5/12550

XML Based Scientific Data Management Facility

P. Mehrotra
NAS Division, M/S T27A-1
NASA Ames Research Center
Moffett Field, CA 94035 USA
pmehrotra@arc.nasa.gov

M. Zubair
Department of Computer Science
Old Dominion University
Norfolk, VA 23529 USA
zubair@cs.odu.edu

Abstract

The World Wide Web consortium has developed an Extensible Markup Language (XML) to support the building of better information management infrastructures. The scientific computing community realizing the benefits of XML has designed markup languages for scientific data. In this paper, we propose a *XML* based scientific *data management facility, XDMF*. The project is motivated by the fact that even though a lot of scientific data is being generated, it is not being shared because of lack of standards and infrastructure support for discovering and transforming the data. The proposed data management facility can be used to discover the scientific data itself, the transformation functions, and also for applying the required transformations. We have built a prototype system of the proposed data management facility that can work on different platforms. We have implemented the system using Java, and Apache XSLT engine Xalan. To support remote data and transformation functions, we had to extend the XSLT specification and the Xalan package.

1. INTRODUCTION

We are entering the second phase of the World Wide Web revolution where the target for information is not a human, but a machine. In the first phase, a digital document was represented using HTML, which is rendered for display by browsers for human consumption. It was soon realized that HTML representation of a digital document has limitations. In particular, it makes the document unsuitable for machine processing, which is essential for building a distributed information infrastructure that can be efficiently searched and managed. The World Wide Web consortium has developed an Extensible Markup Language (XML) to support the building of better information management infrastructures. XML allows a community to describe its own grammar that meets its needs more efficiently. For example, it is now possible for a community to separate the structure of the document from its presentation. One can define a set of tags to represent the abstract structure of the document, which makes it suitable for machine processing.

The scientific computing community, also realizing the benefits of XML, has designed markup languages to represent scientific data. There are several initiatives focusing on this issue, such as the Extensible Scientific Interchange Language (XSIL) [1], and the eXtensible Data Format (XDF) [2]. We hope that finally the community will agree on one language for the scientific data representation. We believe that this language will have two components: a core component describing the structure of the scientific data and the second, discipline specific component, may contain metadata describing the circumstances of the data collection and other information for understanding the data details.

A workshop on Interfaces to Scientific Data Archives organized by California Institute of Technology made a strong case for an XML based scientific data management infrastructure [3]. In this paper, we propose a *X*ML based scientific *d*ata *m*anagement *f*acility (XDMF). The project is motivated by the fact that even though a large amount of scientific data is being generated, both experimentally and programmatically, relatively little is being shared among the scientists because of lack of standards and infrastructure support for discovering and transforming the data. The proposed XDMF will make the process of discovering data along with the relevant

transformations required for sharing such data, easier and more efficient. In particular, the focus is to automate this process and make it location independent such that the user, the data and the transformation code may be in distributed locations. Consider the situation in which a scientist wants to use some specific kind of data, for example, wind tunnel data, in the course of a simulation. The user visits the XDMF hosted, say in Virginia, and executes a search using some specific metadata fields. He is presented with a list of registered data satisfying his query. He selects one of the data sets from this list after examining the detailed description. The selected data is available from a site located, say in California (Note that the XDMF only keeps the XML document describing the data and not the data set itself). However, in many cases the data will be in a format not directly useful to the user and it would have to be transformed into another format before it can be utilized. The user can then search the XDMF for a list of applicable transformation functions. The user selects a transformation function located, say in Seattle. The XDMF retrieves the data from California, retrieves the transformation function from Seattle, applies the transformation and sends the transformed data to the user (we are assuming that the data and transformation functions are accessible through HTTP). In a more general scenario, the data would be required only at the time that the code is to be executed as a part of a larger application. In such situations, the proposed XDMF can be integrated into a larger framework and can facilitate the downloading and transformation of the data at runtime.

