Astronomical Data Analysis Software and Systems X
ASP Conference Series, Vol. 238, 2001
F. R. Harnden Jr., F. A. Primini, and H. E. Payne, eds.

Specifics on a XML Data Format for Scientific Data

Ed Shaya,[1] Brian Thomas

Raytheon ITSS/NASA/ADC

Cynthia Cheung

NASA/ADC

Abstract. An XML-based data format for interchange and archiving of scientific data would benefit in many ways from the features standardized in XML. Foremost of these features is the world-wide acceptance and adoption of XML. Applications, such as browsers, XQL and XSQL advanced query, XML editing, or CSS or XSLT transformation, that are coming out of industry and academia can be easily adopted and provide startling new benefits and features. We have designed a prototype of a core format for holding, in a very general way, parameters, tables, scalar and vector fields, atlases, animations and complex combinations of these. This eXtensible Data Format (XDF) makes use of XML functionalities such as: self-validation of document structure, default values for attributes, XLink hyperlinks, entity replacements, internal referencing, inheritance, and XSLT transformation. An API is available to aid in detailed assembly, extraction, and manipulation. Conversion tools to and from FITS and other existing data formats are under development. In the future, we hope to provide object oriented interfaces to C++, Java, Python, IDL, Mathematica, Maple, and various databases. `http://xml.gsfc.nasa.gov/XDF`

1. Introduction

The eXtensible Data format is an XML-based language for describing and encapsulating scientific data. Its primary purpose is to be the mathematical and computer science kernel for other, more fully-featured, discipline-oriented formats. With the advent of global acceptance of XML as the mark up language for data, now is the time to rethink our data formats. It is a time when cross disciplinary interchange of data is greatly needed. At NASA there is a pressing need for earth scientists, solar physicists, astronomers, and biologists to communicate and to share data and processes. XML provides a common set of rules for documents that are a great boon to interoperability. A data format grounded

[1]Physics Department, U. of MD

in XML will not only be understandable to other scientists but could also be crafted so it can be viewed by the public with common browsers.

XML is a very important breakthrough in documentation and metadata expression. It brings in or formalizes the concepts of self-validation, self-description, recursive variable substitution, separation of presentation from information, hierarchical structures, the multi-file document, and standard multilangual character sets and tagging. It continues to grow in popularity. It is not likely that any of these important concepts in documentation will ever disapear.

The development of XML techniques now will, in the end, result in great cost benefits. Tools designed to use XML will likely be useable by other groups with minor adjustments. This is because all XML data are parsed with standard parsers and converted to a standard Document Object Model (DOM), which has standard application programming interfaces.

But we can go one step further in interoperability by accepting a common XML language specifically for data description. The present lack of a common object model of data and data description has made it very difficult to develop tools useful to a wide audience. Cross-disciplinary search is nearly impossible with the current technology.

2. The eXtensible Data Format (XDF)

An XDF document describes a structure of related scientific data. Our goal is to develop an XML language that reasonably covers the needed mathematical and computer science constructs in a generic data model. Discipline specific descriptors are left to specific languages that can inherit XDF capabilities. This allows the essential and common data structures to be represented in a consistent manner independent of the scientific specialty involved.

In XDF, data are stored in a manner that accurately reflects scientists' views of data. Most scientific numeric data can be seen as an object assembled from objects of two catagories: (1) a simple parameter set to a single value, possibly plus or minus infinity, or a range of values, possibly of infinite extent (e.g., $x = 3.1$ or $0 < y < 180$), or (2) gridded samples of scalar or vector fields embedded in an N-dimensional space (field arrays). These include tightly sampled grids such as spectra, images, animations, or time-series measurements. And, they include sparsely sampled fields such as interferometric u-v maps, event detection, or sets of pointed telescope observations.

Examples of N-dimensional spaces include physical space, projected space, time, wavelength, frequency, energy scales, or some other parameter space. Complex numbers can be considered as vectors in the complex plane.

