses only XSLT 1.0 and can thus be used with virtually any XSLT processor.

The listing of the entire stylesheet can be found in the appendix 6 . It is also available online².

3 Adding annotations

The annotation system introduces just a single new XML element: `rest`. Its namespace is <http://www.cesnet.cz/ns/rngrest-annotations/1.0>. In other words, annotations are usually included as follows:

¹<http://docutils.sourceforge.net/rst.html>

²<http://staff.cesnet.cz/lhotka/rngrest.tar.gz>

```
<a:rest> ... </a:rest>
```

where the namespace prefix `a` must be properly defined as an abbreviation of the above URL, typically in the grammar element.

Any number of `rest` elements can be created as direct children of the following RELAX NG elements: `grammar`, `start` and `define`. As a matter of fact, they may appear in other places as well but will be ignored there by the XSLT stylesheet.

Any valid `reStructuredText` is allowed inside the `rest` element – multiple paragraphs with tables, figures, hyperlinks etc. Sections should be used with care. If used, the section titles must not be underlined with exclamation marks ! as these are used internally for automatic sections that the stylesheet creates for each `define` element in the schema.

The following example is a simple annotated RELAX NG schema:

```
<?xml version="1.0"?>

<grammar ns="http://www.cesnet.org/ns/football/1.0"
  xmlns="http://relaxng.org/ns/structure/1.0"
  xmlns:a="http://www.cesnet.cz/ns/rngrest-annotations/1.0">
```

```
  <a:rest>
```

```
    Example: Annotated RELAX NG Schema
```

```
    =====
```

```
    :Author: Ladislav Lhotka
    :Contact: lhotka@cesnet.cz
```

```
    .. contents::
```

This is an example of a RELAX NG schema annotated with `reStructuredText`. Every annotation consists of one or more paragraphs and may contain text in *italics*, **boldface** or `monospace font`, numbered or bulleted lists, hyperlinks etc.

We use a simple data model of a `football_ team`.

```
    .. _football: http://en.wikipedia.org/wiki/Football
```

```
  </a:rest>
```

```
  <start>
```

```
<a:rest>
```

The root element is 'football-team'. It contains any number of players.

```
</a:rest>
```

```
<element name="football-team">
```

```
<zeroOrMore>
```

```
<element name="player">
```

```
<ref name="player-content"/>
```

```
</element>
```

```
</zeroOrMore>
```

```
</element>
```

```
</start>
```

```
<define name="player-content">
```

```
<a:rest>
```

Each player has the attribute 'role' with one of the four choices below, and element 'name' which is supposed to contain full name of the player.

```
</a:rest>
```

```
<attribute name="role">
```

```
<choice>
```

```
<value>goalkeeper</value>
```

```
<value>defender</value>
```

```
<value>midfielder</value>
```

```
<value>striker</value>
```

```
</choice>
```

```
</attribute>
```

```
<element name="name">
```

```
<text/>
```

```
</element>
```

```
</define>
```

```
</grammar>
```

Several things are worth pointing out here:

- The annotations must be indented strictly according to the rules of reST so that, for example, normal paragraph text starts at column 1. This slightly breaks the canonical indentation structure of XML documents, but there is no easy way around this problem, as reST is very fussy about indentation.
- One has to be careful when including characters that are not allowed in XML documents: <, > or &. XML entity references such as < ; do not

work here because the reST parser does not interpret them. The solution is to use the unicode directive of reST, for example

```
.. |lt| unicode:: U+003C
```

The < character can then be represented as |lt| in annotations.

- A special case of the previous issue are XML element names inside annotations. Should the author wish to write them with the < and > delimiters – and I generally recommend not to – then he or she might consider creating a new *interpreted text role*, see [Goo05]. An XML element could then be conveniently written as, for example, :xml: ‘football-team’.
- The .. contents:: directive offers an easy way for generating an index of all RELAX NG pattern definitions used in the schema.

4 Transformations

An annotated RELAX NG schema can be converted to reStructuredText by means of an XSLT processor and the XSLT stylesheet `rngrest.xsl` shown in the appendix 6. The following command uses the `xsltproc`³ processor:

```
$ xsltproc --output example.rest rngrest.xsl example.rng
```

When applied to the example schema shown above, this command gives the following reST file as output:

Example: Annotated RELAX NG Schema

=====

```
:Author: Ladislav Lhotka
:Contact: lhotka@cesnet.cz
```

```
.. contents::
```

This is an example of a RELAX NG schema annotated with reStructuredText. Every annotation consists of one or more paragraphs and may contain text in *italics*, **boldface** or ‘‘monospace font’’, numbered or bulleted lists, hyperlinks etc.