We have built a prototype of the XDMF that can work on different platforms. We have implemented the system using Java, and Apache XSLT engine, Xalan [4]. To support remote data and transformation functions, we had to extend the Extensible Style Language for Transformation (XSLT) specification and the Xalan package. For our initial prototype, we have used XSIL for representing the scientific data. Note that by doing this we are not endorsing any one initiative. Our objective is to demonstrate the benefits of an XML based data management facility. In fact, we also show that the current scientific data markup languages will need to be extended to build the proposed facility.

The rest of the paper is organized as follows. In the next section we provide some background on XSIL, a XML based scientific data interchange language and XSLT, an XML transformation language. Section 3 presents an overview and architecture of the proposed facility while Section 4 provides a brief description of the current prototype.

2. BACKGROUND

2.1 Extensible Scientific Interchange Language (XSIL)

The Extensible Scientific Interchange Language (XSIL) [1] has been developed by the Center for Advanced Computing Research, Caltech to represent the basic syntactic structure for scientific data such as *Table, Array*, and *Stream* in XML. The *Table* is similar to a relational table that contains an unordered set of records, each of the same format; the *Array* is collection of numbers of some other primitive data type; and the *Stream* element provides a link to external and encoded data through files and URL's. Two sample XSIL documents, one representing small

```
<?xml version="1.0"?>
<XSIL>
    <Array Name="Coordinates" Type="float">
    <Dim>4</Dim>
    <Dim>2</Dim>
    <Stream Encoding="Text" Type="Local"
    Delimiter=", ">1, 0, 1, 1, 0, 1, -1,1
</Stream>
</Array>
```

```
<?xml version="1.0"?>
<XSIL>
    <Array Name="Coordinates" Type="float">
    <Dim>4</Dim>
    <Dim>2</Dim>
    <Stream Type="Remote" Delimiter=", ">
    data.dat</Stream>
</Stream>
</Array>
```

Figure 1: Sample XSIL documents with local data (left) and remote data (right)

size local (in-line) data and the other representing remote data, are shown on the left and right side of Figure 1 respectively.

2.2 Transformations

The ease of transforming an XML document from one form into another is key to the XML usefulness. The transformations are typically necessitated when we move XML documents between two disparate organizations. In such a case, an XML document in one organization exists in a form different from the one in the other organization. This could be because the two organizations are using different languages to markup their data. For this purpose, the World Wide Consortium has introduced Extensible Style Language for Transformation (XSLT). One uses XSLT to write stylesheets, which essentially represent a set of instructions for transforming one XML document type to another. Note that you need an XSLT engine to process these instructions. An example of XSLT engine that is in public domain is Apache Xalan [4]. The XSLT specification also supports transformations like sorting of document elements, summing and averaging numbers, etc.

3. XDMF

3.1 Overview

In this project we have focused on *XDMF*, an XML based scientific data management facility for discovering and transforming data sets stored in distributed locations. Figure 2 gives an overview of the functionality of the XDMF. The XDMF interacts with three entities: the data generator, the

