MODELING GML AND SVG DATA FOR WEB GIS

by

LIANG ZOU

(Under the Direction of Xiaobai Yao)

ABSTRACT

The development of geographic information system (GIS) is accomplished by increasing popularity of Internet GIS. However, currently most commercial Web GIS applications use proprietary data sets and employ raster images to disseminate maps. The traditional method of Web GIS lacks interoperability, efficiency and quality. To overcome these problems, a conceptual framework is proposed to achieve data interoperability and efficient dissemination of high quality maps by using eXtensible Markup Language (XML). The framework uses the Web Feature Service (WFS) to provide feature level Geography Markup Language (GML) data, which are used to store and exchange geographical information. The Scalable Vector Graphic (SVG) standard is used to visualize GML data in web browsers. The Web Map Service (WMS) is used to provide remotely sensed images. Two prototypes are developed to demonstrate the feasibility and efficiency of the framework. It is concluded that advanced Web-based GIS applications can be implemented by applying GML, SVG, WFS and WMS techniques.

INDEX WORDS: XML, GML, SVG, WFS, WMS, Web GIS, Interoperability

MODELING GML AND SVG DATA FOR WEB GIS

by

LIANG ZOU

B.S., Peking University, China, 2004

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment

of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2006

MODELING GML AND SVG DATA FOR WEB GIS

by

LIANG ZOU

Major Professor:	Xiaobai Yao
Committee:	Marguerite Madden Thomas R. Jordan

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
May 2006

ACKNOWLEDGEMENTS

The writing of the thesis has been a long and arduous journey of learning. From the initial idea to the final product, this thesis has been finished with the generous help of many individuals.

First and foremost, I would like to take this opportunity to express my intensive gratitude to Dr. Xiaobai Yao, my major advisor, for her continued support, constructive comments and encouragement. Without her invaluable professional knowledge and guidance, I would not have finished the prototypes and thesis in such a short period. It is my great pleasure to have the precious opportunity to work with her during the last two years.

My deep and sincere appreciations are due to Dr. Marguerite Madden and Dr. Thomas R. Jordan, who are the Director and Associate Director of the Center for Remote Sensing and Map Science (CRMS), respectively. They have served as my committee members and dedicated so much time to my research. Their generous financial and technical support has made the thesis progress so smoothly.

I am grateful to Ms. Linna Li at the University of South Carolina for her help in searching appropriate references and providing me with suggestions on various matters of my thesis. I also would like to extend my thanks to Mr. Fuyuan Liang and Ms. Bo Xu for sharing their views and commenting on my ideas. Taking the same opportunity, I want to thank those people who have been concerning the progress of my thesis, such as Mr. Jason Randall Ridgeway, Mr. Adam Hinely, Ms. Yanfen Le, Ms. Yanbing Tang, Ms. Qingfang Wang and so on. My appreciations are also reserved for faculty and staff of CRMS and the Department of Geography.

Finally I am grateful to my parents and my family for their understanding, patience and love. Without their support, I would not have been able to make it this far.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ACRONYMS

API	Application Program Interface
ASCII	American Standard Code for Information Interchange
CGI	Common Gateway Interface
CSS	Cascading Style Sheets
DEM	Digital Elevation Model
DGIS	Distributed Geographic Information Systems
DOM	Document Object Model
DOQ	Digital Orthophoto Quadrangles
DRG	Digital Raster Graphics
DTD	Document Type Definition
EPSG	European Petroleum Survey Group
ESRI	Environmental Research Systems Institute
GIS	Geographic Information System
GML	Geography Markup Language
GPS	Global Positioning System
GUI	Graphic User Interfaces
HTML	Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
J2EE	Java 2 Enterprise Edition

JVM	Java Virtual Machine
OGC	Open GIS Consortium
PC	Personal Computers
PDA	Personal Digital Assistants
PHP	PHP Hypertext Processor
RDBMS	Relational Database Management System
SGML	Standard Generalized Markup Language
SVG	Scalable Vector Graphics
URL	Uniform Resource Locator
USGS	U.S. Geological Survey
VB	Visual Basic
VRML	Virtual Reality Modeling Language
W3C	World Wide Web Consortium
WCS	Web Coverage Service
WFS	Web Feature Service
WKB	Well-Known Binary
WKT	Well-Known Text
WML	Wireless Markup Language
WMS	Web Map Service
WPS	Web Processing Service
WWW	World Wide Web
XML	eXtensible Markup Language
XSLT	eXtensible Stylesheet Language Transformation

CHAPTER 1

INTRODUCTION

1.1 Background

In recent years, our society is becoming more and more dependent on geographical information. As the predominant technology in the 21st century, the Internet has driven the whole world into a new era. To meet the increasing demand of public access to geographical information, it is therefore inevitable to combine the Internet technology and geographic information systems (GIS).

The development of GIS has resulted in increasingly popular Internet GIS, such as the websites of MapQuest and Google Map. Internet GIS is an Internet-centered GIS technology that uses the Internet as the primary means to access data, conduct spatial analysis, and provide location-based services (Peng 1999; Peng and Tsou 2003). World Wide Web (WWW) is the current dominant application of the Internet.

Researchers, practitioners, and vendors are all actively interested in exploiting Internet GIS and searching various ways to promote the accessibility to geospatial data and data processing services on the Web (David *et al*. 1998). For instance, several very famous GIS vendors have developed their proprietary Internet GIS software, such as Environmental Research Systems Institute (ESRI)'s ArcIMS, Autodesk's MapGuide, Intergraph's Geomedia WebMap Professional, MapInfo's MapXtreme, ER Mapper's Image Web Server and GE SmallWorld Internet

Application Server. This trend has accelerated the public application of GIS and helped to overcome several limitations of desktop GIS software packages.

To disseminate geospatial information, GIS takes advantage of networks, especially the WWW. Concerning the techniques used nowadays to display maps on the Web, it is evident that most of them are recorded in one of the standardized raster file formats like GIF and JPEG. They are then embedded in HTML-based (Hyper Text Markup Language) web pages and published to the public. The major advantage of this method is that every user can browse the information despite different operating systems or web browsers. This is because the raster image formats and the HTML are all standardized and accepted by the official World Wide Web Consortium (W3C). However, the original design of HTML focuses on the display of text and graphical information rather than geospatial information. Thus there are several limitations of HTML. For example, it is difficult to project, integrate, and exchange spatial data by using HTML. In addition, the raster structure of GIF or JPEG graphs severely limits the quality and interactivity of geographical maps. The authors have to “either choose a low resolution image with inherent low graphic quality, or a high resolution, with a better quality image, resulting in large file sizes and therefore long loading times” (Köbben 2003 p. 1). Another important limitation of most current Internet GIS is the lack of interoperability. Interoperability is the ability of a system, or components of a system, to provide information portability and inter-application cooperative process control (Zhang *et al*. 2003). Generally, there are two meanings for interoperability when it is concerned. First, it means that a data set can be accessed by different programs. Second, it refers to an interoperable program which is able to utilize a range of data formats (Laurini 1998).

Implemented by commercial Internet GIS software, Internet GIS depend on specific data structures, formats, databases, and system architectures, *etc*. Thus the data and processing services between different Internet GIS programs cannot be shared and interoperated. Despite the fact that many important GIS vendors are trying to make their products more interoperable, it is almost impossible to achieve the dramatic progress in the near future. Finally, although some Internet GIS sites can perform many sophisticated functions, such as panning, zooming, toggling layers, or even buffering and overlay, the fact is that most of these systems need to use the server to generate a simple GIF or JPEG image after a user sends a request from the client side. The architecture used here is apparently inefficient if we need to interact with the geospatial data frequently or if we want to customize the view of the displayed images.

To overcome these limitations described above, a promising solution would be to use eXtensible Markup Language (XML) for information discovery, exchange, and display over the web browsers (Peng and Tsou 2003). Conceptually, an XML document comprises a sequence of elements nested inside one another. The structure of the resulting document is referred to as a document tree. One single parent element may have several child elements. The document root is the outermost parent element, which does not have its parent element. For the elements, they may be qualified by one or more attributes (Green and Bossomaier 2002).

The Open Geospatial Consortium[1] (OGC) is an international industry consortium of more than 270 companies, government agencies, and universities participating in a consensus process to develop publicly available interface specifications. The mission of OGC is to develop spatial

[1] OpenGIS Consortium has changed its name to Open Geospatial Consortium

interface specifications that are openly available. OGC Specifications support interoperable solutions that “geo-enable” the Web, wireless and location-based services, and mainstream IT. The specifications empower technology developers to make complex spatial information and services accessible and useful with all kinds of applications. The OGC has initiated several Web mapping interoperability initiatives and specifications, which include the Geography Markup Language (GML) Implementation Speciation, the Web Map Service (WMS) Implementation Specification, the Web Feature Service (WFS) Implementation Specification, the Web Coverage Service (WCS) Implementation Specification and the Web Processing Service (WPS) Implementation Specification. (OGC 2005a). The focus of these specifications is to define interface standards which allow seamless and vendor neutral access and interchange of GIS data. Many companies and organizations are adopting and implementing OGC standards to provide Web-based services. For example, the Terraserver USA server is a joint project by the U.S. Geological Survey (USGS), Microsoft Company and Hewlett-Packard (HP) Company to provide OGC WMS services to the public for free. Services currently mainly are USGS digital orthophoto quadrangles (DOQ) and digital raster graphics (DRG), which are scanned USGS maps. With the Terraserver USA server, we can add digital orthoimages or raster maps of any place in the U.S. by a simple Web link in a Web browser or a WMS client interface. Another example is the ESRI, the leader of GIS software domain. The company is a principal member of OGC and fully encompasses OGC standards and specifications. The ArcGIS data interoperability extension developed by ESRI is now supporting several OGC specifications, such as GML, WFS, WMS, WCS, *etc.*

Scalable Vector Graphics (SVG) standard is a significant Web standard recommended by the W3C. It is an emerging two-dimensional vector graphics standard. Unlike raster images, such as JPEG, GIF and PNG, SVG is a kind of vector graphic data format, so it can provide resolution-independent 2D graphics and animation for the Web. This characteristic makes SVG suitable for publishing high quality interactive maps over the Internet.

Proposed by OGC, GML provides a vendor-neutral, as well as platform-neutral, format suited for distribution of geospatial data over a network. Both GML and SVG are XML dialects that add flexibility to cartography and provide a dynamic way of presenting spatial data to a user based on his or her context and profile. They are thus suited for context sensitive mapping. Fully compatible with XML, their native capabilities can be extended in order to implement certain cartographic generalization operations. The combined SVG and GML technology can be used adequately in order to perform GIS functions and multiple representations as an alternative to traditional geometric techniques.

The efficient use of these technologies for mapping applications requires an efficient data structure that can support both data manipulation and cartographic generalization. Thus, there is an urgent need to design a Web GIS service infrastructure that can incorporate interoperable geospatial data and disseminate high quality interactive maps. However, no Web GIS service exists that meets the needs up until now. Most Web GIS applications nowadays are based on proprietary software packages, which mainly use proprietary data formats and therefore lack interoperability. In addition, many Web GIS software vendors are developing their programs which still rely on raster images to display maps. This method, to some extent, affects the quality, interactivity, and customization of cartographic maps. To conquer these problems, this thesis

mainly concerns the current limitations of Web GIS services, which are interoperability and map dissemination issues.

1.2 Research Objectives

This research will take advantage of the OGC implementation specifications, especially the GML interoperability, WFS implementation specification, as well as the W3C SVG standard. GML will be used to code and transfer geospatial data to achieve interoperability and SVG will be employed to display and disseminate maps on the Internet. WFS servers respond and provide GML data files according to users' requests. The purposes are to increase the interoperability of current Web GIS service, to improve user interfaces and the cartographic quality of Web maps, and to promote the use of standard specifications on Internet GIS. The following objectives will be accomplished in the research:

1. Review and analyze widely used Web GIS techniques according to their advantages and disadvantages.

2. Examine OGC interoperability specifications of GML and WFS to determine their usability in this research.

3. Propose a conceptual framework for the implementation of a Web GIS by using a GML/SVG data model to achieve interoperability.

4. Based on the conceptual framework, design a prototype to demonstrate the feasibility and efficiency of the framework. Evaluate and discuss the performance of the framework.

CHAPTER 2

PROBLEMS WITH CURRENT GIS TECHNIQUES

2.1 Heterogeneity and Interoperability Issues

With the development of Web GIS, there is a huge amount of geographic data stored in advanced geospatial databases. It is an ideal situation if all the available geospatial data are reusable and sharable among different organizations and applications since geographic data collection is a time consuming and costly process. It is a waste of resources if similar data are acquired again and again by different organizations. However, it is often difficult to reuse of these data in new GIS applications due to obscure semantic of data and metadata, diversity of geographic data, variety of data modeling, data encoding techniques, storage structures, access functionalities, *etc*.(Devogele *et al*. 1998).

Another problem occurs with the publicity of available geospatial data. Generally one organization does not know about the data that belong to other organizations. Even if an organization knows what data are available from another organization, several questions may also arise.

- What is the data type?
- How to get access to the data?
- Where to contact to obtain the data?
- What is the quality of the data?

- How reliable are the data?
- Which information is available?

These questions often prevent the reusability of existing data from different organizations. The usual solution is to acquire the similar or same data again by digitizing or on-site spatial data collection. The current scenario is shown in Figure 1.

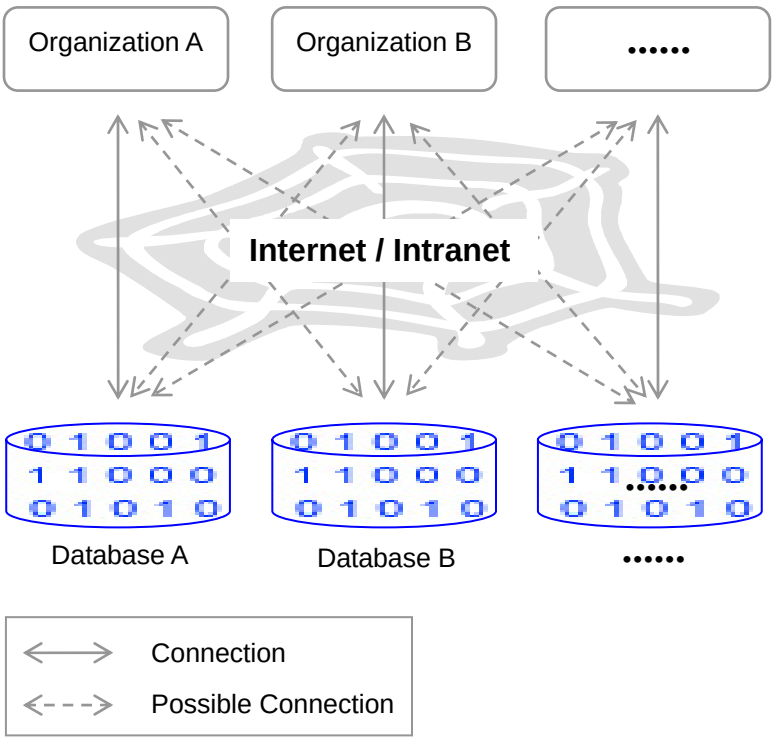


Figure 1 An Illustration of Heterogeneity in GIS Domain

Generally, when an organization A needs to set up a GIS application, it will build a new database A to store various geographic data. Organization B will also do the same thing. This is a 1-to-1 way. There are many databases and organizations, but many data users do not know the data providers and the provided data may not be compatible with the existing data of the user. In addition, it is realized that all the information needed by our society cannot be generated at one

organization. A GIS application may fetch geospatial data from different departments, such as USGS, National Resources Conversation Service (NRCS), U.S. Census Bureau, *etc.* Since several different organizations are participating in the data creation process, the data compatibility issue is of great importance and should be addressed. Otherwise two data sets from different departments will not be used and incorporated in GIS applications.

