Centre for Geo-Information

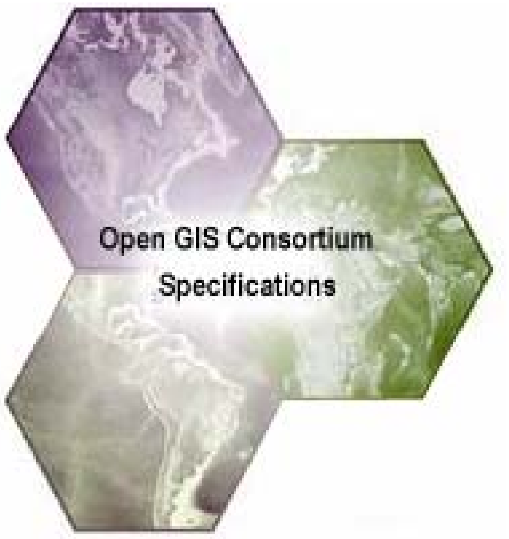
Thesis Report GIRS – 2004-2

OGC Specifications For Seamlessly Interoperable Geo Web Services

-- An Outlook On LBS

Author: Cong Ma

February 2004



WAGENINGEN UNIVERSITY

WAGENINGEN UR

OGC Specifications For Seamlessly Interoperable Geo Web Services

-- An Outlook On LBS

Author: Cong Ma

Registration number: 761104536010

Supervisors:

Ron Van Lammeren

A thesis submitted in partial fulfillment of the degree of Master of Science at Wageningen University and Research Centre, The Netherlands.

February 2004
Wageningen, The Netherlands

Wageningen University and Research Centre
Laboratory of Geo-Information Science and Remote Sensing
Thesis Report GIRS-2004-2

Abstract

Spatial data is being used by an ever-increasing number of organizations - from city to national governments, and from small companies to large corporations - who all view spatial data as a strategic asset. Traditionally, it has been very difficult for an enterprise or organization to consolidate its disparate map data into a single, seamless database, and integrate this significant asset into the decision making process.

Nowadays, geographic data has been bounded up in the proprietary data format from many predominant and monolithic data providers. All those data formats are not interchangeable for most of GIS applications. Moreover, web service is rapidly becoming a paradigm for geographic data sharing, by leveraging Internet, the most widely available distributed platform. However, with certain devices or applications, the user is merely able to access the particular data resources. Especially in the scenario of LBS, the users is normally moving, which require more broad and extensive data sharing. Non-conformance is obviously becoming the major hurdle of LBS. Therefore, the interoperability for data providers is getting more imperative, which is expected to allow users to move transparently and smoothly among data providers.

Aiming to realize this boundless and transparent geographic data sharing and remarkable interoperability, the OpenGIS Consortium devised a series of abstract and implementation specifications, including most of the facets of data operation, namely data register (Catalog Interface), data structure (GML) and data transaction (Web Feature Service).

This paper is attempting to explore an integrated and pragmatic solution to implement OGC WFS (Web Feature Service) specification, serving to rambling mobile user with a common and seamless interface for data retrieval.

Keywords: OGC, XML, GML, WFS, Web Service, Mysql spatial extension.

Chapter 1 Introduction

1.1 Context and background

Location Based Services is a growing technology field that focuses on providing GIS and spatial information via mobile and field units. The wireless Internet is poised to tap our daily life. The fruitful handheld devices and wireless technology take Location Based Services out of the realm of fantasy and make it real-world possibility.

To carry out any kind of LBS, the geo-data is most crucial underpinning. However, the enormous geo-data litter around the thorough network, collected or compiled in heterogeneous format. For the sake of performance or their own convenience, the predominate data producer developed all proprietary format. Historically, the task of moving geographic data from one format to another has been difficult. As a result, users with large data stores have been attached into a single vendor's format and have been restricted to using one vendor's analysis and decision support tools. It's quite imperative to appeal for a unifying benchmark regarding geo-data's transport and storage, especially for those data providers who disseminate their data via Internet in term of web service.

The Internet is quickly evolving from today's Web sites that just deliver user interface pages to browsers to a next generation of programmable Web sites that directly link organizations, applications, services, and devices with one another. These programmable Web sites become more than passively accessed sites - they become reusable, intelligent XML Web services. This rapidly emerging paradigm for Web-based distributed computing is perfectly tailored for organizations that produce and use geospatial information. Open GIS Consortium (OGC) members are specifying the interoperability interfaces that enable software vendors to deliver geoprocessing services in the Internet's open, standards-based environment. This will make publishing, discovery, access and use of geo-data and geoprocessing resources much easier and less expensive.

One of the most remarkable achievement of OGC, the Geography Markup Language (GML) attempts to alleviate these difficulties by increasing organizations' ability to share geographic information. GML, which is based on the extensible Markup Language (XML), is an open and non-proprietary specification used for the transport and storage of geographic information. Besides, OGC also contrived many specifications at both abstract and implementation level, in order to dramatically enhance the interoperability with a common interactive interface. We can foresee all those standards are about to spur the more pervasive and ubiquitous data sharing, without trampling too much on the toes of existing works.

1.2 location based service

1.2.1 Introduction

Since the advent of wireless network technologies, this past year saw the number of mobile phone users in China soar to a enormous quantity. GPRS wireless connection has been provided in most city of China. While numbers are smaller in other parts of the world it is clear that wireless Internet is here to stay.

One of the key requirements of any mobile wireless user is to know what is going on around them. Questions such as “Where am I?” “What is near by?” have special meaning to a moving user often navigating in an unfamiliar environment. Location is a common denominator in every human endeavor. Where you are matters. Where you're going matters. Where you've been matters. What's around you? Where do you want to live? Where's the office? Grocery store? Church? How do you get there? Want to take a trip? How about a virtual trip?

As far as business is concerned, location matters even more. Where are my customers? Where do I locate my facilities? What's my most effective approach to the distribution or the supply chain? Where are the hot prospects? Competition? Where are my fixed assets? Mobile assets? What location-based services can I provide my customers?

Let's take an example. I arrive late at night in a strange city, rent a car and head for my hotel 50 miles out of town. En route I realize that the highway ahead requires local currency, and in my haste to make the plane I did not arrange to change any money. I could go back to the ATM's in the airport, but that means finding a parking space and navigating to the airport. In addition why lose the time of going in the opposite direction. I need to locate the closest ATM before I reach one of those tollbooths.

Many of us think of location information as the graphics and text that are captured on maps. But let's look beyond the map. There's geocoded information such as street addresses, postal codes and zip codes; positional data that are captured for navigation purposes, like GPS, GSM and LORAN; satellite and aerial imagery that represent our earth from a birds eye view, even capturing spectra that are imperceptible to the naked eye; route information and directions that tell us how to navigate from one place to the next; time-sensitive events, like accident reports, weather reports and the location of service fleets; directories, like the yellow pages; countless databases with demographic and psychographic data, customer data, asset inventories, and more; gazetteers with place names; books, documents and reports that contain places, addresses, etc.; sites and landmarks that have special meaning to us; transaction reports; resource inventories; and so much more. Location information is ubiquitous.

However, how all such information could be conveyed to those moving user who need it? There are many possibilities. It could be a voice message ("the closest ATM is located at … "). It could be a route displayed on a map on my in-car computer, or on my cell phone or PDA. It could be driving directions sent to my in-car navigation system. The key thing to understand here is that it may be any or all of these different possibilities and more besides. Furthermore we can expect different types of communication modalities to be available at different points in time and for different purposes. In every case, however, the information supplied rests on a geographic model of the world around us.

Unlike the traditional desk-bound surfer that navigates a virtual world in cyberspace, the wireless user navigates the real world through the lens of the Location based Service.

1.2.2 Open Application Service

Location Based Service is predominantly distributed via Internet, which is transforming the landscape of business by providing a broadly accepted open and distributed application architecture for digital communications and services. Creatively, and systematically, new and established businesses are exploring the impact of the Internet upon their businesses. And the results are dramatic. The Internet invites innovative, faster, cheaper and broadly accessible solutions. Barriers that once held competition or limited growth are now disappearing.

However, most of the services provider sticks to their monolithic system running on Internet, for the sake of security or performance. Consequently, the user who once decided to choose one of the service providers will be engaged with that single provider. Likewise, providers are also discovering the need to continuously refine the scope of their offerings and technologies of interoperability. Perhaps most dramatic are the forces of change that are driving the momentum of the open application service.