³<http://xmlsoft.org/XSLT/xsltproc2.html>

We use a simple data model of a football_ team.

```
.. _football: http://en.wikipedia.org/wiki/Football
```

```
::
```

```
<grammar
  ns="http://www.cesnet.org/ns/football/1.0"
  xmlns:a="http://www.cesnet.cz/ns/rngrest-annotations/1.0"
  xmlns="http://relaxng.org/ns/structure/1.0">
```

```
start
```

```
!!!!
```

The root element is 'football-team'. It contains any number of players.

```
.. parsed-literal::
```

```
<start>
  <element name="football-team">
    <zeroOrMore>
      <element name="player">
        <ref name="player-content_"/>
      </element>
    </zeroOrMore>
  </element>
</start>
```

```
player-content
```

```
!!!!!!!!!!!!!!!!!!!!
```

Each player has the attribute 'role' with one of the four choices below, and element 'name' which is supposed to contain full name of the player.

The pattern is referenced by:

```
* start_
```

```
.. parsed-literal::
```

```
<define name="player-content">
```

```

<attribute name="role">
  <choice>
    <value>goalkeeper</value>
    <value>defender</value>
    <value>midfielder</value>
    <value>striker</value>
  </choice>
</attribute>
<element name="name">
  <text/>
</element>
</define>

```

Comparing it to the original annotated schema, we see that the annotation of the grammar element became the introductory part of the reST file. The annotations of the other elements – start and define – were moved before their parent elements and also received automatically generated titles. These titles serve as targets for hyperlinks from the corresponding RELAX NG ref elements as well as from the lists of referring definitions that are also automatically created (note the underline characters at the end of the referenced element names).

The reST file is then converted to the desired presentation format (HTML, \LaTeX or XML) by the standard docutils⁴ tools. For example, to convert our example reST file to HTML, use the following command:

```
$ rest2html example.rest example.html
```

Part of the result rendered by Firefox is shown in Figure 1.

5 Conclusions

This technical report describes a simple yet flexible way of annotating RELAX NG schemas. The annotations use the plain text markup of reStructuredText⁵, which means, in the first place, that the annotated schema is easily readable even in the source form. In addition, XSLT stylesheet `rngrest.xsl` in appendix 6 (also available online⁶) transforms the annotated schema into a valid reST document that can be in turn converted to HTML, \LaTeX or XML with the additional benefits of bidirectional links between RELAX NG pattern definitions and their references and, optionally, a listing of all definitions.

⁴<http://docutils.sourceforge.net/>

⁵<http://docutils.sourceforge.net/rst.html>

⁶<http://staff.cesnet.cz/lhotka/rngrest.tar.gz>



Figure 1: Our example annotated RELAX NG schema rendered as HTML.

A more complex schema annotated using the method described in this report is the data model for FlowMon probe configuration [Lho06].

6 Appendix. XSLT Stylesheet rngrest.xsl

```
<?xml version="1.0"?>
```

```
<!--
```

```
Program name: rngrest.xsl
```

```
Description: This style sheet converts annotated RELAX NG schemas  
              into reStructuredText documents
```

```
Author: Ladislav Lhotka <Lhotka@cesnet.cz>
```

```
Copyright (C) 2006 CESNET
```

```
This program is free software; you can redistribute it and/or  
modify it under the terms of version 2 of the GNU General Public  
License as published by the Free Software Foundation.
```

```
This program is distributed in the hope that it will be useful, but  
WITHOUT ANY WARRANTY; without even the implied warranty of  
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU  
General Public License for more details.
```

```
You should have received a copy of the GNU General Public License  
along with this program; if not, write to the Free Software  
Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  
02111-1307, USA.
```

```
$Id: rngrest.xsl,v 1.4 2006/05/05 17:59:44 lhotka Exp $  
-->
```

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"  
                xmlns:rng="http://relaxng.org/ns/structure/1.0"  
                xmlns:a="http://www.cesnet.cz/ns/rngrest-annotations/1.0"  
                version="1.0">
```

```
<xsl:output method="xml" indent="yes" omit-xml-declaration="yes"/>  
<xsl:strip-space elements="rng:start rng:define"/>
```

```
<xsl:variable name="NL">
```

```

    <xsl:text>
</xsl:text>
</xsl:variable>

<xsl:variable name="NLI">
    <xsl:text>
    </xsl:text>
</xsl:variable>

<xsl:variable name="NLNL">
    <xsl:value-of select="$NL"/>
    <xsl:value-of select="$NL"/>
</xsl:variable>