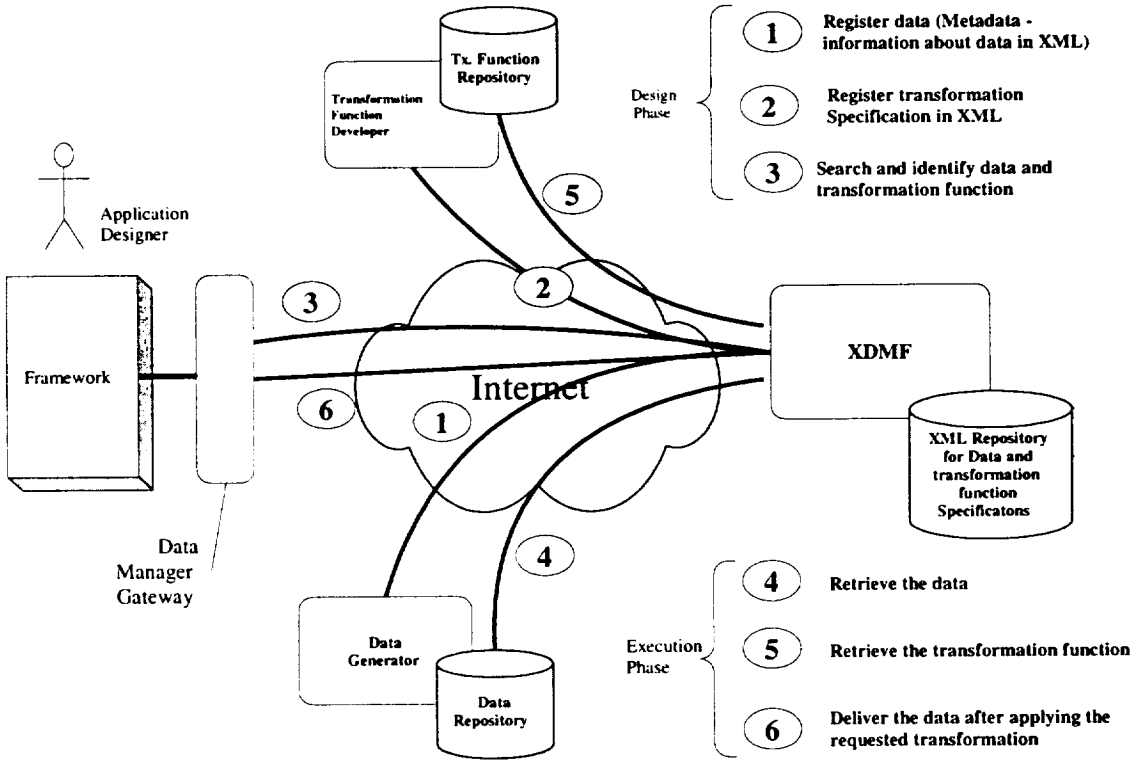

Figure 2: Overview of XDMF

transformation function developer, and a distributed computing framework. The data generator is responsible for registering the scientific data that is to be shared with other researchers. For this he uploads the XSIL file describing the structure of the data along with other metadata providing semantic information about the data. For example, the metadata could contain information about the conditions and constraints under which the data was generated. The transformation function developer registers the transformation function by uploading the XSLT specification along with necessary metadata that describes the type of transformation, the function support and the constraints under which the transformation is applicable. We are basing our approach for transforming scientific data on the XSLT engine. As the required scientific data transformation could be complex, for example converting a node-centered unstructured grid data in a CFD simulation to an edge-centered format, it is not possible to describe these transformations in the XSLT specification file. In such situations we will use the facility provided by the XSLT specification for referencing external transformation functions. Given that in most cases we will have to use external transformation functions, the question arises: why use the XSLT specification at all? The reasons for using XSLT specification are: (1) the input data specification is in XML and the transformed data is also specified in XML thus necessitating the use of a XSLT engine, (2) development cost is low as the XSLT engine, which is a standard module freely available in public domain, provides support for all the other required work like downloading the transformation function and the scientific data from remote sites, applying the transformation function on the downloaded data, and storing the transformed data and its XML specification.

We now illustrate the information flow by considering an application designer, working with a framework, who is in need of scientific data for his application. During the design phase of his application, he visits the XDMF and identifies a data set registered in the XDMF. Along with the data set, he also chooses an appropriate transformation function in the form of an XSLT specification. During the execution phase, the data management gateway requests the transformed data from the XDMF. The XDMF in turn, downloads the data and the transformation functions from remote locations, applies the transformation and returns the XML file describing the transformed data. The gateway software processes the XML file, retrieves the transformed data and supplies it to the application.