Many types of heterogeneity are due to technological differences; for example, differences in hardware, system software (*e.g.*, operating system), and communication systems. The heterogeneity issue can be solved by interoperability. In the Web GIS domain, the interoperability refers to the ability of an information system to manipulate, access, exchange, and share geospatial data from various resources for any kind of application over the Internet.

To achieve geospatial data interoperability, the issues of representation, semantics, and structure of geographic information need to be solved. Thus three aspects of geospatial data interoperability can be differentiated: syntactic, semantic, and software incompatibility. The syntactic aspect puts significant emphasis on the encoding and standardization of geographic information so that one system is capable of understanding the meaning of data from another system. This approach often establishes some standards which must be followed by different organizations. Standardized metadata and data formats can facilitate the processing of geospatial data in various environments. GML is a good example of syntactic interoperability. Although syntactic interoperability makes data easier to share and transform between different systems, the semantic gaps caused by distinct variations in conceptualizations and interpretations of geographic worlds cannot be solved by using standards. The semantic interoperability is also

essential since an ideal database should allow data to be interoperable by different clients. Besides the syntactic and semantic approach to achieve data interoperability, software is also important because interoperable software can manipulate, access, exchange and store different data sets and data formats. Although three interoperable approaches are all important, this thesis places emphasis on syntactic interoperability in Web GIS applications by using several standards.

Traditionally, GIS services were not designed to communicate with other applications and services. Instead most of them were stand-alone applications. Nowadays the OGC serves as an important organization which supports interoperability with open interfaces and protocols over the Web. The OGC has six guidelines for how geospatial information should be made available across any network, application or platform (OGC 2006):

- Geospatial information should be easy to find, regardless of its physical location.
- Once found, geospatial information should be easy to access or acquire.
- Geospatial information from different sources should be easy to integrate, combine or use in spatial analyses, even when sources contain dissimilar types of data or data with disparate feature name schemas.
- Geospatial information from different sources should be easy to register, superimpose and render for display.
- Special displays and visualizations, for specific audiences and purposes, should be easy to generate, even when many sources and types of data are involved.

- It should be easy, without expensive integration efforts, to incorporate into enterprise information systems geoprocessing resources from many software and content providers.

With so much distributed geographic information, standards are very important to the interoperability and wider use of GIS technology. Openness is also the demand for GIS data interoperability. An open standard should be created in an open, international, participatory industry process and include free, public, and open access to all interface specifications. Thus a standard established by one company, a certain organization or a government is not an open standard. Openness benefits all GIS service providers and users in our society.

In summary, heterogeneity exists in the GIS domain and interoperability is an ideal way to solve the problem. However, interoperability cannot be achieved by only a few organizations, departments, and institutions. They are supposed to cooperate to carry out the project. Fortunately the issue has been realized and the development of OGC and the standards provided by OGC are examples of the awareness among the GIS domain. Besides OGC, private companies such as ESRI, Galdos Inc. and more are all playing active roles in building interoperable technology in their products. However, there is still much to be done before GIS interoperability becomes true.

2.2 Web GIS Techniques

The Internet has been around for more than 30 years, but only since 1995 has it emerged as a potentially dominant means of global communications (Plewe 1997). The rapid development of the Internet has been closely followed by the fast progress of Internet GIS. The dominant

architecture of Internet GIS is the Client/Server model on which the WWW and most other Internet services are also based. In the early days of Internet GIS, the solutions were either implemented on the client side or on the server side. The techniques used in Distributed GIS (DGIS) or Internet GIS include Common Gateway Interface (CGI), Java Servlets, Jave Applets, and GIS plugins.

As a link between the Web server and the client browser, CGI is the earliest used technique in Internet GIS applications. HTML web pages are static and cannot be used to transfer dynamic data, thus the CGI is used as an extension of HTML to generate dynamic information for client users. Implemented on the server side, the CGI responds to the requests that are sent from the server and passes them to certain programs. The programs can be written in several programming languages, such as Perl, C, FORTRAN, Pascal and so on. After the programs get the results, the CGI will pass them to the server, which will send the user requested information back through static HTML web pages. The CGI can be used to link Web databases in the server side, so it works like a bridge between the server and the data. The role of the CGI in the Web is shown in Figure 2.

After the advent of the CGI, many new functions were possible and many GIS websites were developed based on the CGI. Researchers investigated the implementation of GIS functions by using the CGI. As early as in 1995, Crossley and Boston (1995) developed a generic map interface to query geographical information with CGI and display the resultant maps over the Internet. Huang and Lin (1999) used the CGI, *i.e*. the ESRI Map web server extension which was

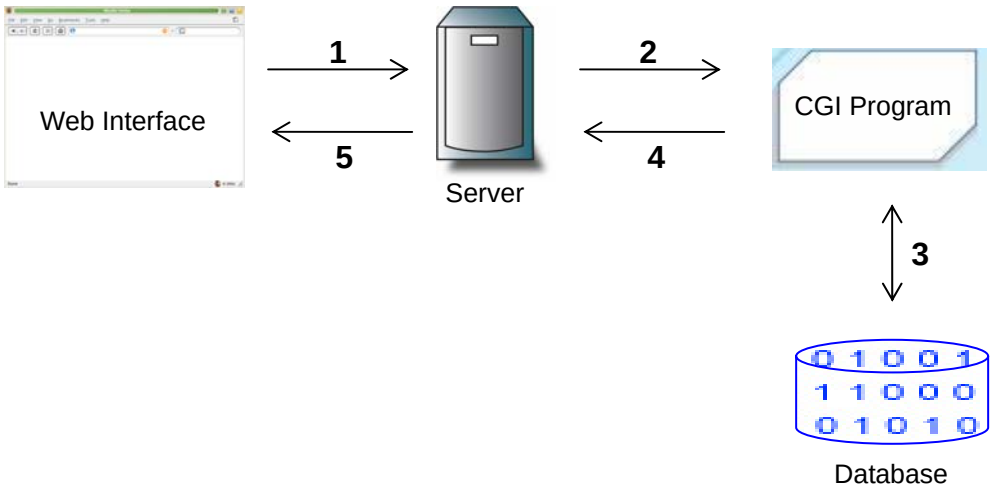


Figure 2 Role of the Common Gateway Interface (CGI)

run on the Web server, to interactively create a 3D scene and virtual reality modeling language (VRML) model from 2D spatial data. The CGI technique was employed in MapServer, a famous open source software program created by the University of Minnesota (Lime and Koormann 2005). Based on the software, Brovelli and Magni (2002) developed an archaeological Web GIS application to provide an easy and suitable Internet tool for archaeologists, museums, and laypersons so that the cultural information can be accessed for educational or tourist purposes.

The CGI technique enables the extension of HTML, however its characteristics also bring a number of performance bottlenecks in real applications. First, the processing will place too large a load on the server side if several CGI programs are operating at the same time, which limits the efficiency of applications. In addition, when a client inputs a new parameter, the CGI program will start a new operating system process so that considerable computer resources will be consumed. If many users are connecting to the server simultaneously, the connections will create

a significant computational overload. For example, Lee *et al*. (1998) described the application of CGI on GIS to distribute agriculture information through the Web. The client-server model with CGI capability was employed to extend GIS technology in agriculture. The performance of the server, however, disappointed them because of the slow responses if too many clients were requesting results from the CGI programs at the same time. Therefore, the CGI-based approach is not suitable for visualization and dynamic modeling, which are common in GIS applications.

An alternative method of using the CGI on the server side is the use of Java Servlet as the interactive interface (Lee *et al*. 1998). Java is an object-oriented programming language for the Web. It can be used to handle graphics and user interfaces and to create applications. Because of its platform independence, Java has increasingly been employed in Internet applications which may be run on any system platform. A servlet is a java program that runs on a web server and provides additional features to the server. Like the CGI, a Java servlet is also a kind of server-side program, but the processing mechanism of the servlet is different from that of the CGI.

The Java Virtual Machine (JVM) is the environmental requirement for Java servlets on the server. Instead of using multiple processes to handle separate programs and separate requests, a servlet is loaded to the single JVM once and will not be reloaded until it changes. The servlet stays resident in memory, so the speed of computation is much faster than that of the CGI. If it is necessary to reload a modified servlet, the server and application do not need to be restarted. In addition, the servlet allows you to share information between different users through sharing static or persistent information across multiple invocations of the servlet. Servlets can combine

each servlet which performs a specific task so that different servlets can talk to each other. Since the servlets are written in Java, Java Application Program Interfaces (API) or other classes can completely be called by the servlets (Huang and Worboys 2001). Therefore, the Java servlet performs much better and more efficiently than the CGI.

To overcome the shortcomings of the CGI, a Java applet-servlet architecture is explored to implement dynamic modeling and visualization, which require frequent and efficient client-server interaction, by Huang and Worbyos (2001). They demonstrate the modeling process of a hydrological model, TOPMODEL, in an Internet environment. Gong *et al*. (2003) also presented a distributed large-scale seamless image database based on the application of Java 2 Enterprise Edition (J2EE). The distributed GIS comprises four components: client, GIS servlet engine, GIS session beans and GIS Entity beans. The GIS engine is built on the Java servlet and the result of the case study proved that the servlet architecture was an efficient way to build the spatial database system.

Disadvantages of server-side solutions are primarily related to limited user interface and generally slow response (Babu 2003). The third solution for Internet GIS would be the Java applet. A Java applet is a small program written in Java and executes on the client users' web browsers rather than on the servers. Similar to the Java servlet, operating-system-independence is also a characteristic of the Java applet. The biggest difference between them is that servlets are for server side while applets are for client view. Applets can be used to provide dynamic Graphic User Interfaces (GUI) and a variety of graphical effects for the web pages. Since the program and

data are downloaded and executed in the client side, the strengths of Java applets are the friendly user interfaces, good interactive ability, fast performance and pleasant cartographic output.

Lbath and Pinet (2000) developed a CASE tool named AIGLE for the modeling and customization of Internet GIS applications. They used the Java applet technique to manage geographic data. After being downloaded from the Web server, the applet will execute in a web browser and provides the ability to download maps, display geographic maps, customize the display geographic information, pan, zoom in and zoom out, and even select and edit geographic objects. However, the applets also may overload the client side so that the clients are too "fat" to interact efficiently. If the applet program and the data are too large, they may even paralyze the client machine. In some cases like dynamic modeling, the programs and raster images are so large that it is almost impossible to implement the applet architecture without balancing the load between the server side and the client side. To overcome this problem, Huang and Lin (2002) implemented a combination of Java and CGI to design a toolkit that could be used to interactively build up virtual environments from GIS database. Since the advantages of both Java and CGI contributed to the development, the approach could balance the computational intensity on both server and client side.

Another commonly used Internet GIS solution is to install plug-ins, which provide extra functions for clients' web browsers, in the clients' computers (Polley *et al.*, 1997). The plug-ins can deal with specific kinds of content in a web page when they are downloaded from the remote server. They provide additional abilities for the browser to display and process spatial data so that the workload of the server is eased. Using this method, the clients can interact with the

server frequently and efficiently since the computation is done on the client side. The plug-in-based solution not only expedites the response speed for the users but also decreases the data volume transferred on the Internet. For example, the Autodesk Company developed the plug-in, *Mapguide Viewer*, which allows users to view, pan, zoom in and out, search features, control separate layers , and print maps within an Internet browser (Autodesk Inc. 2005). Like other software, the users have to install plug-ins before taking advantage of the extra functions. Consequently, various operating systems and web browsers cause the problem of incompatibility since plug-ins are system and browser dependent. For example, the *MapGuide Viewer* has several versions, such as ActiveX Control and Java Edition for Internet Explore in the environment of Microsoft Windows, Plug-in and Java Edition for Netscape Navigator in the environment of Microsoft Windows, and Java Edition for other browsers in Mac OS X, Solaris, or other operating systems.

CHAPTER 3

OGC SPECIFICATIONS AND WEB STANDARDS

Although most of the current solutions for spatial information display on the Web are to display map images or use web browser plug-ins, applets, or ActiveX controls as mentioned above, a more promising solution would be to use GML and SVG for geographic information storage and display since they are OGC and W3C standards and employ open specifications. Open specifications and open source software can facilitate the implementation of Web-based spatial information solutions (Anderson and Sanchez 2003). Using open specifications, Web GIS can move from proprietary systems to standard architecture so that interoperability, the core of new web service models, can be reached (Kim and Kim 2003).

