Group Project Progress Final Report

CS 5513

Prototype for Wrapping, Integrating and Visualizing Geo-Referenced Information in Distributed Environment

Using XML Technology

Submitted to:

Dr. Le Gruenwald

Submitted by:

• Jianting Zhang

• Amir Shaheen

• Muhammad S. Javed
Index

Introduction .................................................................................................................. 2

Markup languages and Web-database applications .................................................. 2
Geo-referenced information related applications in Web environment .................. 3
Information Integration in Web Environment ......................................................... 4
Geo-referenced Information Integration in Web Environment ......................... 5
Our motivations .............................................................................................................. 6

Related work ................................................................................................................ 7

Proposed prototype overview ..................................................................................... 10

System Implementation in detail .............................................................................. 14

XML: semi-structured web database, its generation and parsing ......................... 14
Wrapper: Generating Geo-XML from Flat File(s) ...................................................... 16
Mediator: Geo-referenced information integrator .................................................... 19
Client Side Visualization ........................................................................................... 22

Results .......................................................................................................................... 24

Discussion and future work ....................................................................................... 29

References ................................................................................................................... 31

Journal/Conference/Book References ....................................................................... 31
Internet References ....................................................................................................... 32
Introduction

Markup languages and Web-database applications

During the past few years, we have witnessed dramatic developments in applications of Internet. With ever growing data and information present in governmental, enterprise and personal websites, the Internet can be seen as a huge loosely coupled distributed database. As the industrial standard of World Wide Web (WWW), Hyper Text Makeup Language (HTML) has played and will continue to play an important role in rendering data and information from servers to clients. HTML uses tags to describe how a web browser should arrange text, images and push buttons on a page (HREF 1). In this sense, HTML is a combination of content and visualization. Perhaps this is the right reason why it is so popular.

However, the limitations of HTML are also obvious due to this characteristic. The most important one is, as programming legend Brian Keringhan once noted, “What You See Is What You Get” turned into “What you see is all you’ve got” (HREF 2) i.e. it is very difficult to exchange information between different systems. Standard Generalized Markup Language, or SGML was an attempt to solve this problem by using tags to “say what the information is” instead of “what it looks like” in HTML. However SGML is too general for web browsers to cope with. XML, a simplified version of SGML (HREF 3), has gained more and more popularity in the past few years.

Compared to binary-based information exchange specifications, such as CORBA or COM, text-based specifications such as HTML and XML are more suitable for information
exchange in loosely coupled heterogeneous systems say, Web database applications. Our
point is that, HTML and XML are complementary rather than superseding. The reason is,
although XML specifies “what the information is”, HTML is still required to specify, “what
it looks like”. In fact, visualization is an indispensable component in information processing,
especially in web applications that are designed to target common users.

Although mark up languages work well with text and image data types, their earlier
versions had no built-in mechanism to deal with some unconventional data types. Two types
of extensions have been developed to solve this problem, namely the server side extension
and the client side extension. For the server side extension, CGI (Common gateway
interface), SUN’s Java servlet/Java server page (JSP) and Microsoft IIS API/Active server
page (ASP) are the most commonly used technologies. For the client side extension plugins,
Java Applet, ActiveX and script languages such as Java Script, JScript and VB script are
designed to interact with user’s selection. For more details see (Connoly, 1999). These
extensions give developers the ability to manipulate unconventional data types, which cannot
be interpreted directly by browsers.