Application services will add increasing levels of automation to business processes, with more software dedicated to providing these services ("business intelligence") when and where needed. But, for this to happen, software that is now locked in closed systems and applications must be reconfigured into open application services, such that their functions are then available to any client on the Net. This is a crucial technology trend to understand and bring to bear in any IT projects.

Understanding the importance of open technology approaches, providers are busy recasting their technologies into open application service with common interface. They have heard the cry of their clients for greater extensibility, flexibility, and scalability. They have also seen the value of fully leveraging their core technology products and services through open application service that optimize interoperability with other technologies. Open, interoperable application services are fast emerging, and flexible XML technology form their underpinnings.

1.2.3 Prospect

The next few years will see the rise of location services as an integral part of the New Information Economy. Governments, businesses and consumers will benefit from location as a unifying, foundational property that cuts across a wide spectrum of integrated applications and services. Powerful new location-based business services will dot the landscape of the Web. An entire new spectrum of mobile location services will emerge for the Palm Organizer, mobile phone, vehicle dashboard, and other mobile devices. The robust location services will be delivered to anyone, anywhere, anytime, and on any device.

Chapter 2 Challenges and Research Objective

2.1 Challenges

Nowadays, geographic data has been bounded up in the proprietary data format from many predominant and monolithic data providers. All those data formats are not interchangeable for most of GIS applications. Moreover, web service is rapidly becoming a paradigm for geographic data sharing, by leveraging Internet, the most widely available distributed platform. However, with certain devices or applications, the user is merely able to access the particular data resources. Especially in the scenario of LBS, the users is normally moving, which require more broad and extensive data sharing. Non-conformance is obviously becoming the major hurdle of LBS. Therefore, the interoperability for data providers is getting more imperative, which is expected to allow users to move transparently and smoothly among data providers.

The world of location based services demands standardization and openness just as for the Internet on which it will build.

2.2 Research Objective

This thesis work have been carried out following this research objective, which is to realize the interoperable and seamless geo web service interface for a big variety of data providers, in order to curb the hurdle of proprietary system and data format, and allow LBS client freely retrieving data with high automation from this common interface, by the mean of applying OGC specifications.

The research questions:

1. How is the web service deployed?
2. How do OGC specifications define the standard interface of web service?
3. How is XML and GML constructed?
4. To build a prototype system

Adhering the research objective and research questions, this paper starts with the introductory chapter to present some pertinent knowledge, like web service, OGC, XML and GML. And the prototype system design ensues to demonstrate some of the operations defined in WFS. The chapter of discussion discusses the data repository, coding work and pitfalls.

Chapter 3 Web Service for Geoprocessing

3.1 Introduction of Web-service

Nobody would dare to say that they don't know what is Web nowadays, while the Internet is rapidly becoming predominant in many realms of our society. The Web sites have been flourishing everywhere in recent couple of years, and the information it brings is incredibly huge yet growing surprisingly. The Internet Technologies are mightily reuniting the disparate information resources and communities to a virtual cyberspace, which is globally accessible. The URL is the address index on the Web, which enable users to locate where they are going. Behind your computer, type the URL in your Internet browser, subsequently the web sites show up. You may surf up and down at your pleasure, without knowing where they are geographically, how they transfer via cable. Indisputably, the Internet is pervading our life by many means and the impact is palpable for almost everybody.

However the web sits is absolutely not the only tangible incarnation of Internet. The web page is in fact simplest form of data flow on the platform of Internet. The Internet is in fact the distributed computation environment. The another modality of application that I am going to introduce is "web service". It is fundamentally the same like a web site but targeting differently.

A website is predominantly serving to end user who is surfing on Internet, in term of web sites or web pages, or any content as long as it could be parsed and presented in browser. But the web service is targeting on software developers and system integrators. It's rapidly becoming a broadly accepted approach for software producer to disseminate their products or services.

Web service is a term that comes more from the marketing domain than the technical domain. There are many companies or organizations have struggled to define it rigorously, and have not come to a consensus.

Here, I am attempting neither to introduce a precise definition according to my personal interpretation, nor to give you any heavy computer terminology or any technical jargon, like "service oriented architectures" and "loosely-coupled distributed systems," though both are important technical underpinnings of why web services come to being. I can foresee it's pretty hard for the audits that don't have pertinent background to digest the principle of the web services.

Let's take some examples. You developed an application that you want others to use. That is, you have a piece of software that initiates or accepts data transactions, provides or updates enterprise information, or perhaps manages the very systems and processes that make your operations run. You may want to make this accessible to people in other parts of your organization, or a business partner, or a supplier, or a customer.

I am really thinking here about software-to-software communication rather than the person-sitting-at-a-browser-talking-to-server-software situation, though it turns out that Web services can be used there as well.

This example is not something new. We are already able to achieve this by many conventional means, namely COM, Java Beans and ActiveX. Often at the same time, those approaches could be myriad of headache for IT technician. They cause intense IT staff training, difficulties for maintenance and troubleshooting.

With the advantages of Internet, the web service delivers greater value to your applications, or more clearly differentiate you from your predecessors. Web service reduces the complexity of this Software distribution, by bringing down dramatically the programming feat to a standard service that can be used over and over again, in different combinations. This means less training for IT staff, greater efficiency in building new applications and significantly greater and faster interoperability. It appears fairly appealing.

As web services get increasingly deployed as part of many software giant’s strategy, we will see greater automation and software that can dynamically locate and use the services they need.

The Internet is quickly evolving from today's Web sites that just deliver user interface pages to browsers to a next generation of programmable Web sites that directly link organizations, applications, services, and devices with one another. These programmable Web sites become more than passively accessed sites **-** they become reusable, intelligent web services. At last of this introduction, I quoted some words from Microsoft, for supplement.

“Web services provide a means for applications to communicate over networks as seamlessly as Internet Browsers do today. Their loosely coupled architecture means quick and easy development of enterprise and business-to-business applications in a manner that is not tied to a single operating system, platform, or development tool. These resources give you the information you need to create your own Web services, as well as the ability to consume other Web services.“

3.2 Establishment and Deployment

Web service gets increasingly deployed as part of many predominant software producers’ marketing strategies, like “DOTNET Framework” from Microsoft and the “e-business on demand” from IBM. They all provide their own solution for establishment and deployment of web service. Certainly you could never expect it would work as they asserted. Especially when you are going to apply them practically in your own transaction, the drawbacks would inevitably transpire. Here, I won’t try to judge or justify any products in this prosperous market, and I am not qualified either. Simply following my personal preference, I will concentrate on the pertinent technologies and concepts presented by Microsoft.

Web service, this term seems even suggest some similarity with web sites. Yes, it does indeed. In order to provide a web service, the web server is required. Even the service port is “80” that is exactly same as the conventional web sites. All the web service transaction is carried out in the conformance with HTTP protocol. The distinctive difference is all message exchanged with web service will be encoded into SOAP (Simple Object Access Protocol), instead of HTML.

“SOAP provides the definition of the XML-based information which can be used for exchanging structured and typed information between peers in a decentralized, distributed environment. SOAP message is formally specified as an XML Infoset, which provides an abstract description of its contents. Infosets can have different on-the-wire representations.

SOAP is fundamentally a stateless, one-way message exchange paradigm, but applications can create more complex interaction patterns (e.g., request/response, request/multiple responses, etc.) by combining such one-way exchanges with features provided by an underlying protocol and/or application-specific information. SOAP is silent on the semantics of any application-specific data it conveys, as it is on issues such as the routing of SOAP messages, reliable data transfer, firewall traversal, etc. However, SOAP provides the framework by which application-specific information may be conveyed in an extensible manner. Also, SOAP provides a full description of the required actions taken by a SOAP node on receiving a SOAP message.” [Reference 19]

So we now have a service we want others to use. Just because we build it, that doesn't mean others will know how to find it or interact with it. So we need a standard that says "this is how I write down how I talk to the service and what it says back to me." With that in hand, we can then standardize how to publish enough information about the service so that others can find it. Well, that’s why W3C developed WSDL.

“The Web Services Description Language (WSDL) is an XML-based format maintained by W3C used for describing a Web service in an implementation-neutral manner. It is the common thread that ties together the many different ways of creating and consuming Web services ”. [Reference 19]

3.3 Reliabilities and Securities

We're still not put the web service on use because we haven't dealt with reliability and security. Let’s get real with this cyber world. There are countless black eyes scattered around where Internet expanded, who call proudly themselves hacker. They are rat-likely sniffing, listening, tapping and scanning, to any possibly vulnerabilities of your system. They hold the contempt for any proprietary content and any private information can turn them on. What makes it worse is the restless loophole of commercial platforms. Don’t they know we are counting on their system?