3.1 Extensible Markup Language (XML)

XML is derived from the Standard Generalized Markup Language (SGML), which enables precise document specifications and precisely controls the structure of a document. As a simple, very flexible text format, XML was originally designed to cater to electronic publishing. However, it is now playing an increasingly important role in the exchange of a wide variety of data on the Internet and elsewhere. The following example can show the structure of a simple XML document.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE email SYSTEM "InternalEmail.dtd">
<email>
        <to> Kyle </to>
        <from> Lyon </from>
        <heading> Meeting </heading>
        <body> Can we meet this afternoon? </body>
</email>
```

As we see, an XML document is designed to describe data with a sequence of one or more elements, which are nested inside one another. Unlike the HyperText Markup Language (HTML), the tags in XML are not predefined and the users must invent their own tags, so XML is very flexible and extensible. It was created to structure, store, carry, and send information in well structured tags (W3Schools 2005).

An XML document is composed of two main components. The first component is made up of a header and elements. The header states some information to the parsers, which analyze the XML elements to a tree-like structure in the computer memory. This statement declares that this file is marked up in XML and the text encoding information. Below the XML header is the elements, the main part of the file. The second component is the XML validator, a Document Type Definition (DTD) or an XML Schema. With a DTD or XML schema, it is designed to be self-descriptive. A DTD or XML schema contains a list of rules which must be followed by an XML document. Once a document has been created based on a DTD or an XML schema, it will

be compared to the DTD or the schema, which will cause the validation of the document. Only a valid XML document is effective in storing and transmitting information.

There are several merits for XML:

1. XML can be used to create new languages

XML is free and extensible and it is the mother of GML, SVG, and WML (Wireless Markup Language) just to name a few. As mentioned above, the tags in XML are not predefined, so the users can extend its capability to invent other specific markup languages. For example, GML is based on XML structure and used to encode geographic information.

2. XML documents are easy to understand

XML is based on the American Standard Code for Information Interchange (ASCII), so it is easy to understand by human beings and computers. We can simply open an XML document in a Notepad program and analyze the structure.

3. XML supports a wide variety of applications

Since XML is a standard now, many applications support this format. Thus we can write an XML document just once for several different applications which need the same information. In addition, XML is a cross-platform, software and hardware independent tool for transmitting information on the Internet.

4. XML is easy to create and process

XML uses pairs of user-defined tags to describe data. The structure is so simple that programs can be easily developed to write and process XML documents. An XML document can even be written in a Notepad program.

3.2 Geography Markup Language (GML)

Based on the XML structure, the GML is “an XML encoding for the transport and storage of geographic information, including both the geometry and properties of geographic features” (OGC 2004). It is a recommended standard to encode or mark up spatial and nonspatial information in XML format by OGC. The most important characteristic of GML is the support of interoperability among different data models and feature representations with a set of basic feature tags to describe the spatial features. The advent of GML provides a mechanism for creating and sharing application schemata and a common data model.

The abstract model of geography which is developed by the OGC is the basis of GML. The model is feature-based and describes the world in terms of geographic entities. Similar to the vector model we use in conventional GIS, a feature has geometries and several properties. Points, lines and polygons are three basic components of primitive geometry elements, which can compose geometry collections. Properties have their names, types and values to describe specific features. While the initial GML specification is restricted to 2D geometry, GML 3.0 can represent real-world phenomena by using complex feature types. According to the OGC GML Implementation Specification 3.1 (p. 12), GML 3 can “represent geospatial phenomena in addition to simple 2D linear features, including features with complex, non-linear, 3D geometry, features with 2D topology, features with temporal properties, dynamic features, coverages, and observations.” In addition, GML can encode spatial reference systems, which are essential components of a geographic system. The current version of GML incorporates the main projection and geocentric reference frames in use today. Thus all of the reference systems made

by the European Petroleum Standards Group (EPSG) are supported by GML. The encoding scheme also allows users to define units and reference system parameters.

Since GML is a standard recommended by OGC, it is an effective way to transmit geographic information over the Web. When transported, the GML-coded geospatial data contain self-descriptive elements which are marked by feature tags. The receiving party will know exactly the same information as the sending party. Therefore, no information is lost and distorted during the transport process. GML is not only effective in transporting geographic information but also is an important means to store geographic information. GML provides two key elements, XLink and XPointer, to develop a multidirectional association among various features and their properties. Geographic data are naturally distributed over the surface of the Earth and the Internet. XLink and XPointer provide an ideal mechanism for distributed data and hold great promise for building complex and distributed GIS. They will be able to readily link spatial (geometric) elements to other spatial or non-spatial elements. If we use GML-encoding mechanism in all GIS databases, they will be integrated seamlessly and accessible anywhere. Combined with XLink and XPointer, GML will provide useful contributions to the development of GIS database.

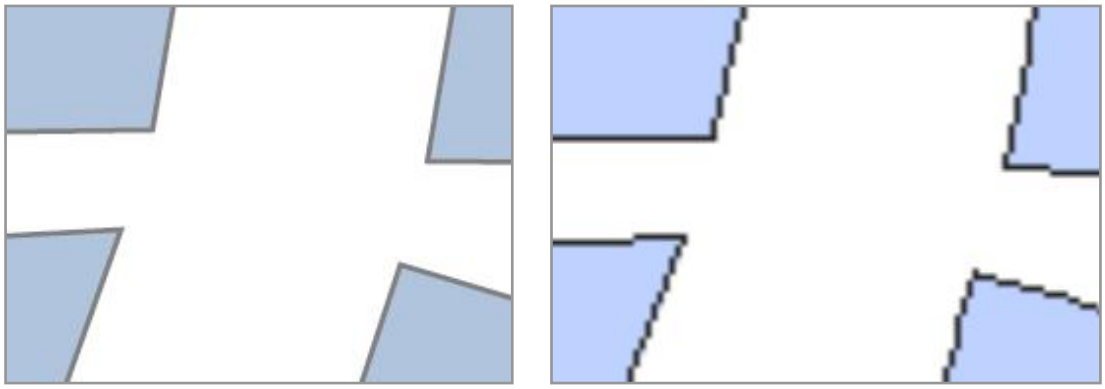
GML is derived from XML, thus it has all the characteristics of XML. For example, it is text-based and intuitive and thus can be easily read and understood by human beings. A GML document has a strict hierarchy structure to clearly identify the relationships among the document's content elements. However, GML has its own unique characteristics that make it different. GML documents are written by complying with GML schemata, which are used to

validate the documents and composed by the OGC. Before GML 2.0, the validator of GML documents was Document Type Definition (DTD). The GML schemata are developed in GML 2.0 and later versions because GML schemata have several advantages over the DTDs. GML is not a presentation language. With the separation of structure from presentation, GML is used to focus on the structure of the document without considering the presentation styles. Thus the visualization of GML data content can be customized with different styles. GML also Permit the easy integration of spatial and non-spatial data, especially for cases in which the non-spatial data are XML-encoded. GML provides a set of common geographic modeling objects to enable interoperability of independently developed applications.

3.3 Scalable Vector Graphics (SVG)

SVG also uses XML syntax to define resolution-independent 2D graphics and animation for the Web. It is defined as “a new breed of Web-enabled graphics that will change the way the users edit, transit, interpret and display maps on interoperable desktops” (Gould and Ribalaygua 1999). Compared with raster image formats such as GIF and JPEG, a vector graph has several advantages. First, vector images can be easily zoomed in and resized without any loss of image quality. As shown in Figure 3, an SVG image has much better quality than a pixelated JPEG image when they are zoomed in to a high level.

Second, the volume of a vector file describing geographic phenomena is much smaller than that of a raster file showing the same phenomena. Therefore, vector-based maps are more appropriate for Web GIS applications than raster-based maps. Currently SVG and Macromedia



(a) SVG Image (b) JPEG Image

Figure 3 Comparison between an SVG Image and a JPEG Image

Flash format are two widely used vector graphics for the Internet. Apparently the SVG format has an advantage because it is a standard format recommended by the W3C and complies with other standards, like Cascading Style Sheets (CSS), eXtensible Stylesheet Language Transformation (XSLT), XLink, XPointer, *etc*. However, Flash format relies on proprietary technology that is not open source. Since September 2001 when the SVG specification was accepted by the W3C, both commercial and non-profit organizations have started implementing it. Although plug-ins must be installed in order to display SVG files now, all major web browsers will support SVG in the very near future without requiring additional plug-ins. In fact, the Mozilla Firefox has already incorporated the SVG support in its Version 1.5. In addition, SVG can be extended to any suitable device of any size, like PDAs or mobile phones, without loss of information. The Mobile SVG version 1.1, another W3C recommendation since January 2001, introduces two subsets of standard SVG: SVG Tiny for mobile phones and SVG Basic for PDAs (Brinkhoff 2003).

Another advantage of SVG is the interactivity. Complex interactive mapping applications can be built by controlling SVG with Javascript, a standard scripting language designed for the Web. Using Javascript, programmers can open SVG files, retrieve elements, change object attributes, add or delete features. The scalability, another strong point of SVG, also can be managed by Javascript so that users can zoom in, zoom out or pan the graphs as per their needs. SVG provides several filter and gradient functions that can be used to develop high quality vector maps. For example, we can add terrain shading in a map based on an internal DEM with the help of SVG's filter functions. Furthermore, every element and every attribute can be animated in SVG files, so animated and dynamic maps will be more easily incorporated in Web GIS, especially for Temporal GIS which deals with position, elevation and time simultaneously.

Besides the difference of data format between SVG and GIF or JPEG, the most important characteristic of SVG is the use accessibility to search the map contents. Although we can see text in a GIF or JPEG picture, the computer does not recognize the content of the text. As a result we are unable to search a specific word in a raster image. Conversely, SVG is based on XML and conforms to all the standards and techniques of XML, so the plain text in SVG files can be searched and edited by users. Query also is available in SVG maps. XLink, which is available in SVG, provides hyperlinks to many other resources as well as raster and vector graphics. The combination of these characteristics of SVG means that it will be an essential part for the future development of Internet GIS.

In summary, the benefits of SVG are listed as follows:

1. Better visibility on zooming

2. Small volume size
3. A standard recommended by W3C
4. Adoption by PDAs and mobile phones
5. Interactivity
6. Animation and graphic filter effects
7. Easy editing and text-searching ability

The SVG technique is applied widely to produce high quality interactive vector Web maps. Carrara *et al*. (2003) designed a software prototype BANCO (Browsing Adaptive Network for Changing user Operativity) to enable users to interact with remote sensing, spatial and multimedia data over the Internet. Relying on SVG, BANCO proposes a novel architecture so that the client users can customize their environment based on their needs. BANCO is a good example of applying SVG to extend the feasibility Web sites with raster and vector images without resorting to any proprietary and expensive commercial programs. Köbben (2003) presented the applications of SVG and GML in the TOP10NL project, which was officially titled "the second generation TOPvector data-model project" in Netherlands. The scalability, interactivity, and animation of SVG were studied and implemented in the project, which proved the effectiveness of SVG to disseminate Internet maps. Analyzing the adaptive zooming in Web cartography, Cecconi and Galanda (2002) argued that the best data format to meet the interactive and dynamic requirements of Web mapping is the W3C standard SVG. The geographic applications of SVG even have been extended to mobile devices, like Personal Digital Assistants (PDA) or mobile phones, which support the SVG standard (Brinkhoff 2003). In the GiModig

project (Harrie *et al*. 2002), which addressed real-time generalization of spatial data suited to mobile devices with different display resolutions, the authors emphasized the need for research on real-time generalization and small-display cartography as well as methods to transfer vector-formatted data to wireless devices using the standards GML and SVG.

As a new technology, SVG inevitably has some limitations, although its advantages far outweigh its shortcomings. First, the SVG file size may be very large when it describes a complex vector map because it is a text-based file. However, the file size will decrease dramatically if a compression algorithm is used. For instance, an original SVG file of roads in Athens, GA can be as large as 1.52 Megabytes, while the compressed file in SVGZ format is minimized to only 222 Kilobytes. Furthermore, an SVG plug-in or view may use the CPU extensively when it is working on rendering complex vector maps. SVG can produce animated, dynamic and interactive maps, thus it consumes considerable system resources when maps are displayed. With the rapid development of computer equipment, the CPU limitation becomes trivial. Although we have to install SVG plug-ins to view the data for most web browsers right now, the limitation will disappear as the SVG gains wider acceptance, which is coming soon.

3.4 Web Feature Service (WFS)

Web Feature Service (WFS) is one of the major services described by OGC for sharing geographic information. Using GML for data exchange, the OpenGIS Web Feature Service Interface Standard (WFS) is an interface allowing requests for geographical features across the web to be highly interoperable. It is one of the most powerful data services of OGC web service.

Based on the HTTP protocol, WFS allows a client to retrieve and manipulate geospatial data which are encoded in GML. Two functions of WFS are important for Web-based GIS applications and services. First, a data provider can establish a WFS server with a spatial database to provide Internet spatial data services. Second, it can implement interoperability between heterogeneous systems so that data operations, such as data query, browse, update, *etc*., can be achieved between different GIS software.

To support transaction and query processing, WFS defined five operations as follows:

GetCapabilities

The GetCapabilities operation describes services which can be provided by a WFS server using XML. This operation must be implemented by each WFS server. Specifically, it must indicate which feature types it can serve and what operations are supported on each feature type. GetCapabilities operation is used as a WFS request to retrieve service capabilities which are encoded in XML and returned to the client. For example, if the following URL is sent to a WFS server, an XML document showing feature types and supported operations will be retrieved. Taking the following result as an example, the supported operations are included in the <Capability></Capability> tag pairs, while the feature types are listed in the <FeatureTypeList></FeatureTypeList> tag pairs. As indicated in the XML document, three operations (GetCapabilities, DescribeFeatureType, and GetFeature) are supported by the WFS server on server 128.192.49.105. The feature types include streets, rivers, property, buildings and so on.

Request:

http://128.192.49.105/cgi-bin/mapserv.exe?map=/ms4w/apps/wfs_ser/test.map&SERVICE=WFS&VERSION=1.0.0&service=WFS**&request=getcapabilities**

Result (*fragment*)*:*

```
<WFS_Capabilities
   version="1.0.0"
   updateSequence="0"
   xmlns="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wfs
http://ogc.dmsolutions.ca/wfs/1.0.0/WFS-capabilities.xsd">

<Service>
  <Name>MapServer WFS</Name>
  <Title>Liang's WFS Demo Server</Title>
  <OnlineResource>http://localhost/cgi-bin/mapserv.exe?</OnlineResource>
</Service>

<Capability>
  <Request>
     <GetCapabilities>
        ……
     </GetCapabilities>
     <DescribeFeatureType>
        ……
     </DescribeFeatureType>
     <GetFeature>
        ……
     </GetFeature>
  </Request>
</Capability>

<FeatureTypeList>
  <Operations>
```

```
    <Query/>
  </Operations>
    <FeatureType>
        <Name>streets</Name>
        <Title>Streets</Title>
        <SRS>EPSG:26917</SRS>
        <LatLongBoundingBox minx="-76.8644" miny="82.6589" maxx="-74.6985" maxy="82.8993" />
    </FeatureType>
    <FeatureType>
        <Name>airports</Name>
        <Title>Airports</Title>
        <SRS>EPSG:26917</SRS>
        <LatLongBoundingBox minx="277460" miny="3.75889e+006" maxx="285015"
maxy="3.75899e+006" />
    </FeatureType>
    <FeatureType>
        <Name>rivers</Name>
        <Title>Rivers</Title>
        <SRS>EPSG:26917</SRS>
        <LatLongBoundingBox minx="269272" miny="3.74789e+006" maxx="290439"
maxy="3.76871e+006" />
    </FeatureType>
    <FeatureType>
        <Name>boundary</Name>
        <Title>Boundary</Title>
        <SRS>EPSG:26917</SRS>
        <LatLongBoundingBox minx="265555" miny="3.74764e+006" maxx="292818"
maxy="3.76917e+006" />
    </FeatureType>
    <FeatureType>
        <Name>buildings</Name>
        <Title>UGA Buildings</Title>
        <SRS>EPSG:26917</SRS>
        <LatLongBoundingBox minx="279143" miny="3.756e+006" maxx="282680"
maxy="3.76048e+006" />
    </FeatureType>
    <FeatureType>
        <Name>property</Name>
        <Title>UGA Property</Title>
        <SRS>EPSG:26917</SRS>
        <LatLongBoundingBox minx="279111" miny="3.75597e+006" maxx="282700"
```

```
maxy="3.76069e+006" />
    </FeatureType>
    <FeatureType>
        <Name>roads</Name>
        <Title>UGA Roads</Title>
        <SRS>EPSG:26917</SRS>
        <LatLongBoundingBox minx="279631" miny="3.75603e+006" maxx="282707"
maxy="3.76046e+006" />
    </FeatureType>
    <FeatureType>
        <Name>routes</Name>
        <Title>My Routes</Title>
        <SRS>EPSG:26917</SRS>
        <LatLongBoundingBox minx="280168" miny="3.75636e+006" maxx="280923"
maxy="3.75905e+006" />
    </FeatureType>
</FeatureTypeList>