**Geo-referenced information related applications in Web environment**

Geo-referenced information (GRI) is essentially a kind of spatial information with
earth related projections. It is extensively involved in governmental administration, public
affairs and private everyday lives. Although geo-referenced information can be expressed as
numeric values like other textual information but it makes little sense to the end user in the
numeric form. Instead, a map is the most commonly used form to display geo-referenced
information.
Unfortunately, GRI is among the aforementioned unconventional data types, which are not specified either in HTML or XML. To enable web-based GRI browsing, generally two strategies are adopted, namely image-based and vector-based. For the image-based strategy, GRI is stored as images either pre-generated or dynamically generated by server side extensions. For the vector-based strategy, GRI is first transferred to client side in vector format \((x, y)\) coordinates for a point and a sequence of points for a polyline and polygon) and then client side extension(s) render it in a graphic format for user visualization. Compared to HotMap, which is a feature in higher versions of HTML specifications, sophisticated client side extensions can provide more functions to support user operations such as Zoom In/Zoom out/Pan and other more advanced queries, etc. In the industrial practices, ESRI (HREF 4) and MapInfo (HREF 5) provide both image and vector based solutions while Intergraph (HREF 6) and AutoDesk (HREF 7) mainly focus on vector-based strategy. Java Applet based visualization for geo-referenced information research has been reported by (Sorokine, 1998).

**Information Integration in Web Environment**

Web database applications adopt client-server architecture and most of them are two-layer, i.e., client’s browser is directly connected to the Web server. The advantages of multi-layer architecture are well recognized and adopted by leading companies in the industry, such as SUN’s J2EE (Java 2 Enterprises Edition) (HREF 8) and Microsoft’s DNA (HREF 9). The middle layer, which is usually called business logic layer, is designed to incorporate information from several data servers and then send the final integrated information to client side user. The similar thing happens to web database application i.e., in
many situations, users might want to integrate information from several web sites to perform a comprehensive analysis. Compared to intra-enterprise information integration that usually has well-developed protocols, information integration in web environment is more difficult. This is mainly because each website involved is heterogeneous and autonomous. The visualization nature of HTML, which lacks the capability of conveying semantic information, is another important factor. The newly emerged XML technology is a promising technology for information integration in web-environment. The reason is that, just like HTML, XML can act as a common protocol between different Web servers and clients. Therefore, it is possible to build a mediator to integrate information from different Web servers and to build a virtual website to provide value-added information to clients. Although HTML based information integration is also theoretically possible, XML is more powerful and convenient. We'll discuss more on this topic later.

**Geo-referenced Information Integration in Web Environment**

Most of current web based mappings, no matter image based or vector based as we have mentioned earlier, adopt two-layer architecture. For a detailed survey see (Hardie, 1998). It is understandable from both software vendor’s perspective and data provider’s perspective. The reason being that two-layer architecture is simple and more efficient if no integration issue is considered. In fact, software vendors tend to make their customers loyal to their products by making it difficult for them to use other sources of data. Data providers usually are reluctant to share their data with others.

However, in the near future, our view is that, probably due to wide application of GPS, personal-based instead of governmental based geo-referenced information will be
overwhelming. Geo-referenced information will not be provided by few data providers any
more. Virtually everyone could provide geo-referenced information in distributed web
environment. It will be the case that some data vendors provide basic geo-referenced
information while users put thematic geo-referenced information together with text
information associated with it for domain specific applications.

Our belief is that the basic geo-referenced information is public information and
should be free. Compared to buying such data for each application and physically combining
it with user’s own thematic data, it is better to provide a mechanism to let user logically refer
to the public website data as the base map and allow him to simply integrate it with his own
thematic data. This is essentially a problem of information integration in Web environment.

On the other hand, just like the autonomous nature of Internet needs a searching
engine, with more and more independently captured geo-referenced information being put
on the Internet, searching engines with geo-referenced capabilities are highly demanded.
Unlike current text-based search engines, which only provide redirect function, most geo-
referenced information is only meaningful under context. In this sense, a good web-based
geo-referenced information search engine is actually a virtual geo-referenced information
system to perform intelligent querying.

Our motivations

Our motivations of developing a prototype for wrapping, integrating and visualizing
geo-referenced information in web environment are closely related to the above discussions.
1. Developing a three-layer XML based architecture that allows flexible web databases integration.

2. Integrating geo-referenced information from distributed and heterogeneous web sites that provide geo-referenced information.

3. Visualizing integrated geo-referenced information for better understanding and decision-making.

**Related work**

Much work has been done on web databases. In a survey paper, Florescu etc. have raised several fundamental issues in this field, such as data representation for Web DB tasks, modeling and querying the web, information integration and web site construction and restructuring (Florescu, 1998). Among these topics, web query that is very important for web database integration seems to attract more interest (Konopnicki, 1998, Bhowmick, 1999).