These two basis objects can be assembled in several ways, into lists of lists or field arrays. A record is a list of values for a selection of parameters for a particular item or target. A list of these records is a table.

Because observational data always has some finite resolution, it is always gridded (sometimes variably). Therefore both field arrays and tables can be represented similarly as ordered N-dimensional data cubes. Software can take advantage of this similarity by sharing input/output methods between the two. In fact much of the data handling can be similar: subsetting, taking cross-sectioning, etc. However, the fundamental difference between the two categories is the fact that tabled properties rarely form a continuous space and therefore interpolation and analyses depending on interpolation are not sensible.

One of the key concepts in object oriented methodology is that data should be wrapped with the information necessary to read it and to make it useful. The XML language allows for this by including in data documents either references to applications or code in the form of ECMAscript or Java. An XML document can have references to files containing data and different types of data files can be handled by different applications. This not only allows input and preliminary processing to be self directed, but it also allows some of the data to be generated on the end users' machines. Eventually XDF will include functions for calculating values along axes, and if necessary, calculating positional information for every grid point. When applications are referenced or embedded within the data documents, it greatly reduces the learning curve needed to begin working with scientific data.

3. Examples

Here is a simple example:

```
<XDF>
   <structure name="structure A">
      <array name="array I" description="reduced data">
         <units>&Jansky;</units>
         <dataFormat>
            <float width="10" precision="5"/>
         </dataFormat>
         <axis name="x" axisId="x">
            <valueList count="512" start="23.32" step="2.22"/>
         </axis>
         <axis name="y" axisId="y">
            <valueList count="1024" start="12.11" step="2.26"/>
         </axis>
         <read readId="format">
            <for axisId="x">
               <for axisId="y">
                  <readCell/>
                  <skipChar/>
               </for>
            </for>
         </read>
         <data href="imagefile1"/>
      </array>
      <array name="array b" description="flat field">
         <unitless/>
         <dataFormat>
            <integer width="5"/>
         </dataFormat>
         <axis axisIdRef="x"/>
         <axis axisIdRef="y"/>
```

```
        <read readIdRef="format"/>
        <data href="imagefile2"/>
        </array>
    </structure>
</XDF>
```

One sees that the data description takes only a few minutes to understand. This is a structure with two arrays of data. Each is a 2-D image with axes "x" and "y" and the axis values are specified with start and step attributes. The second array reuses the information from the first one for axis and read instructions. The read section tells one to increment the x value after each row of y is read. The dataformat is F10.5 for the first array and I5 for the second with a single ignorable character between numbers. The units are Janskys. An entity &Jansky; is automatically replaced from an entity list with its SI equivalent. All XDF data end up in SI units in this way.

In our data model the axes are critically important. High resolution images can be subsections of the low resolution ones. Understanding how to do that comes from the fact that the axes data are placed in a well established manner. Axes from one array can point (using axisIdRef mechanism) to ones in other arrays to indicate that they are parallel or aligned. Arrays can be merged or appended to other arrays. Each component of vectors indicate which axis they line up with.

High dimensional tables are possible. A group of tables with the same layout that differ essentially by the value of some parameter would be represented by adding another dimension specifying the varying parameter. Data that need to be split at variable sizes can be segmented with each segment having a name. Field headings can also be grouped by higher level field names.

Other features include parameters with scoping, and inheritability. Another paper (Thomas, Shaya, & Cheung 2001) describes the FITSML language which has astronomical keywords and inherits XDF properties.

4. Future Versions

The XDF is being used at the Astronomical Data Center where it is getting a good solid testing of a large and disparate set of data. It is working out well but we welcome any suggestions from the scientific community for improvement or new ideas to be included in future versions. We expect new versions to come out every so often. There will be no problems with earlier version documents because applications can easily include XSLT scripts that transform earlier versions to newer ones immediately before processing. We feel it is quite possible that in a short time one data format will take you from observation, through XML manuscript creation `http://xml.gsfc.nasa.gov/article`, to XML query of archives.

References

Thomas, B., Shaya, E., & Cheung, C. 2001, this volume, 487