……
</WFS_Capabilities>
```

DescribeFeatureType

A WFS server must be able, upon request, to describe the structure of any feature type it can service. Upon request, a schema description of feature type serviced by a WFS implementation will be generated. The schema descriptions define how feature instances to be encoded on input and how feature instances will be generated on output. A DescribeFeatureType request may contain the TypeName elements which encode the names of feature types to be described. If no TypeName element is specified, all feature types that a WFS server can service will be interpreted.

GetFeature

A WFS server must be able to service a request to retrieve feature instances. In addition, the client should be able to specify which feature properties to fetch and constrain the query spatially

and non-spatially. A GetFeature request is processed by a WFS server and a resultant GML document will be returned to the client.

A GetFeature request must contain version, service, and TypeName parameters. Thus a simple example of GetFeature request is:

http://128.192.49.105/cgi-bin/mapserv.exe?map=/ms4w/apps/wfs_ser/test.map&SERVICE=WFS&VERSION=1.0.0&service=WFS&**Request=GetFeature&TypeName=rivers**

The URL indicates the WFS server address and necessary parameters for retrieving corresponding feature, which is the “rivers” layer, in GML format. Besides the TypeName variable, there are several parameters such as Query, Filter, PropertyName and so on. The GetFeature request can contain one or more Query elements, each of which contain the description of a query. The Filter parameter can be used to define constraints on a query. Both spatial and non-spatial constraints can be specified to refine query results. The constraints are encoded in XML and follow the OGC Filter Encoding Specifications (OGC 2005b). If the Filter element is not specified, the request is unconstrained and all feature instances will be retrieved according to the specified TypeName variable. The PropertyName element is used to enumerate the feature properties that should be included in the response to the GetFeature request. To determine the properties of a feature, a DescribeFeatureType request described above can be applied before sending a GetFeature request. After obtaining a GML application schema from DescribeFeatureType operation, the client will be able to select the properties to be fetched. If there is no PropertyName parameter in the request, all feature properties will be selected in the result.

Corresponding to the GetFeature request URL above, a fragment of the GML document is shown below:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<wfs:FeatureCollection xmlns:myns="http://www.ttt.org/myns" xmlns:wfs="http://www.opengis.net/wfs"
xmlns:gml="http://www.opengis.net/gml" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wfs http://ogc.dmsolutions.ca/wfs/1.0.0/WFS-basic.xsd
http://www.ttt.org/myns
http://localhost/cgi-bin/mapserv.exe?SERVICE=WFS&amp;VERSION=1.0.0&amp;REQUEST=Describ
eFeatureType&amp;TYPENAME=rivers&amp;OUTPUTFORMAT=XMLSCHEMA">
    <gml:boundedBy>
        <gml:Box srsName="EPSG:26917">
            <gml:coordinates>
                269271.514693,3747887.804374 290439.022526,3768713.339878
            </gml:coordinates>
        </gml:Box>
    </gml:boundedBy>
    <gml:featureMember>
        <myns:rivers>
            <gml:boundedBy>
                <gml:Box srsName="EPSG:26917">
                    <gml:coordinates>
                        280082.919803,3765335.638495 280534.839664,3768713.339878
                    </gml:coordinates>
                </gml:Box>
            </gml:boundedBy>
            <myns:msGeometry>
                <gml:LineString srsName="EPSG:26917">
                    <gml:coordinates>
280137.706818,3768713.339878 280400.104320,3768194.734560 280248.505579,3767854.516459
280151.856907,3767462.015969 280303.779960,3767298.844928 280317.692907,3767134.924940
280159.848267,3766984.597615 280082.919803,3766699.365511 280187.049954,3766668.903955
280208.234669,3766574.284768 280165.733413,3766579.213956 280157.481120,3766531.661029
280369.656049,3766353.750281 280245.776909,3765899.052012 280441.532530,3765824.223574
280492.138023,3765724.236633 280379.437257,3765641.986084 280279.293034,3765466.117998
280345.767497,3765453.334621 280339.596242,3765384.140641 280412.863486,3765335.638495
280534.839664,3765351.652876
                    </gml:coordinates>
                </gml:LineString>
```

```
            </myns:msGeometry>
            <myns:FNODE_>11</myns:FNODE_>
            <myns:TNODE_>21</myns:TNODE_>
            <myns:LPOLY_>2</myns:LPOLY_>
            <myns:RPOLY_>2</myns:RPOLY_>
            <myns:LENGTH>0.00549</myns:LENGTH>
            <myns:RIVER059_>17</myns:RIVER059_>
            <myns:RIVER059_I>1067318</myns:RIVER059_I>
            <myns:FEATURE>Artificial Path</myns:FEATURE>
            <myns:FEATURE_ID/>
            <myns:FEATURE_TY>Lake/Pond</myns:FEATURE_TY>
            <myns:STATE_FIPS>13</myns:STATE_FIPS>
            <myns:COUNTY_FIP>059</myns:COUNTY_FIP>
            <myns:HUC>3070101</myns:HUC>
            <myns:NAME>Sandy Creek</myns:NAME>
            <myns:CLASS>Major</myns:CLASS>
            <myns:SOURCE>USGS 7.5 min. Quad</myns:SOURCE>
            <myns:EDITOR>ITOS</myns:EDITOR>
            <myns:SOURCE_DAT>19980331</myns:SOURCE_DAT>
            <myns:FEAT_MOD_D>19980615</myns:FEAT_MOD_D>
            <myns:InLine_Oid>0</myns:InLine_Oid>
        </myns:rivers>
    </gml:featureMember>
    ……
</wfs:FeatureCollection>
```

Transaction

A WFS server may be able to service transaction requests. A Transaction request is composed of operations that modify features (*e.g.* create, update, and delete operations on geographic features). A WFS service may process Transaction operations directly. It also can indirectly translate the operation into the language of a target database which the WFS server is connected to so that the database can execute the transaction. After a transaction has been implemented, the

WFS server will respond to the client with an XML document indicating the completion status of the transaction.

For a WFS implementation, the Transaction operation is not required to be supported. For example, the example WFS server (http://128.192.49.105/cgi-bin/mapserv.exe?map=/ms4w/apps/wfs_ser/test.map) mentioned above does not support the Transaction operation. The GetCapabilities operation should advertise the Transaction operation if it is implemented in a WFS server.

A Transaction operation may contain zero or more Insert, Update, or Delete parameters that request the server to create, modify or kill feature instances. In response to a transaction operation, a WFS server will notify the termination status of the transaction in an XML file. For example, if the Insert operations are sent as a Transaction request, the WFS server must report the feature identifiers of all newly created features. The WFS server will also show failure information in the response if the Insert operations fail to execute.

LockFeature

A WFS server may be able to process a lock request on one or more instances of a feature type for the duration of a transaction. Because of the inherent statelessness of web connections, the success of transactions cannot be guaranteed. For example, after the client retrieves a feature instance, the feature is then modified on the client side and submitted back to the database via a Transaction request for update. During this process, seralizability may be lost if another client fetches the same feature and updates it in the database. With a feature locked, the access to it will

be denied if another client tries to retrieve and update it. The mutually exclusive manner ensures serializability which guarantees that while a data item is being edited on the client side, another client will not come along and modify the same data item. The purpose of the LockFeature operation is to expose a long-term feature locking mechanism to ensure consistency. The long term here means that network latency would make feature locks last relatively longer than common native database locks. This ensures that serializable transactions are supported. Like the Transaction operation, the LockFeature operation is also optional for a WFS implementation to conform to this specification.

Based on the interface descriptions above, two classes of WFS servers have been defined, Basic WFS and Transaction WFS. A basic WFS implements the GetCapabilities, DescribeFeatureType and GetFeature interfaces. This would be considered a read-only WFS server. A transaction WFS server would support all operations of a basic WFS server and in addition it would implement the Transaction operations. Optionally, a transaction WFS could implement the LockFeature operation.

3.5 Web Map Service (WMS)

A web map service (WMS) produces maps of geo-referenced data according to clients' requests. It is one of the OGC web services and an important Web GIS visualization system. The WMS is a simple, yet powerful interface to web mapping services. A map is not the geographic data itself. It is created from raw spatial, vector or coverage data. Generally, a WMS server

produces maps rendered in graphic formats such as JPEG, GIF, and PNG. A few WMS servers also create SVG high quality vector maps.

OGC WMS specification defines three main operations to provide GIS mapping services. The operations include GetCapabilities, GetMap, and GetFeatureInfo. The first two operations are required to be implemented by a WMS to produce a map while the GetFeatureInfo is an optional operation (Sayar *et al*. 2005).

GetCapabilities

Like WFS's GetCapabilities operation described before, the GetCapabilities operation of a WMS server returns service-level metadata, which is a description of the service's information content and acceptable request operations. The server can advertise accessible layers, supported output projections, available output data formats and other general service information which can be obtained by WMS clients with the GetCapabilities request. After a WMS server receives the GetCapabilities request, it parses and processes it. If no error occurs during the process, an XML file describing the service metadata will be sent to the WMS client. If the WMS server encounters any problem, an exception message also will be sent to the client in XML format. For example, the following response corresponds to the GetCapabilities request sent to the WMS server at http://www2.demis.nl/mapserver by using the following URL.

Request:

http://www2.demis.nl/mapserver/request.asp?Service=WMS&Version=1.1.0&**Request=GetCapabilities**

Result (*fragment*)*:*

```
<?xml version="1.0"?>
<!DOCTYPE WMT_MS_Capabilities SYSTEM "http://www2.demis.nl/wms/capabilities_1_1_0.dtd" >
<WMT_MS_Capabilities version="1.1.0">
    <Service>
        <Name>OGC:WMS</Name>
        <Title>World Map</Title>
        <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
xlink:href="http://www2.demis.nl"/>
        <ContactInformation>
        ……
        </ContactInformation>
        <Fees>none</Fees>
        <AccessConstraints>none</AccessConstraints>
    </Service>
    <Capability>
        <Request>
            <GetCapabilities>
                .....
            </GetCapabilities>
            <GetMap>
                <Format>image/png</Format>
                <Format>image/jpeg</Format>
                <Format>image/gif</Format>
                <Format>image/bmp</Format>
                <Format>image/swf</Format>
            </GetMap>
            <GetFeatureInfo>
                <Format>text/xml</Format>
                <Format>text/plain</Format>
                <Format>text/html</Format>
                <Format>text/swf</Format>
                <Format>application/vnd.ogc.gml</Format>
            </GetFeatureInfo>
        </Request>
        <Layer>
            <Title>World Map</Title>
            <SRS>EPSG:4326</SRS>
            <LatLonBoundingBox minx="-180" miny="-90" maxx="180" maxy="90"/>
            <BoundingBox SRS="EPSG:4326" minx="-184" miny="-90" maxx="180" maxy="90"/>
```

```
            <Layer queryable="0" opaque="1">
                <Name>Bathymetry</Name>
                <Title>Bathymetry</Title>
                <BoundingBox SRS="EPSG:4326" minx="-180" miny="-90" maxx="180"
maxy="90"/>
            </Layer>
            <Layer queryable="1" opaque="0">
                <Name>Countries</Name>
                <Title>Countries</Title>
                <BoundingBox SRS="EPSG:4326" minx="-184" miny="-90" maxx="180"
maxy="85"/>
            </Layer>
          ……
        </Layer>
    </Capability>
</WMT_MS_Capabilities>
```

As shown in the <Service></Service> tag pairs, this WMS server has no access constraints. The GetCapabilities, GetMap and GetFeatureInfo operations are all supported and available sub-layers in the World Map layer include Bathymetry, Countries, and so on. The output formats of the GetMap opration include PNG, JPEG, GIF, BMP, and SWF.

GetMap

The GetMap request is designed to retrieve a map, which is either a pictorial format or a set of graphic elements. By referring to the GetCapabilities operation, a client will be able to know available layers and output formats. Then a GetMap request may be sent to the WMS server. Upon receiving the request, the server shall either satisfy the request by returning the result as an image defined in the GetMap request or throw an exception if an error occurs.

The GetMap request is typically encoded as a URL that is invoked on the WMS using the HTTP GET operation. The URL is required to provide nine parameters: version, request, layers,

styles, srs, bbox, width, height, and format. The detailed description of each variable is listed in table 1.

The WMS server first parses the necessary parameters which are encoded in the URL and determines which layers are requested, in which bounding box, in which form, and so forth. The

Table 1 Description of WMS Request Parameters

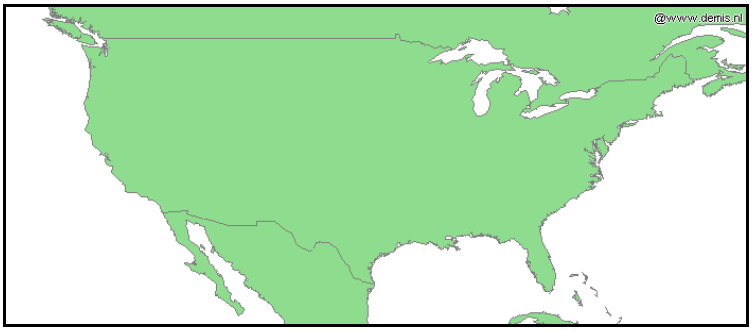
Request Parameter	Description
Version	WMS Version
Request	Request Name (GetMap)
Layers	Comma-separated list of one or more map layers
Styles	Comma-separated list of one rendering style per requested layer
SRS	Spatial Reference System
BBOX	Bounding box corners (lower left, upper right) in SRS units
Width	Width in pixels of map picture
Height	Height in pixels of map picture
Format	Output format of map

server then generates requested maps which are returned to the client. The follow example shows a sample URL request and corresponding generated map from the WMS server at http://www2.demis.nl/mapserver/request.asp?Service=WMS.

Request:

http://www2.demis.nl/mapserver/request.asp?Service=WMS&Version=1.1.0&**Request=GetMap&Style=&BBox=-132,22,-62,52&SRS=EPSG:4326&Width=700&Height=300&Layers=Countries,Borders,Coastlines&Format=image/gif**

Result:



GetFeatureInfo

The GetFeatureInfo is not a required operation to create a map and it is an optional WMS service. This operation allows us to obtain additional information from the WMS. Even if the GetFeatureInfo operation is supported on a WMS server, it is only applicable to layers for which the attribute ‘queryable’ is set to ‘1’(true). If a client issues a GetFeatureInfo request to other layers, a service exception response will be returned.

The purpose of GetFeatureInfo is to provide clients of a WMS with more information about features in the pictures of maps which were acquired by the GetMap operation. It works as follows: a user sees the map retrieved from the GetMap request and chooses a point of interest. Because the WMS protocol is stateless, the user needs to supply (X, Y) Cartesian coordinates and the layers of interest, original GetMap request parameters (all but Version and Request), and the requested information format (HTML, GML or ASCII). With the spatial information and the X, Y position that the user defines, the WMS may provide additional attribute information about that position. The following URL can be sent to the WMS server used before to obtain

information at the image position (100, 100). The position is defined with the upper left corner as the original point (0, 0). Figure 4 shows the appearance of the HTML result in a web browser.

Request:

```
http://www2.demis.nl/mapserver/request.asp?Service=WMS&Version=1.1.0&Request=GetFeatureInfo&Style=&BBox=-132,22,-62,52&SRS=EPSG:4326&Width=700&Height=300&Format=image/gif&x=100&y=100&Query_layers=Countries
```

Result(*html*)*:*

```
<html>
    <head>
        <title>Query results</title>
    </head>
    <body>
        <table border="1" width="100%">
            <tbody>
                <tr>
                    <th width="25%"> Layer </th>
                    <th width="15%"> ID </th>
                    <th width="25%"> Description </th>
                    <th width="25%"> Value </th>
                </tr>
                <tr>
                    <td>Countries</td>
                    <td>US</td>
                    <td>United States</td>
                    <td>  </td>
                </tr>
            </tbody>
        </table>
    </body>
</html>
```

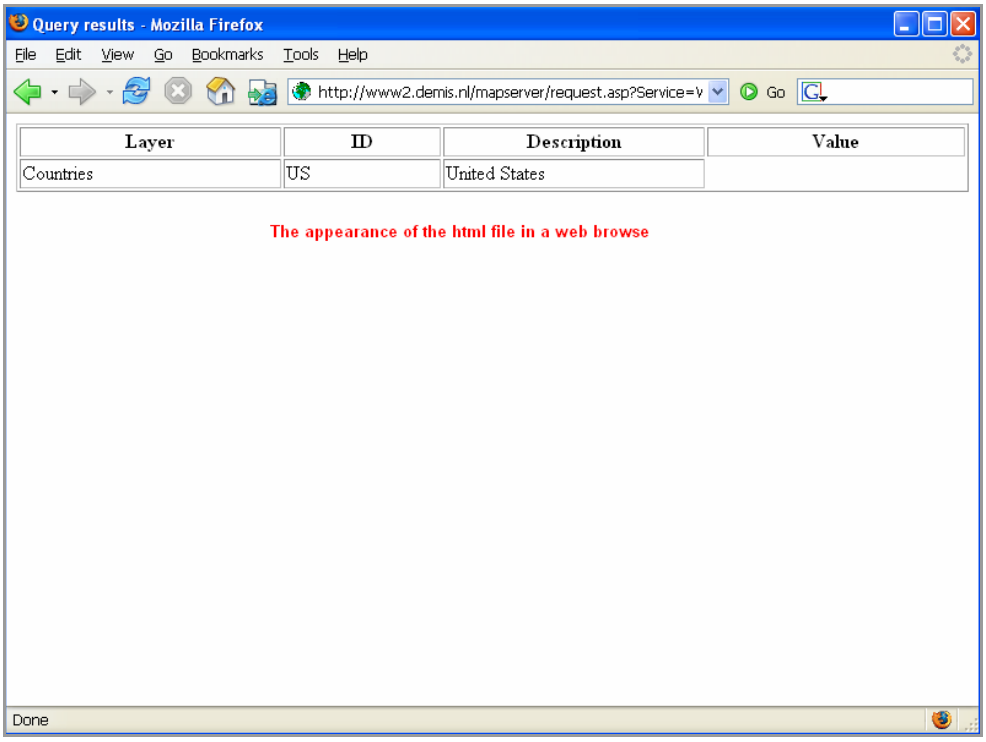


Figure 4 The Appearance of the HTML Result in a Web Browser

CHAPTER 4

METHODOLOGY

4.1 Conceptual Framework

Basically, we can distinguish three approaches to visualize and manipulate geo-spatial data within wired and wireless Internet environments: conversion of all data in the HTML/SVG file directly, on-demand dynamic conversions from the source-data file to SVG/HTML using a script, and building a GML-based interoperable spatial database for multi-purpose uses. However, the first two approaches suffer from several drawbacks. For example, the format of proprietary GIS data may not be compatible with SVG if the data are converted directly. Thus the problem of interoperability also may exist when the first method is used. For the second method, only small geographic areas can be generated concerning the work load. Therefore, building a GML-based interoperable spatial database for multi-purpose is the most elaborated approach. Using the database, the presentation and content are separated and large datasets can be handled. The database is flexible and manageable, so mobile applications also can take advantage of the data.