The idea of wrapping heterogeneous databases to provide common interface for information integration is also well recognized (Thiran, 1999). XML has been recognized as a good method for web-based information integration, as noted by Florescu etc, “XML may lead web site builders to export the data underlying their sites in a machine readable form, thereby greatly simplifying the construction of wrappers” (Florescu, 1998). More recently, XML based information mediation has been extensively studied in MIX project (Baru, 1999). These studies suggested that XML has the potential capability to serve as a common protocol for information integration in web environment.

As we had mentioned before, XML has no specifications on visualization issues. It is often the case that XML is combined with HTML in Web-based applications, i.e., XML is responsible for information exchange and HTML is responsible for display. In this case
XML is finally mapped to HTML usually through XSL (XML Stylesheet Language) for display. However, there are many XML elements that cannot be mapped to HTML elements and thus cannot be visualized directly. A solution proposed by Ciancarini is to use java applets to display Z specifications written in XML (Ciancarini, 1999).

As far as geo-referenced information is considered, there have been continuous efforts to integrate multiple sources and multiple formats of geo-referenced information. Generally, there are two directions towards this goal. The ideal one is to follow the same standard. In this field the most significant contribution is Open GIS Consortium (OGC). OGC has published its Abstract Specifications Version 4 and implementation specifications for OLE/COM, CORBA and SQL (HREF 10). If all GIS vendors follow the same standard, then both the software and the data are interpretable locally and in distributed environments. However, the history of processing digital geo-referenced information is over 30 years old and there are various types of software systems, data formats. It is highly unlikely that all the data and the associated applications can be transferred to new systems in a short time. The practical direction is to develop a set of APIs for each data format. Geo-referenced information can be integrated by calling APIs (with or without transferring the data file to other formats) locally or in distributed environment. Experiment on developing a Japanese interoperable test-bed on Open GIS follows the first direction (Shimada, 1999). In this system, the authors developed a three-tier model that consists of clients, legacy database wrappers and Geo Spatial Mediator (GSM). And the also proposed the CORRBA implementation. Alternatively, (Larouche, 1997) followed the second direction and proposed a practical solution to integrate geo-reference information which is now commercially available (HREF 11).
The XML technology also has a great influence on web-based geo-referenced information applications. Oracle has announced its Oracle8/GeoXML Server and has put it in the OGC website for testing (HREF 12). It transfers spatial information from ORACLE database to XML but no visualization function has been provided. OGC has recently announced its Geography Markup Language (GML) Recommendations, although no formal publication is available yet.

With more and more web-based semi-structured information expressed in SGML or XML format, the demand for integrating XML and geo-referenced information has also emerged. A related study (Amann, 1999) shows that although it is possible to integrate SGML with commercial GIS software to provide browser-based visualization and query, the cost is not little. It would be very interesting if all the information, both text-based and geo-referenced information, is expressed in XML format then the task of integration might be easier. As far as our literature review is concerned, no study on integrating and visualizing geo-referenced information has been reported.

Unlike information in relational databases, geo-referenced information has more data types and relationships between different objects. Metadata plays important roles in geo-referenced information applications. Both OGC and national (such as Federal Geographical Data Committee -FGDC) and international organizations (such as ISO /TC211) have contributed a lot to the metadata research. There are 7 parts/topics in FGDC geo-referenced information metadata standard, i.e., identification information, data quality information, spatial data organization information, spatial reference information, entity and attribute information, distribution information and metadata reference information. For more information please visit their websites and also refer to (Gobel, 1998). Unlike current proposals that store metadata and data itself separately, tag based XML has the capability of
integrating metadata and data in one document. At the same time, Data Type Definition (DTD) for XML enables carrying semantics for the semi structured GRI and validating its XML document at the real time.