3.2 Architecture

The architecture of the XDMF, as shown in Figure 3, consists of (1) a digital library that holds the scientific data specification, transformation function specification along with other metadata, (2) data transformation component based on Xalan XSLT engine that retrieves the data and the transformation function from remote sites and applies the transformation, and (3) publication, search, and transformation request handlers. All interactions with the XDMF are based on HTTP. The data generator interacts with the XDMF publication handler to publish the scientific data specification in XSIL along with other relevant other metadata about the data. Similarly, the transformation specification developer interacts with the publication handler to publish the transformation specification and its metadata. The application designer interacts with the search handler to discover and identify the scientific data and the transformation function in the digital library. The framework gateway initiates retrieval request for the transformed data to the transformation handler, which in turn interacts with the data transformation component.

The Digital Library architecture is based on the Java-based search service that was developed for Joint Training, Analysis and Simulation Center (JTASC) [4]. The benefit of this architecture is that it is platform independent, and it can work with any Web server as it is based on Java servlets. Moreover, the changes required to work with different databases are minimal. Our current implementation supports two relational databases, one in the commercial domain (Oracle), and the other in public domain (MYSQL). The architecture employs a three-level caching scheme to improve performance [4].

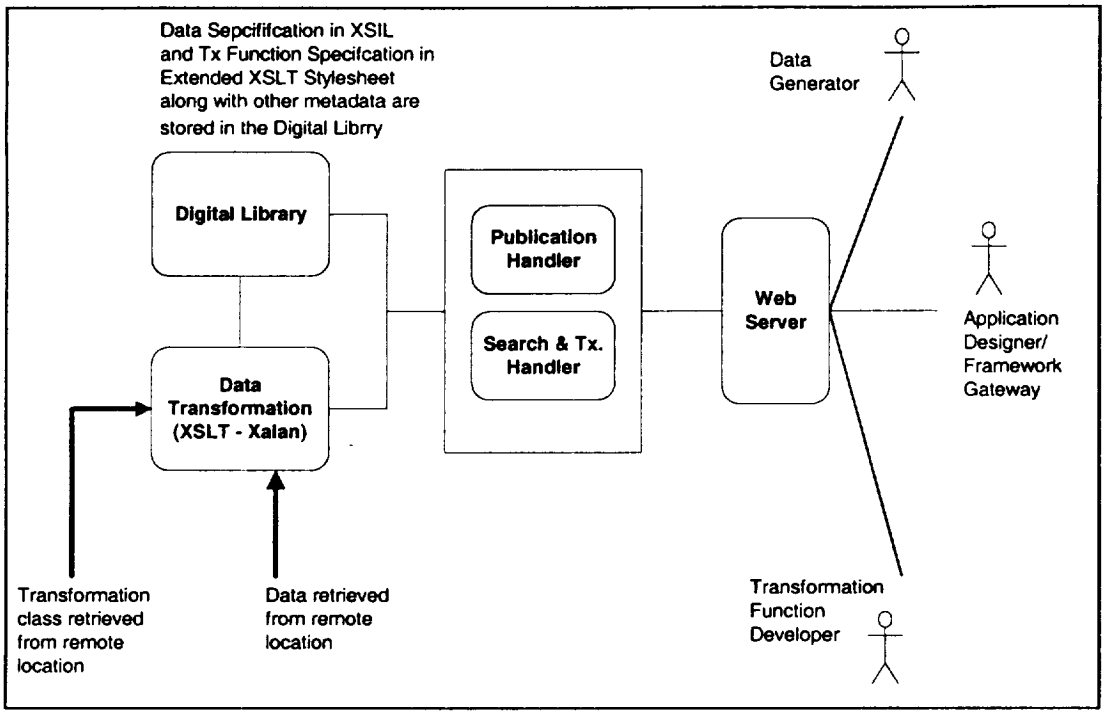

Figure 3: Architecture of XDMF

3.3 Extending XSLT Specification and XALAN

One major problem faced when using XSLT is its limited functionality, especially in performing complex scientific data transformations. The XSLT specification supports constructs for simple operations, such as, sorting and summation, only. However, scientific data transformations are in general much more complex. To address this issue, we have used the XSLT extension support to define new functions that include any scientific transformation logic and to associate them with Java classes. These functions can then be called in a XSLT specification. For this we have to make the following modifications in the XSLT specifications as shown in Figure 4. First, we have to declare an extra namespace for the extension along with an *extension-element-prefix* (lines 4-5, Figure 4). Second, we declare the new function, *polar3* here, and associate it with a remote Java class (lines 6-8, Figure 4). Lastly, we call the extension function, again *polar3,* in the appropriate transformation rule of the XSLT specification (line 14, Figure 4).