Nowadays, ESRI shapefile format is the most popular GIS vector data format, so layers in ESRI shapefile format are important data resources for GML database construction. Besides shapefile layers, data in other proprietary formats also may be available. When new data are collected, they can be recorded in GML format directly and stored in the database. Since GML is a kind of XML, the database can be managed by XML databases, which store, query, retrieve,

update, index, and manipulate XML files. There are three types of databases which can be adopted. One way is to use existing database systems to manage XML. The XML documents are stored in relational databases which describe hierarchical, tree-type structures with relations. This method is called an XML-enabled database. Another way is to manage XML files natively, so this type is named native XML databases. The last method is to employ a GML-enabled spatial database, which stores spatial information in special data formats – Well-Known Text (WKT) and Well-Known Binary (WKB). WKT and WKB are OGC defined representations for geometry (OGC 1999). While WKT is designed to exchange geometry data in ASCII form, WKB is in binary form. They are standards to store spatial information in relational databases. Like XML-enabled database, a GML-enabled spatial database is a relational database. However, it stores spatial geometry information in common relational tables by using WKT or WKB instead of hierarchical, tree-type structures. What makes a GML-enabled spatial database special is that it supports GML conversion at the SQL level. One good example of such a spatial database is PostgreSQL with PostGIS extension, which was utilized to implement the GML/SVG-based Web GIS architecture in this thesis.

The GML-based interoperable geographical database is the foundation of Web GIS. Not until the database is ready to use can the framework be implemented. Figure 5 shows the flow diagram of the process taking place in Web GIS for personal computer (PC) users. On the client side, a user is presented with an interface showing the input queries and map area. The user can type query expressions and send them to the server. According to the user's requests, the WFS server will communicate with the GML-enabled spatial database while the WMS server will send

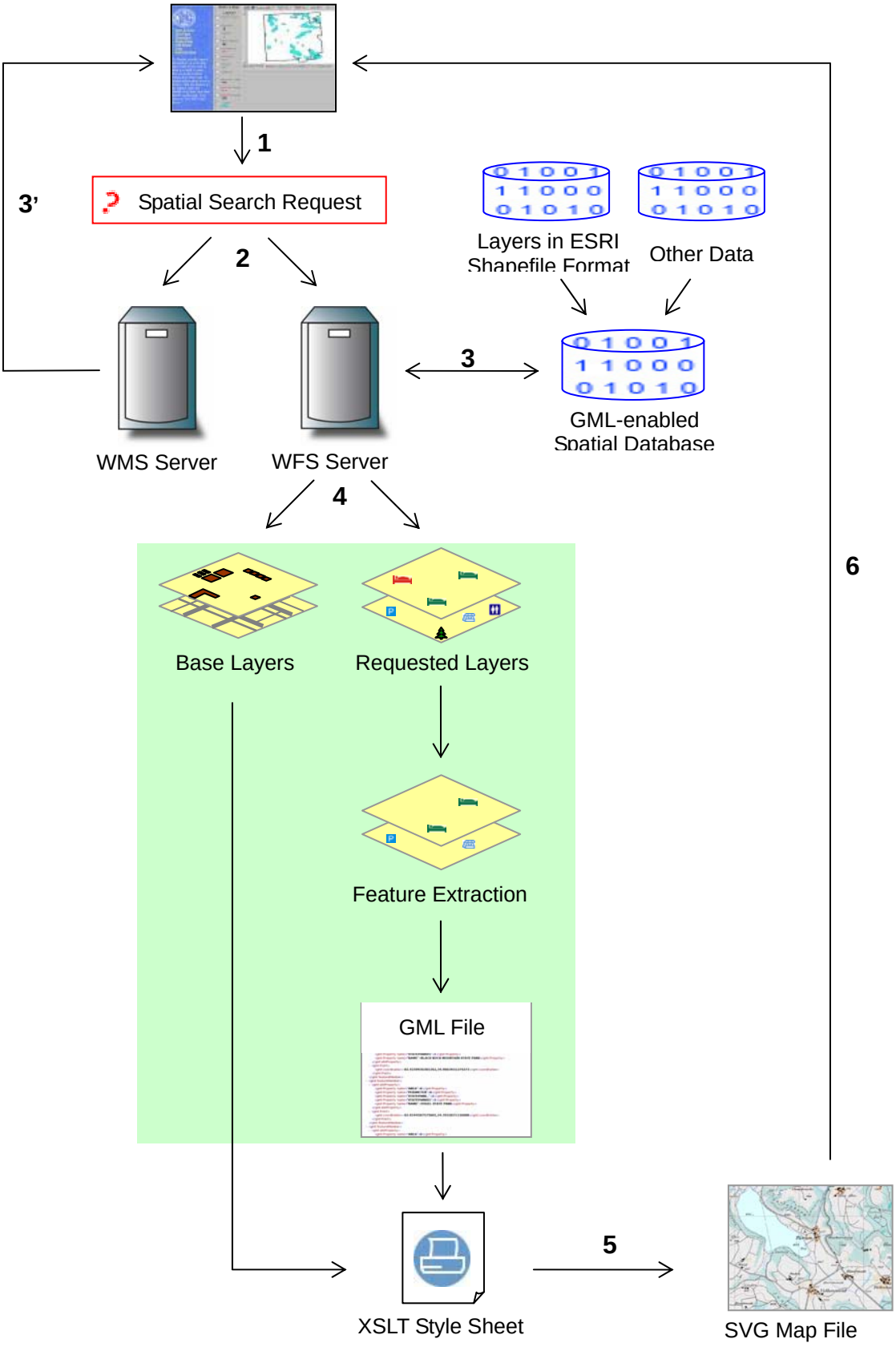


Figure 5 The Flow Diagram in Interoperable Web GIS

corresponding raster layers. For the WFS server, the necessary base layers will be retrieved to show the boundaries, roads *etc*. in the user's area of interest. Other requested layers, *e.g*. river and infrastructure layers, also are sent to the server for further analysis. The requested layers will then be analyzed and filtered to extract features by the WFS server according to search texts obtained from the Web interface. The search texts are converted to XML encoding which follows the OGC Filter Encoding Specification (OGC 2005b) so that the server can understand the search conditions. In the following step, a GML document corresponding to the user's request will be generated. Then both of the GML base layers and generated GML layers will be converted to an SVG file containing several layers for presentation by using an XSLT style sheet. XSL (eXtensible Stylesheet Language) is a specification developed by W3C for applying formatting to XML documents in a standard way. A stylesheet is used to describe how the content of a given structured document should be presented. XSLT stands for eXtensible Stylesheet Language Transformations. It is an important part of XSL and is used to transform XML documents into other XML formats, like SVG. The process of transforming GML files to SVG maps is shown in Figure 6. XSLT style sheet also uses XML syntax as it is an XML file as well. The style sheet contains several rules which tell the XSLT process how to visualize each feature element in the SVG file. The translation behind text GML to SVG is a kind of mapping between GML tags and SVG definitions. Table 2 illustrates some important corresponding tags between GML and SVG.

As soon as the SVG map with user requested information is generated at the server side, it will be transferred back to the client side for display as a map. All of the described procedures above

take place on the server side of the system. The client side is only used to present the user with input forms and graphical output in SVG generated upon his or her requests.

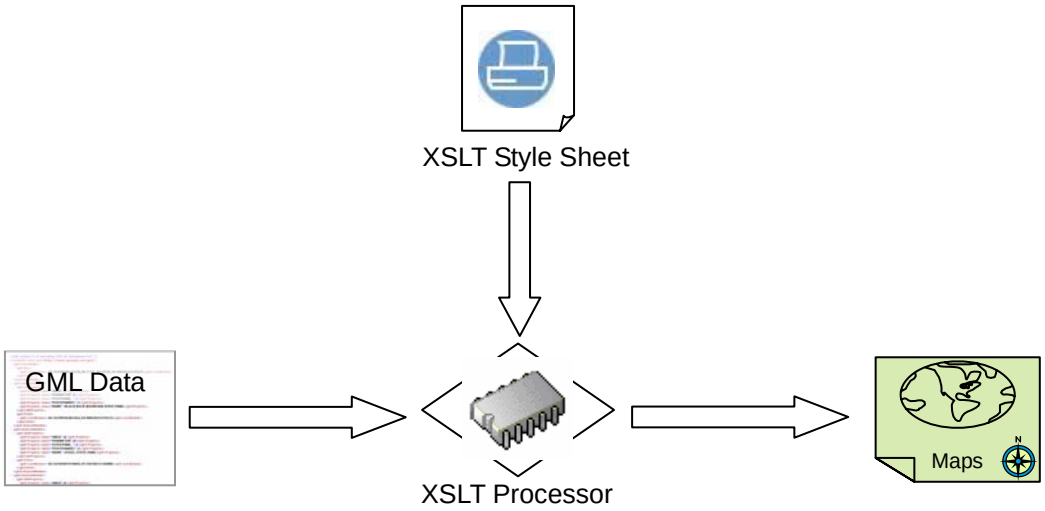


Figure 6 The SVG Map Making Process

Table 2 Corresponding Tags between GML and SVG

GML Geometry Element	SVG Element
Box	Rect
Point	Rect, Circle, or Path
LineString	Path, Polyline
Polygon	Path, Polygon

To implement the interoperability and visualization of the spatial data in the distributed network, the conceptual framework applies a four-layer structure on web services. The four layers include the client layer, web application server layer, WFS server layer, and the database

layer. Figure 7 demonstrates the structure of the four layers. The GML-enabled database stores the data from different GIS platforms and has the ability to export GML format data which conform to the defined GML schemata by the data exchanging components. The web feature

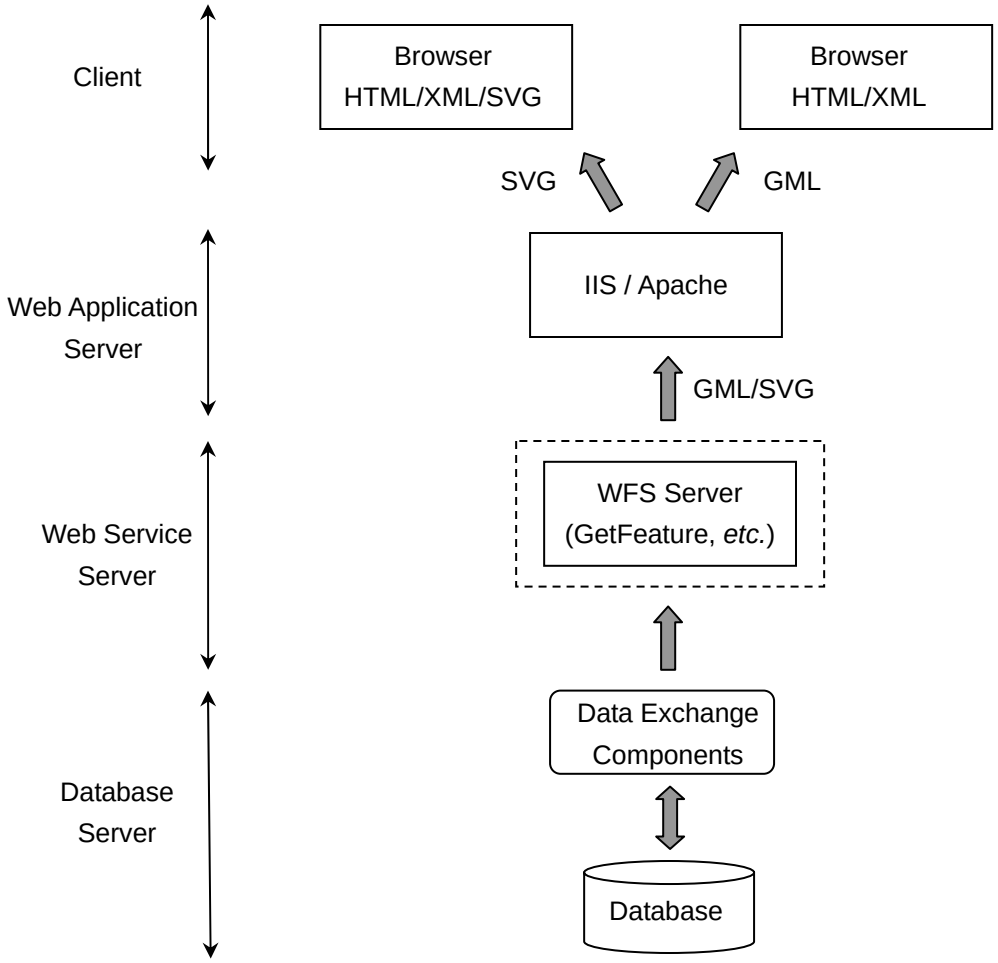


Figure 7 The Structure of the XML-based Spatial Data Interoperability System

service is implemented according to the OGC Web Feature Service Implementation Specification which is described in detail in section 5.1.4. The “GetCapabilities”, “DescribeFeatureType”, and “GetFeature” services should at least be provided by the WFS server. The web application server layer deals with users’ requests and the WFS server’s responses. The GML format is used to

encode and transfer the geographic data from the server to the client. Although some users may need to download the raw GML data directly, most of them often want to visualize them in a web browser or in a certain application. Thus the data exchanging components also may need to transfer the GML data to an SVG document, which is sent to the client by the web application server.

4.2 Implementation

Based on the conceptual framework, two prototypes are implemented by applying various programs, most of which are open source software packages. The main components of the prototypes are Web server and GML-enabled spatial database. The software – MapServer developed by the University of Minnesota – will be employed as the Web server for building the Web-mapping applications. MapServer is a very powerful open source Web GIS software program that aims to facilitate the display and browsing of geospatial data in a variety of vector and raster data formats, such as *shp*, *gml*, *tif*, and *lan etc*. It will help to construct a WFS server to provide interoperability services, such as creating, deleting, or updating a feature instance, or retrieving and querying features. As mentioned in the previous sections, WFS is another OGC implementation specification to conduct queries and extract features from the GML-enabled spatial database to respond to users' requests. By setting up a WFS server with MapServer, the users will be able to retrieve geospatial data encoded in GML.

For the spatial database, the software PostgreSQL with PostGIS extension will be utilized because of its support for GML and spatial data. PostgreSQL is the most advanced open source

Object-Relational Database Management System (RDBMS). PostGIS is an extension of the DBMS to enable the support for spatial information. With PostGIS extension, PostgreSQL works like ESRI's Spatial Database Engine (SDE) and Oracle's spatial extension. Following the OGC's "Simple Feature Specification for SQL" (OGC 1999), PostGIS makes geometry representation possible by using WKT/WKB data format defined by the OGC. There is even a tool for conversion from ESRI Shapefiles to WKT/WKB data format for the database. Therefore, it is an ideal backend database for GML data access through WFS interface in GML/SVG-based Web GIS applications.