**Proposed prototype overview**

In this prototype research, we use XML as the communication protocol between distributed web sites (where GRI resides) and mediator, and between mediator and clients. Java servlet programs are written to translate data in distributed websites into an XML document. Data in distributed website can be stored in a flat file, relational database, object-oriented database or object-relational database. A java servlet program in mediator server will retrieve the data from related distributed websites in XML format upon the request from client side, parse the retrieved XML documents, do merge or other operations on the retrieved XML document to build a new XML document and send it back to the client side. When the client side gets the requested data from the mediator server, it will parse the returned XML document and draw it inside the browser using a Java applet. The architecture is shown in the following fig 1.
Although, there have been studies in web-based database wrapping, XML based information integration, applet-based XML elements visualization, web-based geo-referenced information mapping, geo-referenced information integration with three-layer architecture, our proposal of integrating all these features to build a prototype for wrapping, integrating and visualizing geo-referenced information in web-environment is novel. Table 1 shows the comparison of our prototype with existed studies.
The reasons to use XML as the common protocol for geo-referenced information exchange lies in the following factors:

1. Text-based information exchange protocol ensures platform independence and easy implementation.

2. Data provider can be any system with any data format as long as it can generate XML document, which is an industry standard.

3. The generated XML document is application independent, which can be used in any applications as long as the application can parse XML document.

4. It is more efficient if there are many attributes in the whole data set but few attributes are associated with each object, i.e., relational table is sparse which is usually the case for geo-referenced information.
5. The possible reason against XML is that a text based XML document is much bigger compare to a binary one. However, it can be compensated by applying compression methods between the server and the browser.

The reasons to choose Java as our implementation platform are:

1. Free, easy to use and availability of source code.
2. Built-in network and distributed process support.
3. Built-in XML classes support such as SAX (Simple API for XML) and DOM
4. Java 2D graphic support for visualization.
5. Easy to integrate with other Java based web database applications to extend the functionality of the prototype.

We also want to make the following issues clear:

1. This prototype is built on a practical basis rather than a theoretical one. The primary aim is to show how geo-referenced information can be integrated and visualized using newly emerged XML technology. However, it might play as a test-bed for theoretical studies.
2. Since our work is only a prototype system, many functions are not fully implemented and even some implemented functions are simple. However, the architecture is extensible which allows developments of more functions on integrating geo-referenced information that current stand-alone GIS software system has, such as spatial querying and spatial analysis.
3. We have planned to implement the GRI wrapper on an object-oriented database such as Objectivity and a relational database such as Oracle. Due to time and some practical limitations, we have only implemented the wrapper on flat files.
However, this technique should be similar and easy to be transferred to other wrapper implementations.

**System Implementation in detail**

As we have explained in system overview, there are three main components in the prototype system i.e. geo-referenced data wrapper, mediator and client side visualization. We will only describe what we have done and will leave the future plan discussion in the future work section. Before discussing each component, we’d like to introduce some technical issues on XML generation and parsing which is the basis for XML-based web information integration.

**XML: semi-structured web database, its generation and parsing**

```xml
<?xml version='1.0' encoding='us-ascii'?>
<slide type="all">
    <title>Wake up to!</title>
</slide>
</slideshow>
```

Fig 2 A Simple XML Document

A simple XML file is shown in fig. 2. Just like HTML, there are tags associated with the contents. However, these tags are used to indicate what the data is instead of how it looks like. To make use of XML, there are two major issues, one is how to generate XML and the other is how to parse XML. Up to now, major software companies (HREF 13, HREF 14, HREF 15, HREF 16) have announced their supports for XML. In this prototype, we are using SUN’s XML API to generate and parse XML.
Basically there are two ways to manipulate an XML file. One is based on SAX (Simple API for XML) and the other is based on DOM (Document Object Model). Both of them are W3C recommendations (HREF 3). We'll briefly describe them here. For more detailed information please visit SUN’s tutorial website at (HREF 17).

SAX is the event-driven, serial-access mechanism for accessing XML documents. There are several predefined callback methods in a Document Handler (org.xml.sax.HandlerBase/DocumentHandler in SUN’s XML API, for example), such as startDocument(), endDocument(), startElement(), endElement() and characters(). By calling those methods, an XML document can be generated. Whenever an XML document is being parsed, the system will call these callback methods and pass the parameters of the elements it is processing sequentially. Thus user can get both the name and the value of these parameters inside the functions.