Reliability and security is a significant concern that we have to fortify before we launch our web service. In this context, "reliability" means that when we send a message to a Web service we will know with certainty that either the message got there once and only once or else it did not get there at all. This is important: if somebody use a Web service to place a $1 million order for his manufacturing operation, he will have big problems if the order does not get placed or gets placed more than once. Certainly, calling someone of the phone doesn't count as a solution here!

Security for Web services is more sophisticated than the security we use for sending credit card numbers around the Web, although many people are using exactly that technology for their early Web services deployments. For Web services we want to be able to encrypt selected parts of the information we exchange and also digitally sign different portions.

To my knowledge, there are still very few mature solutions for this requirement. The IT industry is now working on standards to provide these kinds of Web services security features. They have a good roadmap of what needs to be done and we should be able to see products start to appear recently.

3.4 Web Service for Geoprocessing

I put a lot of effort on elaborating the web service, in order to make it clear that why web service attracted many attentions from other domains. In particular, the interests of geo related domain have been further aroused.

Admittedly, the web service is rapidly emerging as paradigm for World Wide Web-based distributed computing is tailor made for organizations that produce and use geospatial information. Besides, some organization, like Open GIS Consortium (OGC), are specifying the interoperability interfaces that enable software vendors to deliver geoprocessing services in the Internet's open, standards-based environment. This will make publishing, discovery, access and use of geodata and geoprocessing resources much easier and less expensive.

Let me illustrate some intrinsic advantage of web service, which enable it to be extraordinarily applicable in geoprocessing realm.

3.4.1 Distributed Geoprocessing

Web Services make the Internet a platform for delivering services, not just data. A "service" refers to software components that can be plugged together to build larger, more comprehensive services and/or applications. A service is a collection of operations accessible through API (application programming interface) that allows users to run ("invoke") a service, which could be a response to a simple request to create a map or a complicated set of image-processing operations running on several supercomputers. Examples of fundamental geospatial services are "get data" (vector or image), "portray data" (as a map), "locate a place," "transform coordinates," etc. I will explain more at next chapter regarding all those specifications.

Web Services are self-contained, self-describing, modular applications that can be published, discovered and invoked across the Web. After a Web Service is deployed on a Web server and made discoverable in an online registry or category of services, other applications--including other Web Services--can "find" and "bind" (i.e., discover and invoke) the deployed service.

3.4.2 Built to Simplify

The web service revolution undoubtedly alleviate the difficulties of many current application development, despite it sounds complex.

The promotion of standard interfaces reduces complexity and time to implementation, supports of multi-vendor and reduces product life-cycle risk. It also makes a perfect sense of the market, because it increases choice and reduces dependency on the more traditional single-vendor, monolithic application approach.

3.5 Conclusion

The Internet is more and more used for application-to-application communication. The internet-based programmatic interfaces are what the term of Web services was coined for. It’s indisputably a revolutionary way to disseminate data and service. Next chapter will tell more about the OGC specification of geo web service change the way we access and use GIS technologies.

Chapter 4 SPECIFICATIONS FROM OGC

The specification from OGC is an essential portion of my thesis work. In this chapter, I will introduce some primary concept of OGC, what are their specifications designed for and take an example to give you a rough idea of how the specifications operate.

4.1 Open GIS Consortium

4.1.1 Introduction

In fact the best way to get to know OGC is looking through their web site, http://www.opengis.org/, where you can find whatever you'd like to know, like the introduction, draft documentations, members and feedbacks. Sure, I could transcribe their description to my own word, my own English. Nevertheless, that would inevitably interfere your perception by my personal interpretation. Hence, I would rather preserve their original expression when it's extracted to this paper.

They have the wonderful introduction for themselves, as I cited following:

"*The Open GIS Consortium, Inc. (OGC) is a member-driven, non-profit international trade association that is leading the development of geoprocessing interoperability computing standards. OGC works with government, private industry, and academia to create open and extensible software application programming interfaces for geographic information systems (GIS) and other mainstream technologies.*"

I do think it's a good way to get familiar with them by the vision and mission they declared.

OGC Vision:

A world in which everyone benefits from geographic information and services made available across any network, application, or platform.

Approximately 80% of business and government information has some reference to location, but until recently the power of geographic or spatial information and location has been underutilized as a vital resource for improving economic productivity, decision-making, and delivery of services. We are an increasingly distributed and mobile society. Our technologies, services, and information resources must be able to leverage location, (i.e., my geographic position right now) and the spatial information that helps us visualize and analyze situations geographically. [Reference 11]

Products and services that comply to OGC's open interface specifications enable users to freely exchange and apply spatial information, applications and services across networks, different platforms and products.

OGC Mission:

Our core mission is to deliver spatial interface specifications that are openly available for global use.

Open interface specifications enable content providers, application developers and integrators to focus on delivering more capable products and services to consumers in less time, at less cost, and with more flexibility. [Reference 11]

4.1.2 What does OGC do?

In order to consolidate your perception of their activities, some part of their FAQ section clarified what OGC does.

OGC manages a global consensus process that results in approved interface and encoding specifications that enable interoperability among and between diverse geospatial data stores, services, and applications. In the OGC, geospatial technology users work with technology providers. Our membership is international and includes universities, Federal government agencies, local government agencies, earth imaging vendors, content providers, database software vendors, integrators, computing platform vendors and other technology providers. OGC facilitates their reaching agreement on OpenGIS® Specifications for interfaces, schemas and architectures. Systems implementing OpenGIS standards can interoperate, whether those systems are running on the same computer or the same network. OGC standards provide essential infrastructure for the Spatial Web, a network of geospatial resources that is thoroughly integrated into Web.

4.1.3 Why is the OGC necessary?

The OGC is necessary because cooperation is necessary to solve the difficult interoperability issues in the geospatial marketplace. Some user needs -- such as the need to share and reuse geodata in order to decrease costs, get more or better information, and increase the value of data holdings -- can only be addressed by cooperation among technology users and providers. The OGC brings together geoprocessing technology users and vendors and provides a formal structure for achieving consensus on specifications. No single vendor can "set the standard" that enables heterogeneous systems to interoperate in an open network environment like the Web.

Standardization is the reason for the success of the Internet, the World Wide Web, e-Commerce, and the emerging wireless revolution. The reason is simple: our world is going through a communications revolution on top of a computing revolution. Communication means "transmitting or exchanging through a common system of symbols, signs or behavior." Standardization means "agreeing on a common system." Someone needs to set standards to help people publish, discover, display, and use digital geospatial data. It serves both providers and users of geospatial technology to have an international, open, inclusive standards-setting process.

4.2 Why Specifications

Till here, you are supposed to come to know with what they are advocating: open standards. Why the specification is required and why the standards make sense?

4.2.1 Value of network

Network is actually a distributed computing environment that exhaustively utilizes system resources, by hooking up independent computers. The open standards are the fundamental to fully take advantages of the value of network. The open standards make it possible that heterogeneous applications can communicate cross platform, get over the barriers of proprietary data format.

One of the significant values of network is “serve once, use many”. Once the web service was established on the server, how many users that can access it depends only on the capacity of the server. That implies as long as the server can handle the request from users, the web service is going to be relayed to the destination users expected. Even the web service is capable to respond to many users’ request simultaneously, disregarding the limit of bandwidth and the ability of processor. This scenario I envisioned obviously entails the common standards or kind of protocol, which enable users to communicate with the web services through a unifying interface. From users’ perspective, nobody would concern what’s going to happen behind their screen, how their request will reach all the quite different web service and be responded. Therefore, there must be some common standards to be complied as we are deploying our web service.

Network has practically bestowed the interoperability between the applications or software components, which is produced or developed by different companies or organizations. They may form a formidable combination by interconnecting the third parties web service, to dramatically consolidate their functionalities. All those realizations are strongly dependant on the common standard.

In the compliance with the common standards, the web service is going to be more valuable as further the network expands. The merit will grow exponentially with the number of nodes.

4.2.2 Avoid the need to centralize

Conventionally, we store our data on a central server, which is supposed to be a powerful computer with huge storage. Consequently this central server will handle all the data transaction or even the data analysis. So we can see the capability of the central server strictly confines our web service performance. The decentralization of our data storage is expected to dramatically raise the performance of web service, by setting up a distributed and synchronized data-sharing environment. As for as the security is concern, the decentralization is able to reduce the risk of storing data on a central server. We have many synchronized data server working parallel. Any one of them encounters the exception, the rest is still working and the user connected will be taken over by others. Likewise, the data management is also going to be more secure and efficient.

