16.1a A RELAX NG Way to Live: The Story of Updating the Live Access Server Configuration XML

Roland H. Schweitzer*, A. Manke[1], K. O'Brien[1], J. Mclean[1], J. Callahan[1], S. Hankin[2]

Weathertop Consulting, LLC*
College Station, TX

JISAO/University of Washington, Seattle, WA[1]

NOAA Pacific Marine Environmental Laboratory, NOAA/OAR/PMEL, Seattle, WA[2]

1. INTRODUCTION

The Live Access Server is a general purpose Web-server for geo-science data sets. Data providers can use the three tiered LAS architecture (see Figure 1) to build custom Web interfaces to their scientific data. Users can then access the LAS site to search the provider's on-line data holdings, make plots of data, create sub-sets in a variety of formats, compare data sets and perform analysis on the data.

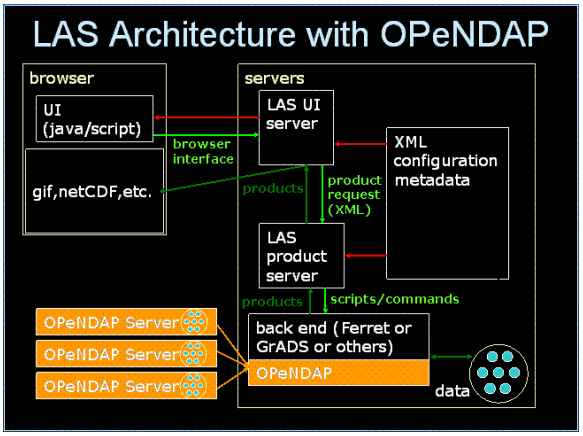

Figure 1. The LAS Architecture

The Extensible Markup Language (XML) has been a part of the LAS design and implementation for much of the history of the product. Early on LAS was able to take advantage of the flexibility and ease of implementation that comes with using XML and to use some the myriad of tools available for XML manipulation. That said, just as LAS has not stood still with many new and innovative features being added with each release, XML has advanced as well. Part of the recent advances in XML have to do with schema languages used to define the contents of XML documents. The LAS team has decided that it is important to take advantage of these new tools and to undertake a redesign of the XML used to configure LAS. This paper will explore the process of redesigning the LAS XML. It will offer many insights in to the process of choosing an XML schema language and designing a schema that is both easy to use and flexible enough to handle the complex needs of LAS configuration.

*Corresponding author address: Roland H. Schweitzer, Weathertop Consulting, LLC, 2802 Cimarron Ct., College Station, TX 77845. E-mail: Roland.Schweitzer@noaa.gov

2. XML Schema Languages

Part of the power of using XML for configuration information comes from the fact that software exists in many languages that can parse XML documents. These parsers can enforce two characteristics of the document: that they are well-formed and that they are valid.

A well-formed document is simply a document that has the proper nesting of elements and other details of the basic XML syntax. A valid document is one that conforms to a structure that is specified either by a DTD or a schema. For the purposes of our XML redesign project we did not consider using a DTD although the schema language we chose can be easily translated into a DTD specification by freely available software. Before talking about the specifics of our project we will continue with a brief discussion of schema languages.

2.1 W3C XML Schema

The W3C XML Schema specification is what most people think of when then encounter the term XML schema. In fact this schema formalization is a popular method for specifying the structure of XML documents. Additionally, many popular open source and commercial XML authoring software programs can be used to create W3C schemas.

Even so, many in the XML development community have criticized XML Schema. The specification is difficult to read and understand and often uses constructs which are not intuitive to even computer scientists. Furthermore, the XML Schema is missing some important features and has a poor abstraction. The debate about the merits and cons of XML Schema is certainly not over, but these issues were enough for us to look at other schema implementations.

2.2 RELAX NG

RELAX NG purports to be a simple, easy to use schema language. It is based entirely on XML syntax and has a solid theoretical basis. RELAX NG also addresses some of the noted shortcoming of XML Schema. RELAX NG has the ability to specify unordered and mixed content and to treats attributes and elements uniformly. XML Schema does not.

2.3 Our Implementation

We decided to implement the schema for our redesigned XML using RELAX NG. One fundamental concept our XML must implement is the ability to uniquely identify elements within the document and to refer to those elements again in other places in the document. Elements can be uniquely identified using the XML Schema Datatype library (also supported by RELAX NG). Additionally, creating elements that were either the originally specification of the element including the specifying the unique ID, or creating an element that referred to an existing element using the IDREF data type was easy to implement using RELAX NG, but was impossible using XML Schema.