The DOM based method maps an XML document to a tree. Each element in the document is a node. In this method, user can create a new document tree, insert or delete elements and travel the document tree by calling APIs. Internally, DOM building also needs to call SAX callback functions. In SUN’s XML API functions such as appendChild(), createAttribute(), createElement(), appendChild(), setNamedItem() are provided to generate an XML document and add elements and their associated attributes to the document. Note that each element can have a hierarchy.

Generally speaking, the SAX based method generates XML elements according to the sequence of inputted data items. There is no need for SAX based method to maintain the whole XML document in the memory during the generating process and therefore, it is more suitable for generating large XML files. On the other hand, DOM based method is
easy to understand and use. The user can directly make as many modifications as he wants to the elements (nodes) in the document (tree) before it is finally outputted.

Based on our experiences, the tree-based DOM model is more suitable for georeferenced information integration using XML technology. The reason is that a georeferenced object often has multiple levels of attributes that can hardly be arranged in one XML element. In SAX based method, each element in a XML file is independently processed sequentially in a first-in-first-out manner and thus can hardly maintain the hierarchy of geo-referenced objects. On the contrary, DOM model allows the user to travel through the whole document tree. The tree hierarchy is a good representation of multiple levels of attributes of a geo-referenced object.

**Wrapper: Generating Geo-XML from Flat File(s)**

Point, polyline and polygon are three major data types in geo-referenced information. We have experimented with two sets of data, one is cities, rivers and countries of the world and the other is name places, rivers and census tracks in Cleveland County, Oklahoma, USA. Both of the sets have point (cities/name places), polyline (rivers) and polygon (countries/census tracks) data types. Both of the experiments were successful.

The format of flat files that we used is GIS industry leading company ESRI’s ShapeFile. To read data from file into memory, several Java classes, developed by the computational geography center in the University of Leeds, UK (HREF 1), are used. Our work is to map those classes into XML file. We adopted the following DTD that is a subset of Oracle GeoXML (HREF 12):
Fig 3 GeoXML DTD in the Prototype System

Note that ATTLIST is a basic XML data type CDATA and CLIST is another basic XML data type PCDATA that means Parsed Character Data. Thus for ATTLIST, DOM API can add pairs of attributes and their associated values to the element. But for CLIST, we need to convert x, y coordinates of a point or a sequence of points into a string by ourselves. In accordance, XML parser will perform further processing to get the set of attribute names and their associated values when parsing a GeoXML document. But for CLIST, the XML parser will simply treat it as a string and we need to develop our own parser to extract x and y coordinates or the sequence of points from the string later.
The process of generating Geo XML file from ShapeFile is as Fig 4:

Fig 4 Process of generating XML document from ShapeFile by the GeoXML wrapper

We put the wrapper procedure in a Java servlet called GeoXmlServlet. A wrapper can provide one XML document or provide multiple XML document by taking parameters. It serves two purposes:

1. First, the user can get the data in XML form by visiting this URL directly.
2. Second, the mediator can get the needed data by standard URL request to the distributed Geo-XML data servers.

The syntax to retrieve a Geo-XML file from our website in this prototype is like:

http://129.15.192.159:8081/servlet/GeoXmlServlet?layers=0

Where parameter “layers” after the servlet name is the layer number. In our two experiments, layer numbers and their contents are listed below:

<table>
<thead>
<tr>
<th>Layer Number</th>
<th>Experiment 1 (World)</th>
<th>Experiment 2 (Cleveland County, OK, USA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Cities (point)</td>
<td>Name places (point)</td>
</tr>
<tr>
<td>1</td>
<td>Rivers (polyline)</td>
<td>Rivers (polyline)</td>
</tr>
<tr>
<td>2</td>
<td>Countries (polygon)</td>
<td>Census Tracks (polygon)</td>
</tr>
</tbody>
</table>

**Mediator: Geo-referenced information integrator**

The mediator is also built into a java servlet, which enables the end user to get the integrated information in XML format by visiting the mediator URL directly to support multiple purposed applications.