How those distributed data server communicates or synchronize? Obviously, the common data format and protocol of data transaction is the imperative. And all operations between the data servers are relied on those common standards.

4.2.3 Vendor independent

Take a look at the current Geo-info market, and most software producers and distributors developed their own proprietary data format. Moreover, their analysis or geo-processing tools is also based on a proprietary data format. Whoever you choose, you will inevitably be attached to that single vendor to carry out your work. Even if you are going to update or extend your work environment, that vendor is your only choice, unless you dare to relinquish all the work have been done and start anew. Another case is that if you are going to integrate data that is from a different vendor, it will be turned out to be difficult.

We can see the significant drawback of proprietary format in this industry. To curb the hurdle of proprietary data format, the common standard or specifications is expected to be developed and applied universally.

4.3 OGC specifications

4.3.1 Introduction of OGC Specification Program

Because there are so many incompatible standards in the geo-information technology area, geospatial information and geoprocessing are not part of most information systems, and sharing geodata between geoprocessing systems and between user communities requires considerable time and expertise. Most of the standards attempt to normalize one of the following: 1) the encoding of information in software systems (data format standards and data transfer standards), or 2) the naming of features and feature relationships (data dictionaries), or 3) schemas for descriptions of data sets (metadata). Uniquely, OGC addresses the Babel-like profusion of data format and data transfer standards by creating open, common interfaces between software system components, and letting those systems use any data format internally. These OpenGIS Interfaces provide access to both information and functionality.

OGC also works to develop open software approaches that address inconsistent data dictionaries and metadata schemas.

In the OGC Specification Program, the OGC Technical Committee reviews specifications for interfaces and encodings developed either in the Interoperability Program by groups of members or through an internal proposals process. The Technical Committee and Planning Committee then approve these as OpenGIS Specifications for release to the public.

Open GIS Consortium devised a suite of implementation specifications to establish the interoperable geo web services, like WCS (Web Coverage Service), WMS (Web Map Service) and CAT (Catalog Interface).

CAT defines a common interface that enables diverse but conformant applications to perform discovery, browse and query operations against distributed and potentially heterogeneous catalog servers.

WMS provides four operations (GetCapabilities, GetMap, GetFeatureInfo and DescribeLayer) in support of the creation and display of registered and superimposed map-like views of information that come simultaneously from multiple sources that are both remote and heterogeneous.

WCS extends the Web Map Server (WMS) interface to allow access to geospatial "coverages" that represent values or properties of geographic locations, rather than WMS generated maps (pictures).

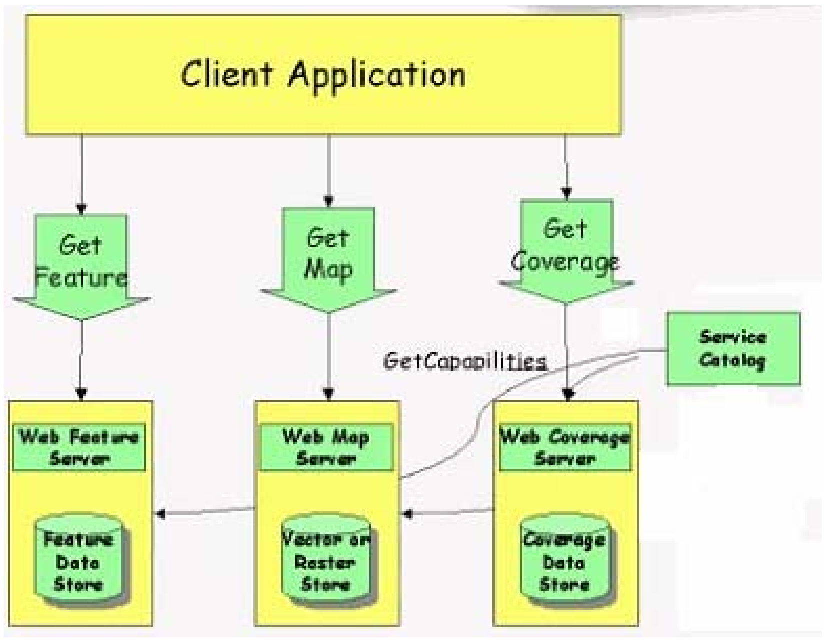


Figure 7.1

Figure 7.1 delineate the way how those web services intermingle and interoperate to serve to client application of data retriever. However, in this paper, I only put the emphasis on WFS. If you'd like to know more about the rest specifications, you may have a look at reference 11.

4.3.2 Web Feature Service Interface Specification

One important achievement of the OGC was the development of a large consensus around open web based interface specifications. Such specifications allow software vendors to implement their products using interoperable interfaces and provide end-users a larger pool of interoperable web based tools for geodata access and related geoprocessing services.

The OGC Web Feature Service allows a client to retrieve geospatial data encoded in Geography Markup Language (GML) from multiple Web Feature Services. The purpose of the Web Feature Server Interface Specification (WFS) is to describe data manipulation operations on geographic features using HTTP as the distributed computing platform. Such servers and clients can "communicate" at the feature level. See figure 4.1

Data manipulation operations include the ability to:

1. Create a new feature instance
2. Delete a feature instance
3. Update a feature instance
4. Get or Query features based on spatial and non-spatial constraints

A Web Feature Service (WFS) request consists of a description of query or data transformation operations that are to be applied to one or more features. The request is generated on the client and is posted to a web feature server via HTTP. The web feature server then reads and (in a sense) executes the request.

At server side, the WFS has to be established in a certain way cater requirement of geospatial data transaction. The requirements for a Web Feature Service are:

1. The interfaces must be defined in XML.
2. GML must be used to express features within the interface.
3. At a minimum a WFS must be able to present features using GML.
4. The datastore used to store geographic features should be opaque to client applications and their only view of the data should be through the WFS interface.
5. The use of a Xlink (Will be introduced in next chapter) expressions for referencing properties.

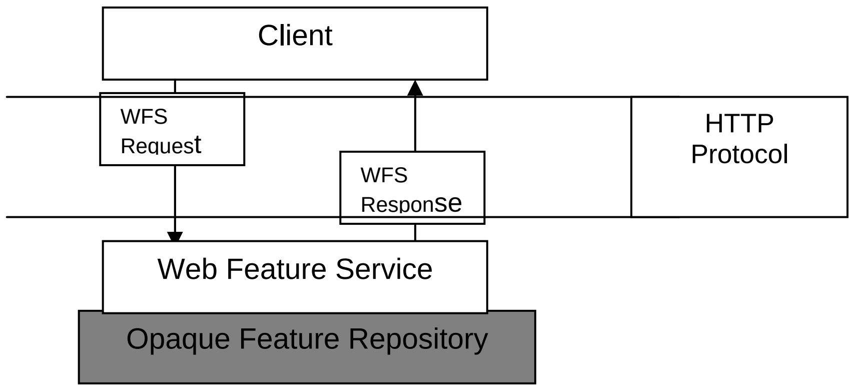


Figure 4.1

Now I am going to outline, in general terms, the protocol to be followed in order to process web feature service requests. Processing requests would proceed as follows:

1. A client application would request a capabilities document from the WFS. Such a document contains a description of all the operations that the WFS supports and a list of all feature types that it can service.
2. A client application (optionally) makes a request to a web feature service for the definition of one or more of the feature types that the WFS can service.
3. Based on the definition of the feature type(s), the client application generates a request as specified in this document.
4. The request is posted to a web server.
5. The WFS is invoked to read and service the request.

6. When the WFS has completed processing the request, it will generate a status report and hand it back to the client. In the event that an error has occurred, the status report will indicate that fact.

To support transaction and query processing, the following operations are defined:

1. GetCapabilities

A web feature service must be able to describe its capabilities. Specifically, it must indicate which feature types it can service and what operations are supported on each feature type.

2. DescribeFeatureType

A web feature service must be able, upon request, to describe the structure of any feature type it can service.

3. GetFeature

A web feature service 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 should be able to constrain the query spatially and non-spatially.

4. Transaction

A web feature service may be able to service transaction requests. A transaction request is composed of operations that modify features; that is create, update, and delete operations on geographic features.

5. LockFeature

A web feature service may be able to process a lock request on one or more instances of a feature type for the duration of a transaction. This ensures that serializable transactions are supported.

Based on the operation descriptions above, two classes of web feature services can be defined:

1. Basic WFS
A basic WFS would implement the GetCapabilities, DescribeFeatureType and GetFeature operations. This would be considered a READ-ONLY web feature service.