GML data cannot be displayed directly because they do not contain any rendering information for visualization. As shown in Figure 6, an XSLT processor needs to be used to convert a GML file to an SVG file to view the GML data. There are three popular XSLT processors which are libxslt, Xalan (Apache 2006) and Saxon (Kay 2006). With the help of the libxslt XSLT processor embedded in MapServer, the GML data are styled into an SVG file for display. This process requires two basic pieces of information: the source GML document to be read into the source tree and the style sheet to apply to the source tree (Tsoulos 2003). Finally, users on the client side will view the generated SVG maps with an SVG viewer installed as a Web browser plug-in. A list of SVG views can be found at the website: http://www.w3.org/Graphics/SVG/SVG-Implementations. The most popular SVG viewer is the Adobe SVG plug-in and it is also recommended to display SVG maps created by the Web GIS applications in this prototype.

The following paragraphs present the step-by-step procedure to create prototypes of the web-based interoperable spatial information system capable of displaying elegant SVG cartographic maps and process attribute query. The prototypes are designed for PC computers with average technical specifications and a fast Internet connection. The server has Microsoft Windows XP as the operating system and Apache installed as a web server. The demo system can be found at http://128.192.49.105/demo/index.htm which provides detailed instructions and illustrations.

4.2.1 Prototype I: Athens Map

Step 1: In the first step, the GML-enabled spatial database was set up with PostgreSQL and PostGIS extension on server 128.192.49.105. Two sample data sets, “airports” and “rivers”, were downloaded from Georgia GIS Data Clearinghouse. Originally the data were in ESRI’s shapefile format. The data were then converted and imported to tables in PostgreSQL DBMS. The conversion is accomplished by using the shp2pgsql utility included as part of the PostGIS extension. With a series of SQL statements (*e.g.* CREATE DATABASE, CREATE TABLE, and INSERT *etc.*) as outputs, this utility takes a shapefile to create a table in the PostgreSQL relational DBMS. The resulting table contains all the attributes of the shapefile and the coordinates that define each feature. Like common SQL commands in relational DBMS, the SQL statements executed in PostgreSQL create a table which represents the shapefile. Geographic projection is also enabled in the shp2sql utility. The PostGIS extension has a file which defines more than 1800 geographical projections. The UTM 17N projection was selected from the

definitions for Athens Georgia area. This procedure was repeated for both layers (“airports” layer and “rivers” layer).

Step 2: After the GML-enabled spatial database was successfully set up, the MapServer program was installed and applied to build a WFS server on the same server where the PostgreSQL database was running. The installation of MapServer was pretty straightforward when the pre-complied Windows version MS4W was downloaded and used. The basic MS4W package installs a preconfigured Web Server environment that includes the following main components: Apache HTTP server, PHP, MapServer CGI and so on. The Apache HTTP server is used to publish web pages so that client users can get web pages when URLs are sent to the server. For example, when the URL http://128.192.49.105/ is sent in the address bar, a default page on the server can be opened. This is achieved with the Apache HTTP server. The PHP processor is used to interpret and process the server-side and HTML-embedded scripting language. PHP provides a way to put instructions into HTML files to create dynamic content. It was used to process controls, user’s inputs, and to invoke and pass parameters to applications with embedded PHP structured codes inside HTML tags. The MapServer CGI is another essential component to implement the WFS. It can read and interpret a mapfile, which is a kind of configuration file for the MapServer software, to implement corresponding functions provided by the software. A specific configuration file corresponds to a service, such as WFS or WMS. For example, the following codes are a part of the mapfile configuration file which provides the needed WFS in this prototype.

```
MAP
    NAME    WFS_Server
    STATUS ON
    SIZE      400  300
    SYMBOLSET "etc/symbols.sym"
    EXTENT 265562.553141 3747702.030423 292867.168071 3769126.796421
    UNITS    METERS
    SHAPEPATH "Data"
    WEB
        IMAGEPATH  "/ms4w/apps/wfs_ser/tmp/"
        IMAGEURL    "/tmp/"
        METADATA
            "wfs_title"             "Liang's WFS Demo Server"
            "wfs_onlineresource"          "http://localhost/cgi-bin/mapserv.exe?"
            "wfs_srs"       "EPSG:26917"
            "ows_schemas_location"     "http://ogc.dmsolutions.ca"
        END
    END
    LAYER
        NAME     airports
        METADATA
            "wfs_title"       "Airports"
            "gml_include_items"     "all"
        END
        TYPE POINT
        STATUS ON
        DATA airports
```

```
        DUMP TRUE
    END
.....
END
```

The mapfile uses pre-defined tags and the reference can be found on the website of MapServer (http://mapserver.gis.umn.edu/). The GML data of the WFS are georeferenced and can have a variety of coordinate systems, In the WEB tag, the “wfs_srs” parameter, which is under the METADATA sub-tag, specifies the spatial reference system of WFS. The parameter uses an official list of codes for each projection and datum created by the EPSG. For example, the UTM 17N (NAD83) coordinate system corresponds to epsg:26917 in the list. Table 3 lists some important codes for projections and datums.

The highlighted area of the mapfile describes the airport layer which includes the layer name, data type, metadata and status. The DATA parameter specifies the data stored in the relational DBMS. The information is used by the software to transfer the “airports” layer to available GML data format. After the mapfile is set with appropriate parameters, MapServer will be able to work with the Apache server to provide georeferenced GML data upon user’s requests as a WFS server.

Step 3: This step employs XSLT to transform GML data to SVG maps for visualization. The XMLSpy, a professional XML software tool, was used as the integrated development environment to develop different XSLT files for GML files. XSLT files are also XML documents, but they provide functions and use XPath to find information in an XML document so that the output XML files have rearranged and sorted elements. XPath is used to navigate through

Table 3 Spatial Reference Codes and Corresponding Projection and Datum Information

SRID	Projection and Datum Information
4326	Datum/Spheroid: WGS84, Projection: Unprojected, Units: Degree, Prime-Meridian: 0 (Greenwich)
26905	Datum: NAD83, Spheroid GRS80, Projection: UTM Zone 5N, Units Meters, Prime Meridian: 0, Central Meridian: -153, False Easting: 500000, False Northing: 0
26906	Datum: NAD83, Spheroid GRS80, Projection: UTM Zone 6N, Units Meters, Prime Meridian: 0, Central Meridian: -147, False Easting: 500000, False Northing: 0
26910	Datum: NAD83, Spheroid GRS80, Projection: UTM Zone 10N, Units Meters, Prime Meridian: 0, Central Meridian: -123, False Easting: 500000, False Northing: 0
26911	Datum: NAD83, Spheroid GRS80, Projection: UTM Zone 11N, Units Meters, Prime Meridian: 0, Central Meridian: -117, False Easting: 500000, False Northing: 0
26912	Datum: NAD83, Spheroid GRS80, Projection: UTM Zone 12N, Units Meters, Prime Meridian: 0, Central Meridian: -111, False Easting: 500000, False Northing: 0
26913	Datum: NAD83, Spheroid GRS80, Projection: UTM Zone 13N, Units Meters, Prime Meridian: 0, Central Meridian: -105, False Easting: 500000, False Northing: 0
26914	Datum: NAD83, Spheroid GRS80, Projection: UTM Zone 14N, Units Meters, Prime Meridian: 0, Central Meridian: -99, False Easting: 500000, False Northing: 0
26915	Datum: NAD83, Spheroid GRS80, Projection: UTM Zone 15N, Units Meters, Prime Meridian: 0, Central Meridian: -93, False Easting: 500000, False Northing: 0
26916	Datum: NAD83, Spheroid GRS80, Projection: UTM Zone 16N, Units Meters, Prime Meridian: 0, Central Meridian: -87, False Easting: 500000, False Northing: 0
26917	Datum: NAD83, Spheroid GRS80, Projection: UTM Zone 17N, Units Meters, Prime Meridian: 0, Central Meridian: -81, False Easting: 500000, False Northing: 0
26918	Datum: NAD83, Spheroid GRS80, Projection: UTM Zone 18N, Units Meters, Prime Meridian: 0, Central Meridian: -75, False Easting: 500000, False Northing: 0
26919	Datum: NAD83, Spheroid GRS80, Projection: UTM Zone 19N, Units Meters, Prime Meridian: 0, Central Meridian: -69, False Easting: 500000, False Northing: 0
26920	Datum: NAD83, Spheroid GRS80, Projection: UTM Zone 20N, Units Meters, Prime Meridian: 0, Central Meridian: -63, False Easting: 500000, False Northing: 0

elements and attributes in XML documents. The principle is quite straightforward for such a transformation. Since the GML data are retrieved from the WFS server and the data follow GML feature schema, geometry schema and application schema, the structure of a GML file can be known. XSLT can simply transform structured GML tags into SVG tags, such as path, polygon, circle *etc*. The attributes of GML also can be retrieved and added to the SVG output. Taking the "airports" GML file as an example, the following section demonstrates the actual process.

GML file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="gml2svgl.xslt"?>
<wfs:FeatureCollection > (Namespace omitted)
    <gml:boundedBy>
        <gml:Box srsName="EPSG:26917">
            <gml:coordinates>
                277460.499801,3758894.999998 285015.156093,3758990.749998
            </gml:coordinates>
        </gml:Box>
    </gml:boundedBy>
    <gml:featureMember>
        <myns:airports>
            <gml:boundedBy>
                <gml:Box srsName="EPSG:26917">
                    <gml:coordinates>
                        285015.156093,3758894.999998 285015.156093,3758894.999998
                    </gml:coordinates>
                </gml:Box>
            </gml:boundedBy>
            <myns:msGeometry>
                <gml:Point srsName="EPSG:26917">
                    <gml:coordinates>285015.156093,3758894.999998</gml:coordinates>
                </gml:Point>
            </myns:msGeometry>
            <myns:AIRPORT_>1</myns:AIRPORT_>
            <myns:AIRPORT_ID>14</myns:AIRPORT_ID>
            <myns:FACTYPE>AIRPORT</myns:FACTYPE>
```

```
            <myns:INFODATE>05/25/95</myns:INFODATE>
            <myns:FAADIST>ATL</myns:FAADIST>
            <myns:STATECODE>GA</myns:STATECODE>
            <myns:STATENAME>GEORGIA</myns:STATENAME>
            <myns:COUNTY>CLARKE</myns:COUNTY>
            <myns:MGRADDR>1010 BEN EPPS RD</myns:MGRADDR>
        </myns:airports>
    </gml:featureMember>
    <gml:featureMember>
        <myns:airports>
            <gml:boundedBy>
                <gml:Box srsName="EPSG:26917">
                    <gml:coordinates>
                        277460.499801,3758990.749998 277460.499801,3758990.749998
                    </gml:coordinates>
                </gml:Box>
            </gml:boundedBy>
            <myns:msGeometry>
                <gml:Point srsName="EPSG:26917">
                    <gml:coordinates>277460.499801,3758990.749998</gml:coordinates>
                </gml:Point>
            </myns:msGeometry>
            <myns:AIRPORT_>2</myns:AIRPORT_>
            <myns:AIRPORT_ID>15</myns:AIRPORT_ID>
            <myns:FACTYPE>HELIPORT</myns:FACTYPE>
            <myns:INFODATE>05/25/95</myns:INFODATE>
            <myns:FAADIST>ATL</myns:FAADIST>
            <myns:STATECODE>GA</myns:STATECODE>
            <myns:STATENAME>GEORGIA</myns:STATENAME>
            <myns:COUNTY>CLARKE</myns:COUNTY>
            <myns:MGRADDR>1230 BAXTER ST.</myns:MGRADDR>
        </myns:airports>
    </gml:featureMember>
</wfs:FeatureCollection>
```

XSLT file (*gml2svg.xslt*):

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet>
    <xsl:variable name="tmp" select="wfs:FeatureCollection/gml:boundedBy/gml:Box/gml:coordinates"
```

```
/>
	<xsl:variable name="ll" select="substring-before($tmp, ' ')" />
	<xsl:variable name="ur" select="substring-after($tmp, ' ')" />
	<xsl:variable name="llE" select="substring-before($ll, ',')" />
	<xsl:variable name="llN" select="substring-after($ll, ',')" />
	<xsl:variable name="urE" select="substring-before($ur, ',')" />
	<xsl:variable name="urN" select="substring-after($ur, ',')" />
	<xsl:variable name="width" select="$urE - $llE" />
	<xsl:variable name="height" select="$urN - $llN" />

	<xsl:template match="/" >
		<svg width = "100%" height = "100%" viewBox=" {$llE - 1000} {$llN} {$width + 5000}
{$height}">
			<defs>
			  <circle id="symbol1" cx="0" cy="0" r="50"/>
			</defs>
			<script   xlink:href="external.js" type="text/javascript"/>
			<xsl:apply-templates
select="/wfs:FeatureCollection/gml:featureMember/myns:airports/myns:msGeometry/gml:Point"/>
		</svg>
	</xsl:template>
	<xsl:template match="gml:Point">
		<xsl:variable name="xCoord" select="substring-before(gml:coordinates, ',')"/>
		<xsl:variable name="yCoord" select="substring-after(gml:coordinates,',')"/>
		<g>
			<circle cx="{$xCoord}" cy="{$yCoord}" r="50" class="airports"
address="{../../myns:MGRADDR}/>

		<text class="airports-text" x="{$xCoord+100}" y="{$yCoord}"><xsl:value-of
select="../../myns:MGRADDR"/></text>
		</g>
	</xsl:template>
<!-- Highlighted area is the actual transformation part !-->
</xsl:stylesheet>
```

Output SVG file:

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" width="100%"
height="100%" viewBox=" 276460.499801 3758894.999998 12554.65629200003 95.75">
	<defs>
```