2.4 A Few Brief Examples

So, for example a collection of LAS grid and axis element can be defined as follows:

```
<grid ID="myGrid">
 <xAxis ID="x1" units="degrees_east">
   <arange size="180" start="1" step="2"/>
 </xAxis>
 <yAxis ID="y1" units="degrees_north">
   <arange size="90" start="-89" step="1"/>
 </yAxis>
 <tAxis ID="t1" units="month">
   <arange size="12" start="1-1-1" step="1"/>
 </tAxis>
</grid>
<grid ID="otherGrid">
 <xAxis IDREF="x1"/>
 <yAxis IDREF="y1"/>
 <tAxis ID="t2" units="month">
  <arange size="5" start="1990-01-01"
          step="1"/>
 </tAxis>
</grid>
```

In the second grid element, the x-axis and y-axis definition are recycled by referring to the previously defined <xAxis> and <yAxis> elements, while an entirely new t-axis is defined within the second grid element.

The ability to define an element which allows only one attribute (in this case the IDREF attribute) or all of the other attributes and sub-elements needed to define the element (the ID and units attributes) is only possible using the RELAX NG.

To complete the discussion, the RELAX NG schema definition for the <xAxis> element is shown below.

```
<define name="XAXIS">
    <element name="xAxis">
       <choice>
          <attribute name="IDREF">
             <data type="IDREF"/>
          </attribute>
          <group>
             <attribute name="ID">
                <data type="ID"/>
             </attribute>
             <attribute name="units"/>
             <choice>
                <oneOrMore>
                   <ref name="V"/>
                </oneOrMore>
                <ref name="RANGE"/>
             </choice>
          </group>
       </choice>
    </element>
 </define>
```

Note that the first block in the definition is a RELAX NG <choice> element which means the xAxis element can either be written using the information in the first block of the choice (using the IDREF attribute only) or the xAxis element can be written using information in the second block (denoted by the <group> element). This second block of information is all of the information needed to define the xAxis element.

3. LAS improvements from using RELAX NG

Of course, all of this technical discussion of XML and XML schema languages is all for naught unless we can realize some significant improvement in the LAS software as a direct result of using these tools

3.1 Automatic XML validation

The first and most significant advantage to using a schema in general and RELAX NG in particular, is that installers of our software validate their instances of XML configuration documents using general purpose XML software. As developers and distributors of software we can easily validate the configuration XML as the first step in the process that ingests the XML configuration into the database. This saves the installer time and effort since mistakes in structure as well as syntax can be discovered and corrected right away.

3.2 Element Identification and Reuse

In our old XML we the name of the element was the thing that uniquely identified the contents of the element, meaning that we had to write the software the enforced the uniqueness of the element names. And when the element was to be reused, a cumbersome XML XPath name was used to identify the previously defined element. By using the XML Schema Datatype library in conjunction with RELAX NG, we can use the ID and IDREF attribute to identify and refer to elements.

This can be carried through to all parts of the software architecture so that operations and data requests can be made by referring to the ID of the element that defines the object.

3.3 OO languages and RELAX NG schema

There is a natural relationship between the RELAX NG definition of an element and collections of elements and objects in an object oriented programming language like Java. This allows us to create abstract programming constructs which easily and intuitively follow the structure of our RELAX NG XML schema definition leading to code that is easier to develop and maintain.

4. CONCLUSIONS

Creating a RELAX NG schema definition for the LAS configuration XML documents has allowed us to add new features, write better code to implement the features, validate our configuration files automatically and to build more robust software.

5. REFERENCES

Hankin, Steve, J. Davison, J. Callahan, D. E. Harrison and K. O'Brien, 1998: A Configurable Web Server for Gridded Data: A Framework for Collaboration, Preprints, Fourteenth International Conference for Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology, Phoenix, AZ., AMS, 417-418

Mertz, David, 2003: XML Matters: Kicking back with RELAX NG, Part 1, http://www-106.ibm.com/developerworks/xml/library/x-matters25.html

Mertz, David, 2003: XML Matters: Kicking back with RELAX NG, Part 2, http://www-106.ibm.com/developerworks/xml/library/x-matters26.html

Mertz, David, 2003: XML Matters: Kicking back with RELAX NG, Part 3, http://www-106.ibm.com/developerworks/xml/library/x-matters27.html

Clark, James and M. Makoto, 2001, RELAX NG Tutorial, http://www.oasis-open.org/committees/relax-ng/tutorial.html

Clark, James, The Design of RELAX NG, http://www.thaiopensource.com/relaxng/design.html