2. Transaction WFS

A transaction web feature service would support all the operations of a basic web feature service and in addition it would implement the Transaction operation. Optionally, a transaction WFS could implement the LockFeature operation.

Figure 4.2 is a simplified protocol diagram illustrating the messages that might be passed back and forth between a client application and a web feature service in order to process a typical transaction request. The elements referenced in the diagram are defined in this document.

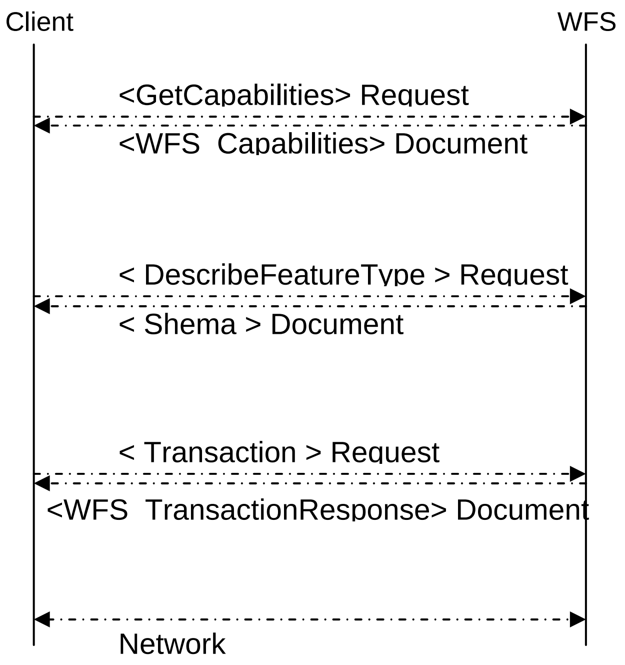

Client
WFS
<GetCapabilities> Request
<WFS Capabilities> Document
< DescribeFeatureType > Request
< Shema > Document
< Transaction > Request
<WFS TransactionResponse> Document
Network

Figure 4.2

4.4 Conclusion

This chapter gives a brief introduction of OGC and their WFS specification. Besides, why the specifications or open standards are required and how much interoperability will be delivered via those specifications or open standards is also presented roughly. However, how those specifications will be implemented still remains unclear. Concurrently, we all know that there is many heterogeneous geospatial data formats. In order to implement the specifications or the open standards, a common data format is imperative. That's is exactly what I am going to introduce in next chapter.

Chapter 5 XML and Geodata

5.1 Introduction

In this section, I am going to examine what are Extensible Markup Language, its advantages and the basics behind it. It also describes the XML structure, and the relation with Document Type Definition, Schema, Document Object Model, and stylesheets, which are common in any XML application. Besides, two already standardized geodata XML applications are described: Geography Markup Language (GML) and Scalable Vector Graphics (SVG).

5.2 XML

5.2.1 what is XML

During the past few years, a new standard for data exchange has been defined, the eXtensible Markup Language (XML). XML is not one predefined or proprietary data format such as DXF (Autocad Exchange Format) or VRML (Virtual Reality Modeling Language). It is a structural and semantic language, which allows for description the encoded information. As it is a meta language, it allows for defining other markup languages such as Mathematical Markup Language (MathML), Chemical Markup Language (CML) or for example Geography Markup Language (GML), a standardized XML based format for the management of geodata.

XML is a markup language, designed to structure, store and send information over the Web. It provides no predefined tags, as in the case of HTML, but provides standards so that the user can define his own tags and document structure. Hence XML is free and extendible. Furthermore, as XML is in plain text format, it provides a software and hardware independent way of sharing data. It enables data to be accessed by all kind of “reading machines” or processors.

Generally speaking, an XML document consists of three different logical structures: tags, Attributes and Data. A simple document can look like the following example:

```
<person gender=”male”>
<firstname>Koen</firstname>
<lastname>Ma</lastname>
</person>
```

Figure 5.1

From figure 5.1, we can see that the tag names are written between ‘<’ and ‘>’ and either occur pair wise as start and end tag or, for empty elements, as an empty element tag. Start tags may include attributes. The corresponding value has to be limited with quotes. So the data can be stored as attribute values as well. However, normally attributes are used to hold some meta information which is used for several purposes by the parser that interprets the document.

Since XML documents are in plain text and structured (encoded) with user-defined tags, it does not do anything without some kind of software. Therefore it has to follow some standards in encoding data to enable decoding by some other programs. For this XML adheres to the standards specified by the XML specifications. At present (October 2002) XML Specification 1.0 is the World Wide Web Consortium (W3C)’s implementation recommendation. XML documents must follow standard rules including the syntax for marking up and the meaning behind the markup. What is a valid markup is defined by a Document Type Definition (DTD) or alternatively by an XML Schema.

The valid structure of an XML document can be defined using two different mechanisms: Document Type Definitions (DTD) and XML Schema Definitions (XSD). As XML Schemas are standardized by the W3C and they are defined in XML and provide more flexibility than DTDs, they are generally used to define XML applications.

Those data format standards is widely accepted and applied in many industrial areas. XML used as data exchange format has gradually reached the maturity. Although some of the mentioned technologies are brand new, they get more and more important due to the fact that they are already implemented in standard software such as Microsoft Internet Explorer and ready to be applied universally.

5.2.2 XML Document Structure

There are certain rules that have to be adopted while authoring XML documents and these can be summarized as follows:

• XML documents need a declaration at the top to signal what they are.
• Every XML document must have a root element (tag) that encloses the content.
• Every start tag must have a closing tag.
• Tags must nest cleanly.
• Empty tags have a different form to make it clear that these are tags with no closing tags.
• All attribute values must be in quotation marks.
• Tags are case sensitive and must match.

The XML documents must be precise, those do not comply with these rules can not be processed by the XML parsers embedded in browsers or standalone processors. An XML document that conforms to these rules specified in XML specification, as determined by an XML parser is classified as well-formed. An XML Parser is a software program that creates the Document Object Model. I will talk more about it later in this section.

An XML document mainly consists of two parts; prolog and body. The prolog contains the declaration, and the body contains the actual marked up document. The content of both parts (whole document) is composed of declarations, elements, comments, character references, and processing instruction, all of which are indicated in the document by explicit markup. An example XML document is given below:

```
<? xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE Thesis SYSTEM "Cong_Thesis.dtd">
<!- - An XML Example - ->
<Cong_thesis>
<title>Geo Web service</title>
<pub_date>3-12-2004</pub_date>
<chapter>Introduction
<para>What is GML</para>
<para>What is HTML</para>
<para>What is XML</para>
</chapter>
<chapter>XML syntax
<para>Elements must have a closing tag</para>
<para>Elements must be properly nest</para>
</chapter>
</Cong_thesis>
```

Figure 5.2

The first line of an XML document is a declaration, which notifies that the document has been marked up as an XML document. The XML declaration itself is a processing instruction and therefore it begins with <? and ends with ?>. The version attribute indicates the version of XML specification that the document complies with and the standalone attribute specifies whether the document has any markup declarations that are defined in a separate document. Thus, value “yes” implies no markup declarations in external documents and “no” leaves the issue open. The document may or may not access external documents. The encoding attribute denotes the character encoding system used in the document. The DOCTYPE declaration (second line in the example) declares the name, type and location of the related Document Type Definition (DTD), which will be further elaborated later.

Elements are the basic unit of XML content. An element consists of a start tag, and end tag, and everything in between. Anything between a < sign and a > sign is a tag except that is inside a comment or a CDATA section. Relationships in XML elements are expressed in terms of parents and children. In the given example “thesis” is the root element. Title, pub date, production, and chapter are child elements of “thesis” element and thus it is the parent element. Title, pub date, production, and chapter are siblings (or sister elements) because they have the same parent. Elements can have different content types such as elements that contains other elements like “thesis” element, mixed that contains both text and other elements as “chapter” element, simple that contains only text like “para” element in the example, or empty that contains no information like “production” element. In case of empty elements no closing tag appears.

The comments are the character data in an XML document that XML processor ignores. The comments follow the syntax of <!- - content - ->. The content of the comment should not have ”-” or ”–” characters that might confuse XML parser and also a comment should not be placed within a tag and cannot be nested.