Mediator in this prototype is first designed to act as a search engine. It visits the distributed data servers automatically and gathers metadata. This data is stored into mediator’s Meta database according to geographical extent and information category. It accepts user’s high-level query and finds relevant web sites from its Meta database. Then the mediator dissolves the query into sub-queries and sends them to related web sites. Finally, the mediator gathers the returned XML files from distributed web sites to build a new XML
document with integrated information and sends it to the client side for visualization. However, due to time limit we are not able to completely fulfill this aim. We’ll leave the rest of the work for future.

Up to now, our mediator only supports XML documents merging to build a new XML document. The merging strategy is relatively simple i.e. extract elements of each XML document and put them into one XML document. Also, metadata stored in the mediator’s Meta database is manually set. The process of GeoXML document integration is shown in fig. 5.

It is often the case that there will be multiple spatial data types appearing in one XML document. To enable the storage of all types of geo-objects, we have defined an abstract class as the interface class of point, polyline and polygon data types. By doing so, all Geo-referenced object can be put into a vector as the instance of the abstract class for GeoXML integration and visualization. The definitions of Java classes are shown in Fig 6.

In our prototype, the syntax for retrieving integrated XML document from the mediator (wrapper server) is as follows:

1. Retrieving meta data:

   http://129.15.192.159:8081/servapp/servlet/GeoXmlMserv?layers=meta

   Currently, a brief XML document description, URL, number of geo-objects and layer sequence number are stored in mediator’s Meta database.

2. Retrieving integrated XML data:

   http://129.15.192.159:8081/servapp/servlet/GeoXmlMserv?layers=110
Where the parameter “layers” after the servlet name is the combination of layers requested. Suppose there are three layers in the Meta database, in the above case, 110 means we want the first and second layer but not the third one. Note visualization applet on the client side will also use the same syntax to retrieve information from the mediator, which we’ll describe in detail below.

![Diagram](image)

**Fig. 5** Process of integrating GeoXML Documents by the GeoXML Mediator
public interface AbstractShape
{
    public abstract int getShapeType();
}

public class NewPoint implements AbstractShape
{
    double x,y;
    public NewPoint(double x,double y);
    public double getX();
    public double getY();
};

public class NewPolyline implements AbstractShape
{
    Vector xx,yy;
    public NewPolyline()
    public void addPoint(double x,double y);
    public double [] getXX();
    public double [] getYY();
}

public class NewPolygon implements AbstractShape
{
    Vector xx,yy;
    public NewPolygon ()
    public void addPoint(double x,double y);
    public double [] getXX();
    public double [] getYY();
}

Fig. 6 Java Classes Definitions for mapping GeoXML document

**Client Side Visualization**

The client side responds to user’s choices and communicate with the mediator to retrieve integrated information and then visualize it. A client can either be any standalone program that can parse XML or can be any object embedded into web browser under the
system architecture. In our prototype, we use Java Applet to embed the visualization functions into the web browser.

First, a form based user-interface (contents are actually metadata of all existing layers retrieved from the mediator) using servlet is developed to get user choices. Another servlet is built to dynamically generate an HTML page containing the visualization applet together with the parameters of user’s selection passed by the first servlet.

Then the client side visualization applet will contact with the mediator and retrieve data from it according to the parameters. After the data is retrieved, the applet will render the data in graphic form for visualization. The process of client side visualization based on GeoXML is shown in Fig. 7.

![Process of Client Visualization based on GeoXML](image)

Fig. 7 Process of Client Visualization based on GeoXML
Suppose there are three data sets registered in the mediator, user can select any combinations of them. The syntax is like:

http://129.15.192.159:8081/servapp/servlet/GotoXML

By submitting selection, the servlet will guide the user to the following URL for visualization:

http://129.15.192.159:8081/servapp/servlet/GenApplet

Currently, the frequently used geo-referenced information visualization functions, such as “Full Extent”, “Zoom In” and “Zoom Out” are implemented in our visualization applet. Compared to existing commercial products, the functions of our prototype are limited in graphic rendering, user customization and complex query support. However, the architecture is developed to allow some extra functions to be implemented, such as panning the map, identifying the attributes associated with the geo-referenced objects and hyper-linking the location to some other URL. In fact, the architecture allows implementation of full-scale web-based geographical information system to support complex query and spatial analysis, which is our long-term goal but it is beyond the scope of this prototype system.