<xsl:variable name="IND">
    <xsl:text>    </xsl:text>
</xsl:variable>

<xsl:template name="underline">
    <xsl:param name="str"/>
    <xsl:text>!</xsl:text>
    <xsl:if test="string-length($str)>1">
        <xsl:call-template name="underline">
            <xsl:with-param name="str">
                <xsl:value-of select="substring($str,2)"/>
            </xsl:with-param>
        </xsl:call-template>
    </xsl:if>
</xsl:template>

<!-- The root element -->

<xsl:template match="rng:grammar">
    <xsl:apply-templates select="a:rest"/>
    <xsl:value-of select="$NLNL"/>
    <xsl:text>:</xsl:text>
    <xsl:value-of select="$NLNL"/>
    <xsl:text disable-output-escaping="yes">    &lt;grammar</xsl:text>
    <xsl:for-each select="@*">
        <xsl:value-of select="$NLI"/>
        <xsl:text>    </xsl:text>
        <xsl:value-of select="name()"/>
        <xsl:text>="</xsl:text>

```

```

        <xsl:value-of select="."/>
        <xsl:text>"</xsl:text>
    </xsl:for-each>
    <xsl:for-each select="namespace:*">
        <xsl:if test="name()!='xml'">
            <xsl:value-of select="$NLI"/>
            <xsl:text>    xmlns</xsl:text>
            <xsl:if test="name()!=''">
                <xsl:text>:</xsl:text>
                <xsl:value-of select="name()"/>
            </xsl:if>
            <xsl:text>="</xsl:text>
            <xsl:value-of select="."/>
            <xsl:text>"</xsl:text>
        </xsl:if>
    </xsl:for-each>
    <xsl:text disable-output-escaping="yes">&gt;</xsl:text>
    <xsl:value-of select="$NLNL"/>
    <xsl:apply-templates select="rng:*" mode="listing"/>
</xsl:template>

<xsl:template match="rng:start" mode="listing">
    <xsl:value-of select="$NL"/>
    <xsl:text>start</xsl:text>
    <xsl:value-of select="$NL"/>
    <xsl:text>!!!!</xsl:text>
    <xsl:value-of select="$NLNL"/>
    <xsl:apply-templates select="a:rest"/>
    <xsl:value-of select="$NLNL"/>
    <xsl:text>.. parsed-literal::</xsl:text>
    <xsl:value-of select="$NLNL"/>
    <xsl:text disable-output-escaping="yes">    &lt;</xsl:text>
    <xsl:value-of select="name()"/>
    <xsl:apply-templates select="@*"/>
    <xsl:text disable-output-escaping="yes">&gt;</xsl:text>
    <xsl:apply-templates mode="listing"/>
    <xsl:value-of select="$NLI"/>
    <xsl:text disable-output-escaping="yes">&lt;</xsl:text>
    <xsl:value-of select="name()"/>
    <xsl:text disable-output-escaping="yes">&gt;</xsl:text>
    <xsl:value-of select="$NL"/>
</xsl:template>

```

```

<xsl:template match="rng:define" mode="listing">
  <xsl:value-of select="$NL"/>
  <xsl:variable name="elname" select="@name"/>
  <xsl:value-of select="$elname"/>
  <xsl:value-of select="$NL"/>
  <xsl:call-template name="underline">
    <xsl:with-param name="str" select="$elname"/>
  </xsl:call-template>
  <xsl:value-of select="$NLNL"/>
  <xsl:apply-templates select="a:rest"/>
  <xsl:value-of select="$NLNL"/>
  <xsl:variable name="refs"
    select="//rng:define[descendant::rng:ref[@name=$elname]]"/>
  <xsl:if test="count($refs)!=0 or //rng:start//rng:ref[@name=$elname]">
    <xsl:text>

```

The pattern is referenced by:

```

    </xsl:text>
    <xsl:value-of select="$NL"/>
    <xsl:if test="//rng:start//rng:ref[@name=$elname]">
      <xsl:text>* start_</xsl:text>
      <xsl:value-of select="$NLNL"/>
    </xsl:if>
    <xsl:for-each select="$refs">
      <xsl:text>* </xsl:text>
      <xsl:value-of select="@name"/>
      <xsl:text>_</xsl:text>
      <xsl:value-of select="$NLNL"/>
    </xsl:for-each>
    <xsl:value-of select="$NL"/>
  </xsl:if>
  <xsl:text>.. parsed-literal::</xsl:text>
  <xsl:value-of select="$NLNL"/>
  <xsl:text disable-output-escaping="yes"> &lt;</xsl:text>
  <xsl:value-of select="name()"/>
  <xsl:apply-templates select="@*"/>
  <xsl:text disable-output-escaping="yes">&gt;</xsl:text>
  <xsl:apply-templates mode="listing"/>
  <xsl:value-of select="$NLI"/>
  <xsl:text disable-output-escaping="yes">&lt;</xsl:text>
  <xsl:value-of select="name()"/>
  <xsl:text disable-output-escaping="yes">&gt;</xsl:text>
  <xsl:value-of select="$NL"/>
</xsl:template>

```