5.2.3 Document Type Definition (DTD)

The validity of markup of XML document is handled by DTD (Document Type Definition). It is a file (or several files to be used together) with “dtd” extension, written in XML’s Declaration Syntax, which contains a formal description of a particular type of document. DTD sets out what names can be used for element types, where they may occur, and how they all fit together. For example the DTD of the above XML document is as figure 5.3:

```
<!ELEMENT Cong_Thesis (Chapter+ | ANY)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Pub_date (#PCDATA)>
<!ELEMENT Production (EMPTY)>
<!ATTLIST Production
id NMTOKEN #REQUIRED
media CDATA #IMPLIED>
<!ELEMENT Chapter (Para+ | #PCDATA)>
<!ELEMENT Para (#PCDATA)>
```

Figure 5.3

In DTDs all keywords must be in UPPERCASE, such as ELEMENT, ATTLIST, #REQUIRED etc.. However user defined elements and attributes can be any case as users choose, as long as they are consistent. A DTD can either be included as part of a well-formed XML document, (standalone=“yes”), or it can be referenced from an external source, (standalone=“no”). When external DTD is referenced, the SYSTEM attribute whose value indicates the location of DTD has to be added to the DOCTYPE declaration. Thus in order to reference an external DTD, both XML declaration and DOCTYPE declaration has to be changed. An XML document that conforms to the rules of a DTD is called a valid document. A valid document is necessarily well formed. When an XML document is parsed by a processor, it firstly to parse the DTD and then read the document to identify where every element type comes from and them related. Using a DTD when editing documents preserve them consistent and valid.

5.2.4 XML Schema

XML Schemas are a W3C Recommendation for defining the structure, content and semantics of XML documents. It is an alternative to DTD written in Schema Definition Language, which is an XML language for describing and constraining the content of XML documents. In general terms, an XML Schema describes how data is marked up and these files are given with “xsd” extension. XML Schema offers many advantages over DTD. One of the greatest advantages is the support for data types. Since DTDs are designed for use with text, they have no mechanism for defining the content of elements in terms of data

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
<!--An XML Schema Example -->
<xsd:element name="Cong_thesis" type="thesisType"/>
<xsd:complexType name="thesisType" mixed="true">
<xsd:sequence>
<xsd:element name="title" type="xsd:string"/>
<xsd:element name="pub_date" type="xsd:date"/>
<xsd:emement name="production" type="productionType"/>
<xsd:complexType name="productionType">
<xsd:complexContent>
<xsd:restriction base="xsd:integer">
<xsd:attribute name="id" type="xsd:positiveInteger"/>
<xsd:minInclusive value="0"/>
<xsd:maxInclusive value="1000"/>
</xsd:restriction>
</xsd:complexContent>
<xsd:attribute name="media" type="xsd:string"/>
</xsd:complexType>
<xsd:element name="chapter" type="chapterType"/>
<xsd:complexType name="chapterType"/>
<xsd:sequence>
<xsd:element name="para" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
</xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

Figure 5.4

The <schema> element is the root element of every XML Schema where “xsd” is the namespace prefix. The fragment xmlns:xsd="http://www.w3.org/2001/XML Schema" indicates that the elements and data types used in the Schema come from the "http://www.w3.org/2001/XMLSchema" namespace. The elementFromDefault="qualified" fragment indicates that any element used by the XML instance document which will declare in this schema must be namespace qualified. All namespaces used in the schema must be declared in prolog and all elements and data types must be defined in the body.

An XML Schema document consists of four basic constructs: declaration of elements, definition of types (simpleTypes and complexTypes), the possibility to define subtypes by extending or restricting supertypes, and use of aliases (substitutionGroup mechanism).

An XML element that contains only text(data in any type) is a simple type element. It cannot contain any other elements or attributes. But it can have a default value or fixed value set. Simple element is defined by <xsd:element name="Name of Element" type="Data Type" default="Default Value" (or fixed="Fixed Value")/>. Common data types in XML Schema are string, decimal, integer, boolean, date and time. The type of a simple element is defined by <xsd:simpleType>.

In Schema an attribute is always declared as a simple type and an element with attributes always has a complex type definition. An attribute is defined by, <xsd:attribute name="Attribute Name" type="Data Type"/>. Attributes also can have a default or a fixed value specified. Even though all attributes are optional by default, with ”use” attribute it can be explicitly specify whether it is ”optional” or ”required”.

Defining a type for XML element or attribute imposes a restriction for the element or attribute content. With XML Schemas, it is able to add own user restrictions to user XML elements and attributes. In the example restriction has imposed on id attribute. The id can have only integer values between 0 and 1000 (including that numbers). Likewise to limit the content of an XML element to a set of accepted values, the enumeration constraint can be used. All restrictions that can be applied to data types are given in XML Schema Specification. Moreover types can be defined by extending existing types.

An XML element that contains elements, mixed content, empty content or attributes is considered to be a complex type element. For example, thesis element is a complex one. Its type has been defined as thesisType separately. The child elements of thesis element are surrounded by the <xsd:sequence> indicator. Separate defining complex types offer more flexibility, because these types can be used by other elements too.

5.2.5 The Document Object Model (DOM)

The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents. The document can be further processed and the results of that processing can be incorporated back into the presented page.

XML standards include specifications of how an XML document should be parsed and represented within any computer irrespective of type or operating system. This internal representation (tree representation) of an XML document, which is generated within a computer by an XML parser is called the Document Object Model (DOM). DOM allows a single document to be accessed in the same way by different applications running on different computer platforms through XML tag references. In a software context, the DOM is a programming API (Application programming Interface) for an XML document, which defines the logical structure of documents, and the way a document is accessed and manipulated. It details the characteristic properties of each element of a document, thereby detailing how these components can be manipulated and, in turn, manipulating the whole document. Therefore with the DOM, programmers can create and build documents, navigate their structure, and add, modify, or delete elements and content. As a W3C specification, one important objective for the DOM is to provide a standard programming interface that can be used in a wide variety of environments and applications. The DOM can be used with any programming language and provides precise, language-independent interfaces. [Reference 19]

5.3 GML

5.3.1 Background

What is GML? It is a set of XML technologies for handling geo-spatial data, endorsed by more than 220 companies and agencies around the world. It is a standard created by OpenGIS, of which IONIC is co-author, and is converging now as a standard for W3C and ISO. As an example, Ordnance Survey in UK has decided and is already distributing their data as GML.

To keep it simple, GML is basically:

- XML encoding of geography

- Enables GI community to leverage the whole world of XML technology
- Provides vector mapping in standard Web applications
- Enables complex features, feature collections and feature associations

Therefore, GML is very useful as an encoding language for geographic information and data modelling (using XML schema). GML can be read and understood by people and is open (not proprietary), yet powerful. It can thereby significantly reduce cost of ownership of geo-spatial data. GML can also easily be mixed with temporal and/or non-spatial data that are available on the Web including text, imagery, video, events, etc. Finally GML can be used to build shareable application schema that can be used across community of users.

There is the definition from OGC:

"The Geography Markup Language (GML) is an XML encoding for the transport and storage of geographic information, including both the spatial and non-spatial properties of geographic features".

In this definition the expression 'XML encoding' is a synonym for 'XML application'. Since I have poured lots of time on XML, it could be extraordinarily instrumental to comprehend the specification of GML. I am hereby not going to copy&paste so much verbose description of it. In stead, only some exceptional issues will be introduced, which I think deserve our additional attention. An online version of the OGC Implementation Specification of GML can be found at the OGC web page, from where you may know much more if you like exploring.

5.3.2 Key Issues of GML

The GML standard enables to model the world according to the OGC Abstract Specifications, which define a geographic feature as "an abstraction of a real world phenomenon; it is a geographic feature if it is associated with a location relative to the Earth.". The GML specification is concerned with the OGC Simple Features, features whose geometry properties are restricted to ' simple'. Point, line, linestring or arc is good examples for such objects. Unlike the simple feature specification, GML allows for 3D coordinates, but it does not directly support 3D geometry constructs. By the way GML is fully compliant to the XML Schema and Namespace Recommendation. The general definition of GML is done using a set of three base schemas:

- Feature Schema - feature.xsd: defines the general feature-property model
- Geometry Schema - geometry.xsd: includes the detailed geometry components
- XLink Schema - xlink.xsd: provides the XLink attributes used to implement linking functionality

As one of the key issues of XML applications, GML supports full extensibility. So it allows for definition of individual types by extending the given schemas. Another benefit of XML is the separation of content and presentation. Therefore GML does not regard how to visualize the stored data. But as described in former chapters, there are several XML relevant mechanisms to transform GML documents into comfortable visualization formats such as SVG or X3D, an XML application that can easily be transformed into VRML, the commonly used visualization format for web presentation of 3D data.