Results

Below are some screen snapshots to show some of the results of our prototype:

1. Generated GeoXML document by flat file wrapper and viewed in IE5.0 (Name Places) (Fig. 8).

2. Integrated GeoXML document by combing two layers (Census Truck and Name Places) (Fig. 9).

3. Client Visualization: Layer selection. (Fig. 10)
4. Client Visualization: Full Extent (fig. 11)

5. Client: Visualization: Zooming In with Selection Box (Fig. 12)

6. Client: Visualization: Zoomed In (Fig. 13)
Fig. 8 Generated GeoXML document by flat file wrapper (Name Places)

Fig. 9 Integrated GeoXML document by combing two layers
Fig. 10 Client Visualization: Layer selection

Fig. 11 Client Visualization: Full Extent
Client: Visualization : Zooming In with Selection Box

Fig. 13 Client: Visualization : Zoomed In
Discussion and future work

In this prototype system, we have implemented XML wrapper for geo-referenced information from distributed node in flat text format, developed a Geo-XML mediator which merges XML documents from distributed websites and provides integrated information in XML format and finally developed a visualization Java applet to retrieve geo-referenced information from mediator and visualize it.

Compared to existing web based GIS systems, our prototype using XML as the underlining geo-referenced information data format and communication protocol has the following advantages:

1. Openness: Any application that supports XML can make use of geo-referenced information provided by GeoXML wrapper. This is a very important advantage for future geo-referenced information applications.

2. Integration: By introducing a three-layer architecture and building a mediator, we can build a virtual GRI data warehouse to provide integrated GRI service. None of existing web GIS has this capability yet.

3. Visualization: Help people understand geo-referenced information better so that they can make better decisions.

At the same time, we are full aware of the limitations of the prototype system. Future enhancements can be followings:

1. More advanced geo-referenced information data model in XML, such as earth-related geometric projections and supporting more GRI data types.

2. More functions of GoeXML wrapper, such as only return part of data set based on client’s query.
3. More functions of GeoXML mediator, such as complex query support, building spatial and attribute index for Meta database to improve search speed, flexible data sets selection and even fuzzy and intelligent spatial searching.

4. More functions in visualization applet, such as pan, identify and hyperlink. It is also a good idea to implement some of spatial query and analysis functions, such as within/intersection based spatial query and buffer/network analysis.

5. User interface improvement. Such as allowing user to select color, size and pattern for geo-referenced objects, decide layers and/or objects sequence, etc. These features have already been provided by current standalone GIS software.
References

Journal/Conference/Book References


Internet References

HREF 1: [http://www.w3.org/MarkUp/](http://www.w3.org/MarkUp/) [Access Apr 20, 2000]


HREF 4: [www.esri.com](http://www.esri.com) [Access Apr 20, 2000]

HREF 5: [www.mapinfo.com](http://www.mapinfo.com) [Access Apr 20, 2000]
HREF 6:  [Access Apr 20, 2000]
HREF 7:  [Access Apr 20, 2000]
HREF 8:  [Access Apr 20, 2000]
HREF 9:  [Access Apr 20, 2000]
HREF 10, OpenGIS Specifications, [Access Apr 20, 2000]
HREF 11 [Access Apr 20, 2000]
HREF 12, Oracle8i GeoXML Server, [Access Apr 20, 2000]
HREF 13: ORACLE: [Access Apr 20, 2000]
HREF 14: MICROSOFT: [Access Apr 20, 2000]
HREF 15: IBM: [Access Apr 20, 2000]
HREF 16: SUN: [Access Apr 20, 2000]
HREF 17: [Access Apr 20, 2000]