```
            <circle id="symbol1" cx="0" cy="0" r="50"/>
        </defs>
        <script xlink:href="external.js" type="text/javascript"/>
        <g>
            <circle cx="285015.156093" cy="3758894.999998" r="50" class="airports" address="1010
BEN EPPS RD"/>
            <text class="airports-text" x="285115.156093" y="3758894.999998">1010 BEN EPPS
RD</text>
        </g>
<!-- Note the highlighted structure in the XSLT file is the same !-->
        <g>
            <circle cx="277460.499801" cy="3758990.749998" r="50" class="airports" address="1230
BAXTER ST."/>
            <text class="airports-text" x="277560.499801" y="3758990.749998">1230 BAXTER
ST.</text>
        </g>
<!-- Note the highlighted structure in the XSLT file is the same !-->
</svg>
```

Besides the “airports” layer, the XSLT file also was created for the “rivers” layer. However, their XSLT files were different because the “rivers” layer was composed of several lines which have a list of coordinate pairs. The coordinate pairs were read from the GML file and converted to specific SVG format. In addition, the Y coordinate must be multiplied by “-1” because the original point of GML coordinate system is the lower left point whereas the original point of the SVG coordinate system is located at the upper left point. This is achieved by using the “transform” function in SVG.

Step 4: Following the above procedures, the WFS server was set up and the GML data could be converted to SVG documents with an XSLT processor automatically. For better cartographic visualization, an SVG map viewer was designed in this step. The purpose of the map viewer was to allow a client user to pan, zoom, and toggle layers on and off. For the Internet Explorer, the

Adobe SVG Viewer plug-in enables common operations like zooming and panning. However it is not likely for native SVG-supported web browsers. Mozilla Firefox is one of the popular web browsers that support SVG natively. This means the browser can display an SVG document without any plug-in installed. However, the SVG document is displayed like a common picture which lacks the ability to pan and zoom. With the SVG Viewer, these operations become possible in Internet Explorer, Mozilla Firefox or other web browsers.

The development of the SVG map viewer was based on the work of SVG navigation graphical user interface (GUI) provided on the website of www.carto.net. The SVG navigation GUI aims to assist "SVG only" web mapping applications in creating 2D map navigation tools. Thus the SVG map viewer is also a pure SVG document with several buttons and controls. Another important part of the viewer is javascript files which make user's operations possible. Different functions are defined in several javascript files. According to the documentation of the navigation tools, the following features are provided:

- Zoom in and zoom out with buttons and fixed steps
- Continuous zooming with zoom slider
- Button to go back to full view
- Button for re-defining map center in main map
- Manual zooming with dragging a rectangle in the main map
- Manual panning with mouse-down and dragging in the main map
- Linked reference map shows current map extent and allows repositioning
- Coordinate display
- Display of map extent (width and height)
- Zoom or pan modes for repeatedly using the same mode

- The new map extent can be set by script (method of the map object)
- History of map extents is tracked, user can go back and forth in list of previous map extents
- Cursors give feedback to user-actions, *e.g.* Indicate zoom or pan mode

In the prototype, the navigation GUI was customized in terms of data source, appearance and arrangement of controls. The boundary of Athens was first incorporated in the SVG map viewer as a reference. The vector data of Athens boundary were originally downloaded from the Georgia GIS Data Clearinghouse in ESRI's shapefile format. Like the "airports" and "rivers" layers, the "boundary" layer was also imported into the GML-enabled database and made available on the WFS server. Then an SVG fragment was produced and added to the SVG map viewer as a layer. Corresponding javascript was also edited and revised to make the script associate with the "boundary" layer so that the buttons and controls worked correctly.

Besides the boundary of Athens, it would be better if the shaded relief of Athens area can be displayed as a layer in the SVG map viewer. To create the shaded relief map, the DEM data were also downloaded from the Georgia GIS Data Clearinghouse and processed with the Erdas Imagine software. Since SVG files can embed and link images with the <image> tag, a raster image was produced in JPEG format. The image was added to the SVG map viewer for more information by using the following SVG code:

```
<image id="ShadedRelief" x="265097" y="-3769497" width="28325" height="22122"
xlink:href="athens_relief.jpg"/>
```

The "x" and "y" specify the upper left corner coordinates of the image. The width and height are actual width and length values of the area in meter. The "xlink:href" parameter links the JPEG file to the SVG so that the image is embedded for display.

Similarly an ASTER image was also created to provide a remotely sensed false color image of Athens area. The original ASTER image was obtained from the Center for Remote Sensing and Mapping Science (CRMS) at the University of Georgia. The original data format was Imagine's proprietary IMG format. The raw image had no projection information, so it was geo-referenced with Erdas Imagine. Like the shaded relief image, the output ASTER image linked to the SVG map viewer was also a JPEG image.

The last layer added to the SVG map viewer was a WMS map layer provided by the Terraserver-USA server. The Terraserver-USA server is a joint project by USGS, Microsoft and HP to provide free OGC WMS services to the public (currently mainly USGS DOQ and DRG data). The DOQ layer is added to the viewer by sending URL request to the Terraserver's WMS server. The URL contains the servername, followed by the service name and the parameters, such as version, request and service. The parameters also follow the rules explained in section 5.1.5. The dynamic update of the DOQ layer is implemented with a javascript function in the SVG map viewer. The function sends a new request each time after the user zooms or pans the map so that a new DOQ layer can be retrieved from the Terraserver-USA server.

After customization, the SVG map viewer can be used as a base layer in the prototype to view the "airports" and "rivers" layers which are GML data from the WFS server in nature. A screenshot of the customized SVG map viewer is shown in Figure 8.

Step 5: The final step included the web form design and PHP coding. The web form is processed by PHP server-side scripts. Figure 9 shows the web form of the prototype. The user selects a layer and may type filter in the text box. The filter follows the OGC Filter Encoding

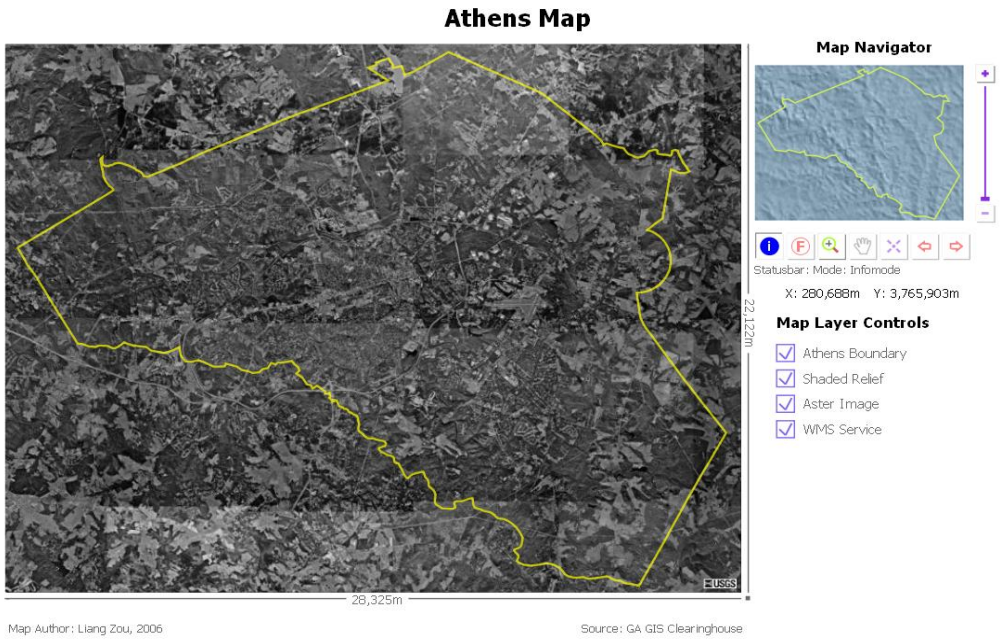


Figure 8 A Screenshot of the Customized SVG Map Viewer

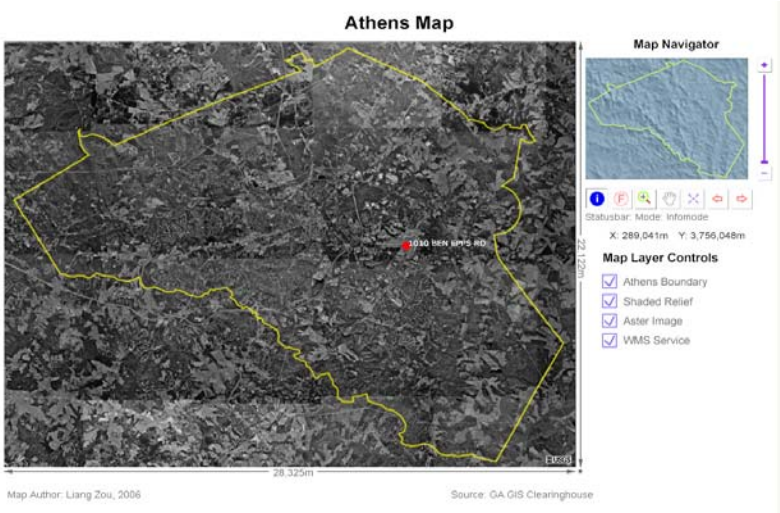
Figure 9 The HTML Web Form

Specification and defines query constraints for the GetFeature operation. If the filter parameter is not specified, all features will be fetched. After the “Request” button is clicked, the PHP script will retrieve web form parameters including layer name and filter text. Based on the parameters, GML data of appropriate layers will be fetched from the WFS server. Then Document Object Model (DOM) functions embedded in PHP will convert the GML data to SVG fragment which will then be appended to the SVG map viewer for display. Figure 10 (a) displays a request with the filter: MGRADDR field equal to “1010 BEN EPPS”, whereas Figure 10 (b) shows a request without any filter.

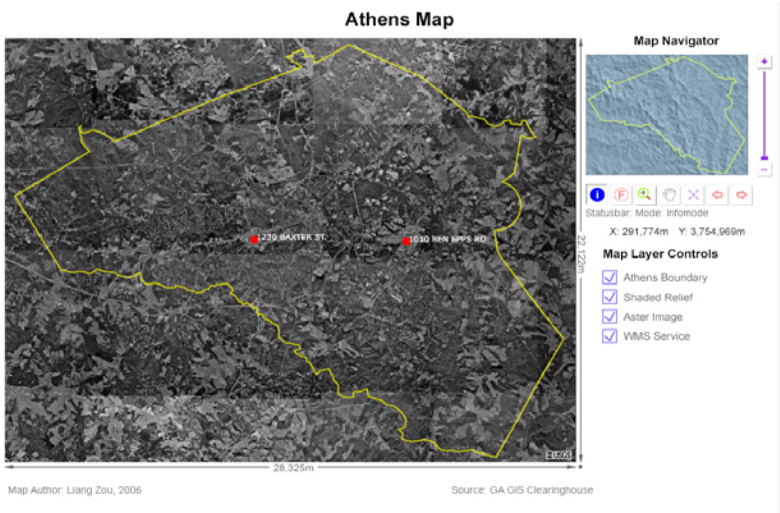
4.2.2 Prototype II: University of Georgia (UGA) Campus

The UGA campus prototype followed very similar steps implemented in the Athens GA prototype. This prototype was developed to further demonstrate the feasibility of the framework and the suitability for various data type. Because of limited development time for the prototype, the SVG maps were generated directly and sent to web browsers for display. That means this prototype was dependent on the Adobe SVG Viewer to pan and zoom the maps. The WFS data included UGA buildings, UGA properties, and UGA roads, all of which contain attribute information for each feature. Their corresponding XSLT files were also created to use for GML/SVG data conversion.

Following similar procedures, the web form was designed as Figure 11 shows. The user can select different layers or all layers together to convert GML to SVG on the fly for display. The principle is the same as the Athens prototype.



(a) SVG Map Generated from Filtered GML Data



(b) SVG Map Generated from Unfiltered GML Data

Figure 10 SVG Maps Generated From GML Data

One characteristic of the UGA campus prototype is the demonstration of animation ability of the SVG format. The default layer of the prototype web page displays the path from my home (Rogers Rd. Bldg. S) to the Geography Department (Field St., Geography/Geology Bldg). To obtain the name of a building, the user can simply click on a feature. When the mouse moves over the feature, its color will also change to notify the user that this feature is selected. In

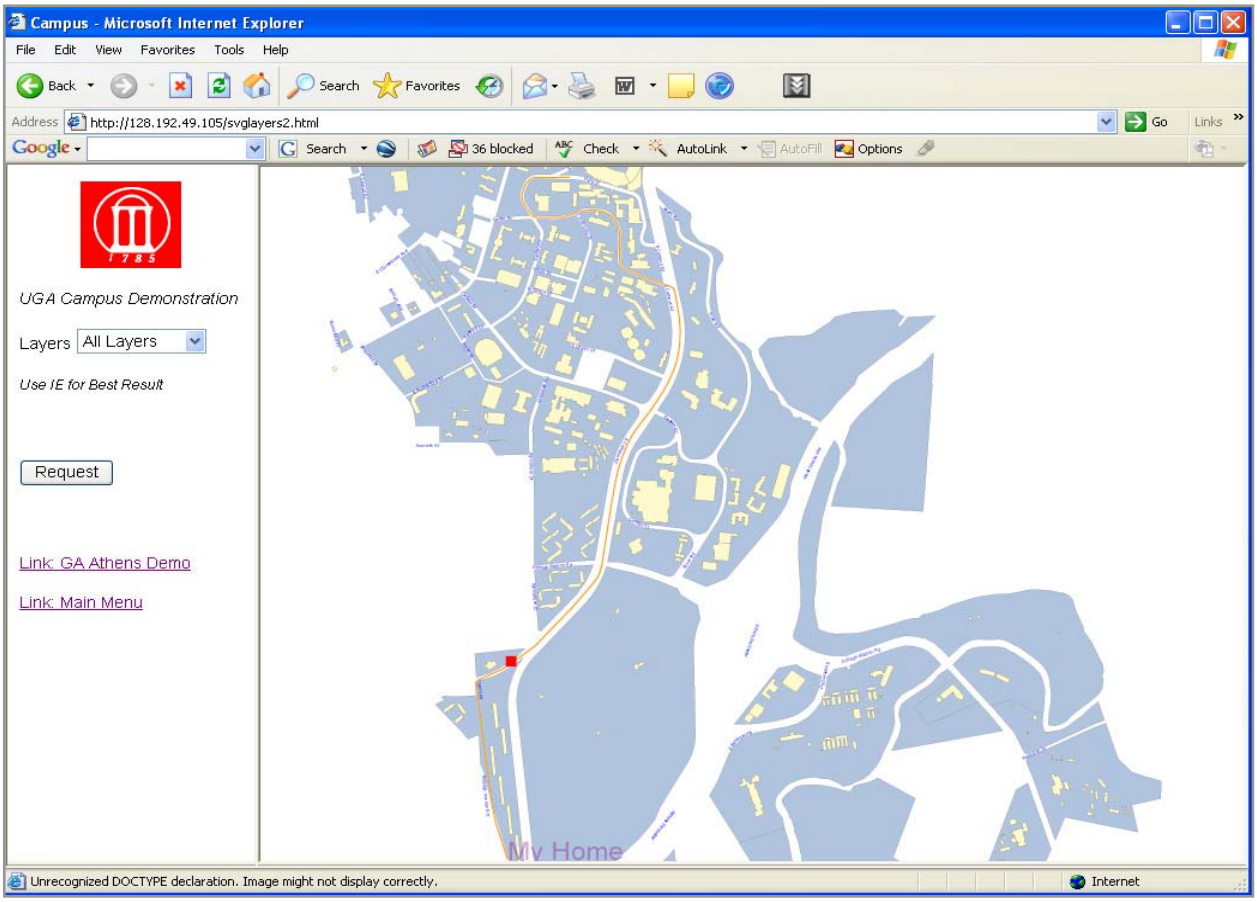


Figure 11 The Campus Prototype Web Form

addition, if a road needs to be searched, the user can type the road name and search the text because the road names are vector texts. This operation is very convenient for a large number of road names.

4.3 Discussion

This paper has proposed a conceptual framework to integrate GML, SVG, WFS, and WMS for developing interoperable Web GIS applications. By integrating GML, SVG, WFS and WMS, the

process of developing two simple prototypes has shown that the open standards are completely compatible and hold great promise in the future development of Internet GIS. The major advantage of the standard-based approach is interoperability. Proprietary data are converted and stored in the GML-enabled database and the GML data are provided through WFS servers to appropriate organizations and institutions who can process and visualize the data without proprietary software applications. With the application of SVG, the output maps have high quality and good interactivity. However, there are still some issues to be considered.

GML, SVG, WFS, and WMS are all standards so that Internet applications based on these standards are compatible with other standards including XML, XSLT, DOM, HTML and so on. For example, the prototypes can be developed by other open standards and APIs of other development environments. Besides PHP, Microsoft C#.net or VB.net can also be used to process users' requests and the transformation between GML and SVG data. Thus open standards enable more choices. In addition, the implementation of interoperability of the standard-based approach is illustrated by several examples. One good example is the GML Relay Demonstration, which demonstrates the interoperable capability of building services that combine data from different sources in a seamless client user interface. Stored in text format, GML data can easily be integrated into other data across a variety of platforms (Lake 2001). Peng and Zhang (2004) addressed the interoperability and graphic image output issues when they researched the role of GML, SVG and WFS specifications in Web-based GIS. They employed GML as a data coding and interchanging mechanism to achieve interoperability and WFS as a data query mechanism to search and access feature-based data on the Web. Their conclusion shows that the combination of

GML, SVG and WFS has enormous potential to distribute spatial information on the Internet. However, the interoperability of GML is not absolute because of a potential problem in GML encoding. GML is extensible in that different tags can be created to encode features, feature collections, and geometries. The unconstrained tags provide great flexibility, but at the same time they make it difficult to develop a common tool to process various GML files. A variety of GML encodings may also cause semantic confusions between different organizations. This may also bring about problems in data interoperability.

The conceptual framework combines server-side operations and client-side applications to balance the work load between servers and clients. After the server processes a request, the SVG data are sent to the web clients. Users interact with the SVG data directly and locally without sending further requests to the server for simple operations like panning, zooming, and toggling layers on and off. Thus the client side approach enables better performance. Nonetheless, both GML and SVG data are text-based. This is a strength for interoperability because text-based files can be viewed and used without proprietary applications, but one shortcoming of this approach is the file size. Compared with some binary GIS data format, the size of GML and SVG documents is quite large. For example, the size of the “building” layer of UGA in shapefile format is 323 Kb whereas the corresponding GML file occupies 632 Kb on the hard disk. The large file size may become a hinder for the popular use of GML as a means of data transportation over the Internet. The size of GML and SVG files is sensitive to the data contents. If a large number of features exist, the file size will be quite large. If the features are composed of numerous very high accurate coordinate pairs, the size will be larger. Therefore, there is a tradeoff between the map

detail/accuracy and the GML/SVG file size. However, raster images are sensitive to resolution instead of data contents. Raster file size can grow substantially with the increase of image resolution. For a small resolution, raster images would be smaller than the SVG and GML file. For a high resolution, raster image will be larger. With GML or SVG data, the user can always have the full resolution map no matter how closely the user zooms in anywhere on the SVG map. Nevertheless, an image will become blurred if the resolution is not fine enough to display map details when the user zooms in.

The prototypes illustrate that SVG can deliver better and higher quality maps over the Internet than raster pixel-based images. Compared with the shaded relief and the remotely sensed ASTER image, SVG maps offer enlarged view without any degradation and can be printed in high quality on the client side. The “staircase” problem, which is common for GIF or JPEG raster data, does not exist for SVG document. In addition, an SVG feature can be assigned to a rich set of event handlers such as “on_click”, “on_mouse_over” so that complex animations are possible via javascript. Since SVG document conforms to the DOM, every feature in the document is accessible and its attributes can be changed with any programming language that provides DOM functions. Concerning the conversion from GML to SVG with XSLT, the appearance of the output SVG map can be various because the XSLT stylesheet determines how to render each feature. GML separates data contents from the presentation. This characteristic offers the user more flexibility to display data. Given a GML file, a number of SVG maps can be created according to a user’s preferences.

During the process of developing the prototypes, it was found that the performance was mainly determined by the file size of the GML data. With a large GML file, the process of parsing GML and converting to SVG will become time-consuming so the user may wait for a much longer time. To improve the performance of the system, a key means is to reduce the file size. There are three basic ways to achieve this aim. The first method is to use compression algorithm. Since both GML and SVG data are based on text, the bulky data can be compressed significantly so that the data file becomes compact. Generally the GML and SVG files are compressed on the server side and decompressed on the client side. As a matter of fact, there is already a compressed SVG file type “svgz” available for Adobe SVG Viewer. However, if a file size is very huge, this method may not be a good choice since the compressed file will still be quite large.

Another means to reduce data amount is to eliminate unnecessary points (coordinates) of the GML file. It is known that the file size of GML and SVG data is proportional to the number of features and how accurate they are described by points. Sometimes a feature is delineated very accurately with a large number of points. For example, a line can be described by two points. It may also be separated by three extra points in the original file. The three points can be eliminated without influencing the data content significantly. Furthermore, web applications may not need very accurately delineated features. In this case, a function can be applied to the original file to extract essential data content and get rid of other unnecessary data. This method is an easy and efficient way in compacting data. In the UGA campus prototype, the shapefile of the “buildings” layer was processed by the “Generalization Tool” of ArcGIS to eliminate some unnecessary

points. The quality of the data degraded so little that it was hard to visually discern the differences between the original data and the processed data. However, the GML file of the original shapefile consumes 880 Kb on the disk while the processed file is only 632 Kb. If lower quality is also accepted, the data size can be reduced further.

The third way to improve the performance of the system is to send the GML and SVG data to the client in stages or progressively. This is the most elegant, but also the most complicated means to promote the processing efficiency of the system. This method uses generalization algorithm to calculate visible feature for a certain scale. If a user is viewing the full extent of map, only limited number of important features will be fetched from the server. If the user zooms in, more details and small features are visible and should be retrieved from the server. Only a small range of area, however, will be displayed at the same time. This means the data volume is also small in this case. With this method, the size of requested data would be reasonably small. With the implementation of WFS, it is feasible to rank each feature in the spatial database so that different ranks are corresponding to some ranges of map scales. The user extracts only a few features rather than the data of the whole area.

Finally the security issues must be considered for GML and SVG since they are text-based and can be opened by a simple text editor such as Notepad or any word processor. Thus they are easy to be obtained and understood by users. The open standards provide advantages for interoperability, but they are also drawbacks for confidential information. It would be difficult to protect GML or SVG data, so they are not ideal formats to store and transport secret or private data for the Internet. Another drawback is associated with copyrighted data because the data in

GML or SVG format can be easily stolen. For example, like HTML files, a user can right click the mouse to see and save the source codes of a SVG file. If a company develops a Web GIS application using SVG as the output format, clients can easily download each map which may be copyright protected. An encryption mechanism must therefore be developed to protect copyrighted or sensitive data (Peng and Zhang 204).

CHAPTER 5

CONCLUSION AND FUTURE RESEARCH

By applying GML, SVG, WFS and WMS techniques, advanced Web-based GIS applications can be implemented. Two Web-based GIS prototypes which mainly deal with spatial data encoding and the presentation of geographic data are addressed in this paper. GML is an effective means to encode, store, and transport geospatial data. It is also an efficient means to foster data portability and interoperability. Combined with javascript language, SVG produces high-quality graphics on the Web, which is ideal for displaying spatial data and making intelligent maps. WFS and WMS provide seamless link of GML data and image data from different sources over the Internet all over the world. GML and WFS provide an ideal mechanism to extract and obtain feature level geospatial data in the Internet environment. When applications need to use raster data, WMS provides raster images, such as remotely sensed images or scanned maps. GML, SVG, WFS, WMS are all standards and thus contribute to the interoperability of geographical information. The standards allow us to build an interconnected geospatial data web which makes it possible to share and integrate data from different sources.

However, there are still many issues to be resolved. The GML and SVG standards are all relatively new, so more research and experiments are needed to test these concepts. For example, the compression of GML and SVG files is an immediate need and probably the easiest issue to be resolved. More sophisticated client-side SVG user interfaces and data processing tools should

be developed to assist users as they interact with GML data and conduct spatial analysis. In addition, GML is text-based, and can be opened using a simple text editor such as Notepad or any word processor. GML files are easy to understand, edit, maintain, and update, and thus not secure enough for copyrighted project or sensitive data.

The integration of standard-based programs represents the future direction of distributed GIS, which is most likely to be Web services or location-based services. That is, the creation of a wide range of services that can be accessed across the Web, and Web services that can perform actions and return the outcome in a standard format to the client. GML and SVG show great promise and could become key standard technologies to facilitate the development of wired or wireless Web GIS. However, the full potential of these new standards are not fully applied in current Web GIS environment and needs to be exploited in future research (Peng and Zhang 2004). By delivering services in the Internet's open, standards-based environment, it will make access to and use of geodata and geoprocessing resources much easier and less expensive. In the future, GML combined with a wide range of scientific and business models could provide GIS professionals and non-technical users with convenient tools to perform complex analyses and build what-if scenarios on the Web based on spatial and non-spatial information. This open and standard-based Internet GIS would allow GIS technology to play an even greater role in society. Increased access to GIS analysis tools and data will certainly help both policy makers and the public to make better-informed decisions.

In summary, both of GML and SVG are XML-based files, so they are compatible with each other naturally. The increasing wide applications of GML and SVG have proved the potential and

advantages of the new Web techniques. It is time for GML and SVG on the Internet to produce high quality interactive, dynamic, and animated maps (Neumann and Winter 2001). However, the applications of them are quite limited right now and we have not taken advantage of the full potential of GML and SVG.

REFERENCES

Anderson, G. and Moreno-Sanchez, R., 2003, Building Web-Based Spatial Information Solutions around Open Specifications and Open Source Software, *Transactions in GIS* 7(4): 447-466

Apache, 2006, Xalan-Java Version 2.7.0, available at http://xml.apache.org/xalan-j/, last accessed on January 22, 2006

Autodesk Inc., 2005, MapGuide View Help, http://www.mapguide.com/help/ver6.5/viewer/en/, last accessed on Nov. 6, 2005

Babu, M. N., Implementing Internet GIS with Java based Client/Server Environment, *Map Asia Conference* 2003

Brinkhoff, T., 2003, A Portable SVG Viewer on Mobile Devices for Supporting Geographic Applications, Proceeedings 6th AGILE Conference on Geographic Information Science, Lyon, France, Presses Polytechniques et Universitaires Romandes, 87-96

Brovelli, M. A. and Magni, D., 2002, An Archaeological Web GIS Application Based on MapServer and PostGIS, *Open Source Free Software GIS - GRASS users conference proceedings* 2002, http://www.ing.unitn.it/~grass

Carrara, P., Fresta, G. and Rampini, A., 2003, Banco: an SVG-based Approach to Create Web Sites for the Management of Remote Sensing, Spatial and Non-spatial Data, *Internal Journal of Remote Sensing*, Vol. 24, No. 20, 3903-3915

Cecconi, A. and Galanda, M., 2002, Adaptive Zooming in Web Cartography, *Computer Graphics Forum*, Vol. 21, No. 4, 787-799

Crossley, D. and Boston, T., 1995, A Generic Map Interface to Query Geographic Information Using the World Wide Web, *AUUG 1995 Asis-Pacific World Wide Web Conference Proceedings*

David, J. A., Kerry, T., and Stuart, H., 1998, An Exploration of GIS Architectures for Internet Environments, *Computer Environment and Urban Systems* 22: 7-23

Devogele, T., Parent, C., and Spaccapietra, S., 1998, On Spatial Database Integration, *International Journal of Geographical Information Science*, Vol. 12, No. 4, pp335-352

Gong, J. Y., Chen, N. C., Zhu, X. Y., and Zhang, X., 2003, Design and Implementation of a Distributed Lange-scale Spatial Database System Based on J2EE, *Proceedings of Society of Photo-Optical Instrumentation Engineers* Vol:4886

Gould, M. and Ribalaygua, A., 1999, *A New-breed of Web Enabled Graphics*, http://www.geoplace.com/gw/1999/0399/399svg.asp, last accessed on Nov. 14 2005

Green, D. and Bossomaier, T., 2002, *Online GIS and Spatial Metadata*, Taylor & Francis, London and New York

Harrie, L., Sarjakoski, L. T., and Lehto, L., 2002, A Variable Scale Map for Small-display Cartography, *the 10th International Symposium on Spatial Data Handling*, Ottawa, Canada

Huang, B. and Lin, H., 1999, GeoVR: A Web-based Tool for Virtual Reality Presentation from 2D GIS Data, *Computers and Geosciences* 25: 1167-1175

Huang, B. and Lin, H., 2002, A Java/CGI approach to developing a geographic virtual reality toolkit on the Internet, *Computers and Geosciences* 28: 13-19

Huang, B. and Worboys, M. F., 2001, Dynamic Modeling and Visualization on the Internet, *Transactions in GIS* 5(2): 131-139

Kay, M. H. Saxon, 2006, the XSLT and XQuery Processor Version 8.6.1, available at http://saxon.sourceforge.net/, last accessed on January 22, 2006

Kim, H. and Kim, M., 2003, XML and Interoperability in Distributed GIS, http://www.fig.net/pub/fig_2003/TS_10/PP10_4_Kim_Kim.pdf, last accessed on Nov. 8, 2005

Köbben, B., *SVG and the TOP10NL Project*. Presented at the EuroSDR Workshop, “Visualization and Rendering”, ITC Enschede, Netherlands, 22-4 January 2003.

Lake, R., 2001, GML2.0 – Enabling the Geo-spatial Web, http://www.galdosinc.com/files/GML2-PoweringTheGeoWeb.pdf, last accessed on Nov. 9, 2005

Laurini, R., 1998, Spatial Multi-database Topological Continuity and Indexing: a Step towards Seamless GIS Data Interoperability, *International Journal of Geographical Information Science*, Vol. 12, No.4, pp. 373-402

Lbath, A. and Pinet, F., 2000, The Development and Customization of GIS-based Applications and Web-based GIS Applications with the CASE tool AIGLE, *Proceedings of the 8th ACM international symposium on Advances in geographic information systems*: 194-196

Lee, B. L., Kim, Y. C. and Yun, J. I., 1998, Web interface for GIS in Agriculture, by *the Asian Federation for Information Technology in Agriculture*: 107-111

Lime, S. and Koormann, F., 2005, *MapServer CGI Reference version 4.6*, http://mapserver.gis.umn.edu/doc46/cgi-reference.html, last accessed on November 4, 2005

Neumann, A. and Winter, A.M., Time for SVG - Towards High Quality Interactive Web-Maps, *Proceedings of the 20th International Cartographic Conference*, Beijing, 2001, pp. 2349 – 2362.

OGC, 1999, *OpenGIS Simple Features Specification for SQL*, available at http://portal.opengeospatial.org/files/index.php?artifact_id=829, last accessed on January 22, 2006

OGC, 2004, *OpenGIS Geography Markup Language (GML) Encoding Specification 3.1.1*, available at http://portal.opengeospatial.org/files/?artifact_id=4700, last accessed on November 3, 2005

OGC, 2005a, OGC Implementation Specifications Website, available at http://www.opengeospatial.org/specs/, last accessed on November 3, 2005

OGC, 2005b, *OpenGIS Filter Encoding Specification*, available at http://portal.opengeospatial.org/files/?artifact_id=8340, last accessed on February 2, 2006

OGC, 2006, available at http://www.opengeospatial.org/, last accessed on March, 26th, 2006

Peng, Z. R. and Tsou M. S., 2003, *Internet GIS: Distributed Geographic Information Services for the Internet and Wireless Networks*. John Wiley & Sons, New York

Peng, Z. R. and Zhang, C., 2004, The Roles of Geography Markup Language (GML), Scalable Vector Graphics (SVG), and Web Feature Service (WFS) Specifications in the Development of Internet Geographic Information Systems (GIS), *Journal of Geographical Systems* 6: 95-116

Peng, Z. R., 1999, An Assessment Framework of the Development Strategies of Internet GIS, *Environment and Planning B: Planning and Design* 26: 117-132

Plewe, B., 1997, *GIS Online: Information Retrieval, Mapping, and the Internet*, Santa Fe: Onword Press

Polley, I., Willamson, Ian, and Effenberg, W., 1997, Suitability of Internet Technologies for Access, Transmission and Updating of Digital Cadastral Databases on the WWW, *The 25th Annual Conference of the Australasian Urban and Regional Information Systems Association Christchurch, New Zealand*, 19-21

Sayar, A., Aktas M. S., Aydin G., Fox G. and Pierce M., *Developing a Web Service-Compatible Map Server for Geophysical Applications*, Submitted to ACM-GIS Conference 2005

Tsoulos, L. Spanaki, M. and Skopeliti, A., 2003, An XML-Based Approach for the Composition of Maps and Charts, *Proceedings of the 21st International Cartographic Conference*, Durban, 2003

W3Schools, 2005, Introduction to XML, http://www.w3schools.com/xml/xml_whatis.asp, last accessed on Nov. 13, 2005

Zhang, C., Peng, Z.R., Li, W., and Day, M.J., 2003. GML-Based Interoperable Geographic Databases. *Cartography*, 32(2): 1-15