5.3.3 Mechanisms of GML for Data Interoperability

GML specification is an important step taken by the geospatial community towards the vision of widespread spatial data interoperability. GML-based geographical databases can communicate with each other. The mechanisms of GML for data interoperability is given as following:

1. GML provides a common schema framework for encoding geo-spatial features. GML uses the W3C XML Schema Definition Language to define and constrain the contents of its XML documents. The GML v2.0 Specification provides two basic XML Schemas: the GML Feature Schema (feature.xsd) and GML Geometry Schema (geometry.xsd). Users can develop their own application schemas according to GML v2.0 Specification conformance requirements. GML schemas reconcile the need for standardization with the need for diversity by providing a standard means of extending the GML format. The direct consequence of applying schemas with GML is that it becomes possible for organizations to define formats to suit their needs and exchange geographic information without the need to involve software developers to create translators for that format. This has impacts on both the cost and risk of exchanging data.

2. While GML builds on XML Schema, it provides a more constrained schema. GML is based on a common abstract model of geography, which describes the world in terms of features. A geographic feature is an abstraction of a real world phenomenon; it is a geographic feature if it is associated with a location relative to the Earth. A feature has both simple properties and geometry properties. Simple properties refer to the usual name, type and value description. Geometry properties are composed of points, curve (linestring) and surface (polygon). By looking at feature schemas and properties, one can readily compare features and integrate data.

3. GML is based on an XML standard. XML is a universal format for structured documents and data on the Web. XML is easy to transform. Using XSLT or almost any other programming language (VB, VBScript, Java, C++, Javascript), users can transform XML from one form to another. By adhering to an open, non-proprietary standard, GML documents can be manipulated, transformed and presented in the same flexible way as XML contents.

4. GML provides XLink and XPointer mechanisms as does XML. The linking mechanism of HTML (one web page linking to another), is one of the key foundations of the Web. GML goes further by providing a mechanism for linking multiple distributed resources into a complex association. As HTML is important to the Internet as a linked collection of web pages, GML can enable the development of a Geo-Web as a linked collection of geo-spatial features. Through Xlink and Xpointer, different features and feature collections, which may be located remotely, can be associated together at the feature level. XLink and XPointer hold great promise for building complex and distributed geographic data sets. They make it possible to access and seamlessly integrate data from different departments, cities, states and countries.

5. GML provides a means to transport geospatial data over the Web. With the help of Web Feature Server (WFS), spatial databases with different formats can transparently communicate with each other by being converted to GML-format data on the fly. As XML is an important Internet data transport technology, GML makes it possible for real-time data access and transport in the Internet environment at feature level. When the GML-marked geospatial data are transported, all the markup elements that describe every spatial and non-spatial features, geometry and spatial reference systems of the data are also transported to the recipient.

6. GML data are stored in plain text. Text is vendor-neutral, so information stored in GML is not locked into a proprietary binary format. Since GML is text-based, it can readily integrate geospatial data with a wide variety of non-spatial data types including text, business transactions, graphics, audio, voice and more. This capability would greatly enhance the value and accessibility of geospatial information. For example, users can easily insert a map in a financial report, or vice versa. In addition, as a text format, GML can be easily transmitted across a variety of platforms over the Internet. Thus GML enables disparate systems to share information easily.

Those GML characteristics allow users to build a large, global map stored and processed in a scalable and redundant distributed architecture. GML makes it possible that all spatial data in the whole world can be integrated into one map. The inherent transformability and accessibility of GML opens a new domain for the geo-community.

5.3.4 Advantages and Shortcomings

GML as an XML application provides the main advantages of XML such as vendor independency, interoperability, extensibility and many more. In the current status it is realized to support the properties of the OGC Simple Features with some extensions such as the capability to handle 3D coordinates.

The possibilities to manage complex geometry data types are rather limited considering only the given definitions. But as GML supports extensibility, complex data types can be defined using collections of simple ones. Regarding large topographical datasets such as point clouds, problems due to the large overhead within XML datasets do occur, caused by the numerous tags and the included meta information. Some promising solutions to these problems will be described later on.

5.4 SVG

A very efficient XML standard for storage, exchange and especially web presentation of geodata is Scalable Vector Graphics (SVG), which is standardized since September 2001 by the W3C. The current version of the specification can be found on the web. [Reference 19].

SVG allows for interactivity and animation and it is compatible to other XML standards such as DOM (Document Object Model) –supports access of the objects entities), XSL (XML Stylesheet Language), SMIL (Synchronized Multimedia Integration Language) and many others. So it is no problem to transform a GML document, which can be the result of a database request for example, into a valid SVG document using XSLT in order to display the resulting document in a common web browser.

5.4.1 Geometry Types

As Vector is part of the standards name, it has the capability to store vector data. However, there are three different groups of object types that can be represented using this standard:

- Vector Data
- Raster Images
- Text

The following basis vector geometry types are defined: rectangle, circle, ellipse, line, polyline, polygon, symbol and path. Within this set of object types, path is the most complex one, because it has the capability to represent line segments, square and cubic Bezier curves and arc segments. To each object type different style attributes can be defined such as line-style, weight, color and so on. The integration of raster overlays allows using raster images in combination with the corresponding vector data. They can be treated like any other geometry object using the same methods. Text objects have the same status as any other basic geometry type. They can be modified using the same attributes.

5.4.2 Advantages and Shortcomings

Scalable Vector Graphics have many abilities to become the standard for geodata representation and visualization on the web. It provides many useful features to handle this kind of data in a quite comfortable way. It is an open and vendor independent standard, and software implementations to enable common used browsers to display such datasets do exist. But they are only available as plugin provided by Adobe, one of the SVG standardization group members. As long as the SVG functionality is not included into the browsers kernel, it is not available to everyone using this software – considering the problems of installing a plug-in on a computer where the current user has no administrators right – and it is rather slow; a real bad ability considering larger datasets.

Within the community of web cartographers there is a large acceptance for this standard. Especially the Swiss Federal Institute of Technology in Zurich, Switzerland, forces its development and application. They established a yearly conference, the SVGOpen (http://www.svgopen.org), which had about 150 participants in the year 2002. But as long as there are no outstanding software solutions that force the developers of web browsers to include SVG viewers into their standard systems, it will not be done. And as long as this viewing functionality is not available to everyone, most software developers hesitate to invest lots of implementation efforts because they are not sure, if the given standard will be accepted. This is a vicious circle, which might have the capability to ruin the impressive capabilities of a standard like SVG. Therefore further work and development has do be done to bring this standard to the public.

Another shortcoming is the fact that SVG only supports 2D geometry elements. For cartographic visualizations in the web, this does not matter. But SVG is no usable format for management and storage of 3D geometries.

5.5 Compressing XML

XML is stored in readable ASCII-Format and every dataset is described using tags. Therefore a large overhead occurs within the datasets. Considering several Gigabytes of data, which might occur within one single laser-scanning project, this fact will cause problems. For example, 600 Millions of points stored in latitude / longitude / height at a precision of 5 digits in lat / long and 3 in height require 16,8 GB disk space without XML tags; converting this point list into an appropriate XML document might increase a big amount of the data.

The transformation and interpolation of the irregular original points into a regular grid array that can be stored efficiently as binary data would reduce this problem. But the high detailed information that is available within the original points is lost. So in many cases this is not a sufficient solution. Nevertheless some considerations on this topic are described in the following chapter. Therefore data compressing might be a better choice to solve this problem. But as data compression is a rather time consuming process, this will only be a good deal for non time critical problems. There are several well-known and often implemented compression algorithms such as Huffman Encoding or Lempel-Ziv Compression (often mentioned as LZ1 or LZ2 compression which are shortcuts of the two widely used implementations of this algorithm). Both algorithms are lossless, for sure, as loss-compression only makes sense in the case of image, video or audio data compressions for certain applications.

Compression algorithms, as the two mentioned above, do a good job on ASCII data and so they also have the capability to compress an XML dataset in an acceptable manner. But as an XML document has a very large overhead, the compressed data set is still larger than the compressed original text file by an amount of about 20 %. So some further pre-processing is necessary.

As one can imagine, there are many similar structures within an XML document. So it is a good idea to sort the elements of the document according to their tag name. This process is called reversible transformation. As we do not want any kind of loss of data, this has to be reversible for sure. Afterwards several other steps are applied. For example, the tag names are stored within a table only once and instead of the original tag name only a unique identifier replaces them within the document. After applying such preprocessing methods, a compression algorithm is applied to this transformed data. One might argue, that this pre-processing is a very memory and CPU capacity intensive process and that is true. To solve this problem, good implementations of XML compression tools allow for splitting a large document into smaller parts and process them sequentially.