The Xalan package does not provide support for a remote Java class, e.g., specified via a URL, which has been associated with the external functions. As described above, access to remote transformations is central to the design of the facility (see Figure 2). To provide this support, we had to extend the Xalan-Java processor to handle the extension function calls specified via a URL by modifying the *ExtensionFunctionHandler.java* in Xalan package org.apache.xalan.xpath.

3.4 Prototype

We have implemented a standalone prototype that has the core functionality of the data transformation and digital library support. The current prototype allows users to select a scientific data specification along with the transformation to be applied. Once selected, the XMDF retrieves the scientific data from a remote location, say from www.icase.edu, retrieves the transformation function from, say www.cs.odu.edu, and applies the transformation, delivering the transformed data to the user over the Web. We have also implemented a Web based publication tool, which allows (a) the data generator to upload the XSIL specification of the data along with other

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:axslt="http://xml.apache.org/xslt"
    xmlns:myxslt=http://www.cs.odu.edu/~zubair/demo/RemoteXSLTExtensions
                extension-element-prefixes="myxslt" version="1.0">
 <axslt:component prefix="myxslt" functions="polar3">
 <axslt:script lang="javaclass" src="http://www.cs.odu.edu/~zubair/demo/RemoteXSLTExtensions.class"/>
 </axslt:component>
        ...
 <xsl:template match="Stream">
  <Stream>
        ...
   <xsl:variable name="stream" select="."/>
   <xsl:value-of select="myxslt:polar3(string(@Delimiter), string($stream))"/>
  </Stream>
 </xsl:template>
</xsl:stylesheet>
```

Figure 4: Modified XSLT specification for transforming data using an external function

metadata into the digital repository, and (b) the transformation function developer to upload the transformation specification along with necessary metadata into the digital repository.

4. CONCLUSION AND FUTURE WORK

In this paper we have proposed a XML based data management facility. The proposed XDMF provides support for: (a) registration of XML documents describing scientific data, (b) registration of XML documents describing transformations functions, (c) association of a scientific data set with the available transformation functions, (d) searching and browsing of scientific data based on specific metadata fields, (e) transformation of data once the user has identified the data and the transformation, (f) remote data and remote transformation functions. The proposed XDMF is easy and efficient to build as it is based on XML standards, thereby allowing reuse of publicly available tools such as parsers, and transformation engines. In future, we plan to develop APIs for the XDMF to allow framework developers to write their own data manager gateway software for integration with XDMF. We plan to work with other researchers in standardizing a XML language for specifying transformation function and other discipline specific component of the scientific data.

5. ACKNOWLEDGMENTS

This work was supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-97046 while the authors were in residence at the Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA 23681. We are also thankful to Tan and Jakatdar for implementing some of the modules for XMDF.

REFERENCES

[1] *Extensible Scientific Interchange Language (XSIL)*, http://www.cacr.caltech.edu/SDA/xsil/

[2] *eXtensible Data Format (XDF)*, http://tarantella.gsfc.nasa.gov/xml/XDF_home.html

[3] R. Williams (Ed.), *Interfaces to Scientific Data Archives*, Workshop Report, California Institute of Technology, Pasadena, 1998. http://www.cacr.caltech.edu/SDA/ISDA398/report.html

[4] Kurt Maly, Mohammad Zubair, Husham Anan, Dun Tan, and Yunchuan Zhang. *Scalable Digital Libraries based on NCSTRL/Dienst*, ECDL 2000.