```

<xsl:template match="rng:*" mode="listing">
  <xsl:if test="name(..)='define' or name(..)='start'">
    <xsl:value-of select="$NLI"/>
  </xsl:if>
  <xsl:text disable-output-escaping="yes"> &lt;</xsl:text>
  <xsl:value-of select="name()"/>
  <xsl:apply-templates select="@*" />
  <xsl:choose>
    <xsl:when test="count(*)=0">
  <xsl:choose>
    <xsl:when test="count(text())=0">
      <xsl:text></xsl:text>
    </xsl:when>
    <xsl:otherwise>
      <xsl:text disable-output-escaping="yes">&gt;</xsl:text>
      <xsl:apply-templates />
      <xsl:text disable-output-escaping="yes">&lt;</xsl:text>
      <xsl:value-of select="name()"/>
    </xsl:otherwise>
  </xsl:choose>
    </xsl:when>
    <xsl:otherwise>
      <xsl:text disable-output-escaping="yes">&gt;</xsl:text>
      <xsl:apply-templates mode="listing" />
      <xsl:text disable-output-escaping="yes"> &lt;</xsl:text>
      <xsl:value-of select="name()"/>
    </xsl:otherwise>
  </xsl:choose>
  <xsl:text disable-output-escaping="yes">&gt;</xsl:text>
</xsl:template>

<xsl:template match="a:rest">
  <xsl:value-of select="."/>
</xsl:template>

<xsl:template match="a:rest" mode="listing" />

<xsl:template match="@*">
  <xsl:text> </xsl:text>
  <xsl:value-of select="name()"/>
  <xsl:text>="</xsl:text>
  <xsl:value-of select="."/>

```

```

    <xsl:text>”</xsl:text>
</xsl:template>

<xsl:template match="@*”>
  <xsl:text> </xsl:text>
  <xsl:value-of select="name()"”/>
  <xsl:text>=”</xsl:text>
  <xsl:value-of select=".””/>
  <xsl:if test="name(..)='ref' and name()='name'”>
    <xsl:text>_</xsl:text>
  </xsl:if>
  <xsl:text>”</xsl:text>
</xsl:template>

</xsl:stylesheet>

```

References

- [Goo05] Goodger, D. *Creating reStructuredText Interpreted Text Roles*. Developer documentation of reST, 2005. Available online⁷.
- [Knu92] Knuth D.E. *Literate Programming*. Stanford, California: Center for the Study of Language and Information, 1992. 368pp. ISBN 0-937073-80-6.
- [Lho06] Lhotka L. *XML Schema of FlowMon Configuration Data*. Available online⁸.
- [RNG] Clark J. and Murata M. (Editors). *RELAX NG Specification*. OASIS Consortium, 2001. Available online⁹.
- [Wag05] Wagner Z. *Tool for Annotation of Relax NG Schemas*. IceBearSoft, 2005. Available online¹⁰.
- [XML] Bray T., Paoli J., Sperberg-McQueen C.M., Maler E. and Yergeau, F. (Editors). *Extensible Markup Language (XML) 1.0*. Third edition. W3C Consortium, 2004. Available online <<http://www.w3.org/TR/2004/REC-xml-20040204/>>.

⁷<http://docutils.sourceforge.net/docs/howto/rst-roles.html>

⁸<http://www.flowmon.org/flowmon-probe/devel/config/flowmon-rng.rest/>

⁹<http://www.oasis-open.org/committees/relax-ng/spec-20011203.html>

¹⁰<http://icebearsoft.euweb.cz/xmltools/rngdoc-en.php>

- [XSch1] Thompson H.S., Beech D., Maloney M. and Mendelsohn N. (Editors). *XML Schema Part 1: Structures*. Second edition. W3C Consortium, 2004. Available online¹¹.
- [XSch2] Biron P.V. and Malhotra A. (Editors). *XML Schema Part 2: Datatypes*. Second edition. W3C Consortium, 2004. Available online¹².

¹¹<http://www.w3.org/TR/xmlschema-1/>

¹²<http://www.w3.org/TR/xmlschema-2/>