An example of an efficient compressor for XML is the software product XMill. This free and open source software is available on the web (http://www.research.att.com/sw/tools/xmill/).

The amazing fact on this implementation is, that the compressed result of an XML document might have only half of the size of the compressed original text file without XML tags! This is enabled, because the used compression method relies on run-length encoding as well. This kind of algorithm is able to compress equal entries if they follow each other. As the XML document was resorted according to its tag names, it is very likely that equal entries are grouped together.

5.6 Binary XML

Another possibility to get rid of the overhead of metadata within XML documents is to combine the advantages of XML data description and binary data storage. Such XML applications are called Binary XML. Most common data formats such as TIFF or MPEG store some descriptive information in a binary header, which is mostly stored as prefix of the actual dataset. This meta information describes for example the extension, the colour table, the source or many other data relevant things.

Binary XML uses an XML document to store this descriptive information in ASCII format and the main dataset is either included as binary data stream within this XML document or it is referenced using a link within a tag. External storage of the binary dataset provides the possibility of direct data access using efficient data manipulation routines. Unfortunately there are no specifications on Binary XML neither in the GML nor in the SVG specification for storage of geometry features in binary format. Obviously raster images are binary data repositories that can be linked to SVG documents using XLink. Nevertheless, it would be a good opportunity to extend these standards for binary geodata handling and enable efficient use of XML dealing with large topographic datasets. In that case, the rather short meta data information could be stored using standardized XML formats.

Chapter 6 Prototype System Implementation

6.1 Introduction

OGC specifications literally provide a common interface for geodata sharing and transporting. As the specifications are being progressively implemented in this industry, we can foresee a significant interoperability throughout full scope of data transaction. I have depicted some pertinent background and technologies in previous introductory chapters. In this chapter, I will attempt to design a succinct program to establish and realize the web service in compliant with the specification OGC Web Feature Service.

6.2 System Design

The emphasis will be put on the data transaction via this web service. To support transaction and query processing, OGC WFS have 5 operations is defined. Since this system is merely designed to demonstrate, I only implemented 3 operations as following:

1. GetCapabilities
2. DescribeFeatureType
3. GetFeature

The above three operations are primary data transaction in most occasion. Figure 6.1 is the designed system structure.

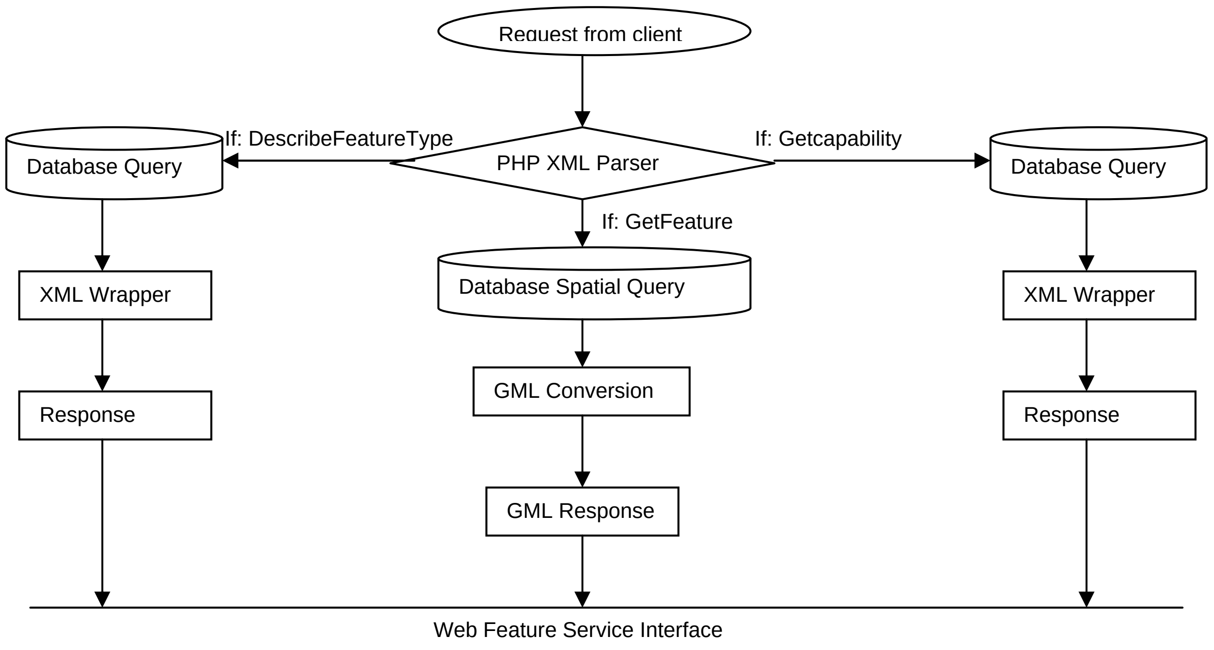


Figure 6.1

Since PHP script has mighty and powerful XML manipulation functionalities, it certainly is a good option for me to build this system. Besides, PHP engine is universally available on most of Web Server, so the system deployment could be done readily anywhere.

6.3 Service establishment

For the convenience of my work and demonstration, the Win2000 system would be an ideal platform. The web server of IIS is built-in component of Win2000. In order to execute PHP script, the PHP engine needs to be compiled with IIS. To connect Mysql database is a breeze since PHP is genetically capable to communicate with Mysql.

Now the service is listening. All the request from client mobile devices will be handled and responded accordingly, on one condition that the request is formed in compliance with WFS specification.

There are two methods of data transaction defined in HTTP protocol, POST and GET. In this demonstrative system, only POST method could be correctly parsed. The following examples elaborate how the service is implemented.

For sake of demonstration, a client program was also built running in a PDA emulator. Figure 6.2 is a screenshot of the emulator.

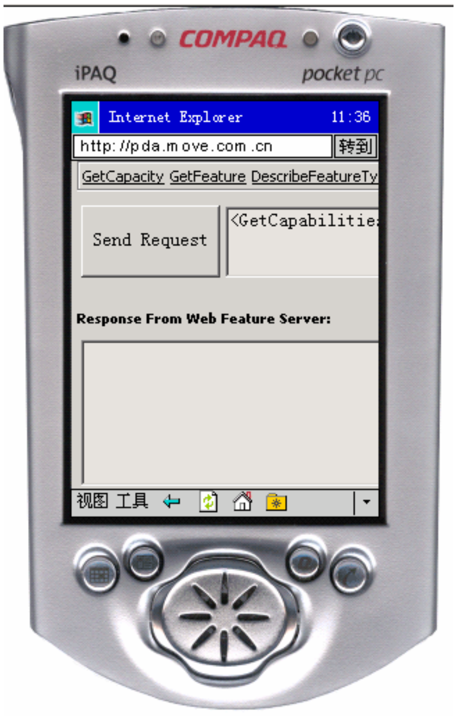


Figure 6.2.

6.4 GetCapabilities Operation

6.4.1 Introduction

The <GetCapabilities> element is used to request a capabilities document from a web feature service.

6.4.2 Request

The request is defined by the following XML Schema fragment:

```
<xsd:elementname="GetCapabilities" type="wfs:GetCapabilitiesType"/>
        <xsd:complexType name="GetCapabilitiesType">
                <xsd:attribute name="version" type="xsd:string" use="optional"/>
                <xsd:attribute name="service" type="xsd:string" use="required" fixed="WFS"/>
        </xsd:complexType>
```

The service attribute is described in OGC specification. The version attribute, unlike the normal case, is not mandatory since a client application may not have knowledge about what versions a server may support.

6.4.3 Response

The schema of the response to a GetCapabilities request is normatively defined using XML Schema in OGC specification, http://www.opengis.org/docs/02-058.pdf.

The capabilities' document comprises of four sections:

1. Service section

The service section provides information about the service itself.

2. Capabilities section

The capabilities section specifies the list of requests that the WFS can handle.

3. FeatureType list

This section defines the list of feature types (and operations on each feature type) that are available from a web feature service. Additional information about each feature type is also provided in this section.

4. Filter capabilities section

The schema of the Filter Capabilities Section is defined in the Filter Encoding Implementation Specification [3]. This is an optional section. If it exists, then the WFS should support the operations advertised therein. If the Filter Capabilities Section is not defined, then the client should assume that the server only supports the minimum default set of filter operators as defined in the Filter Encoding Implementation Specification [3]. The relevant document could be found on OGC website. This section is not available in my demo system.

6.4.4 Data flow

The Figure 6.3 depict the programming model.

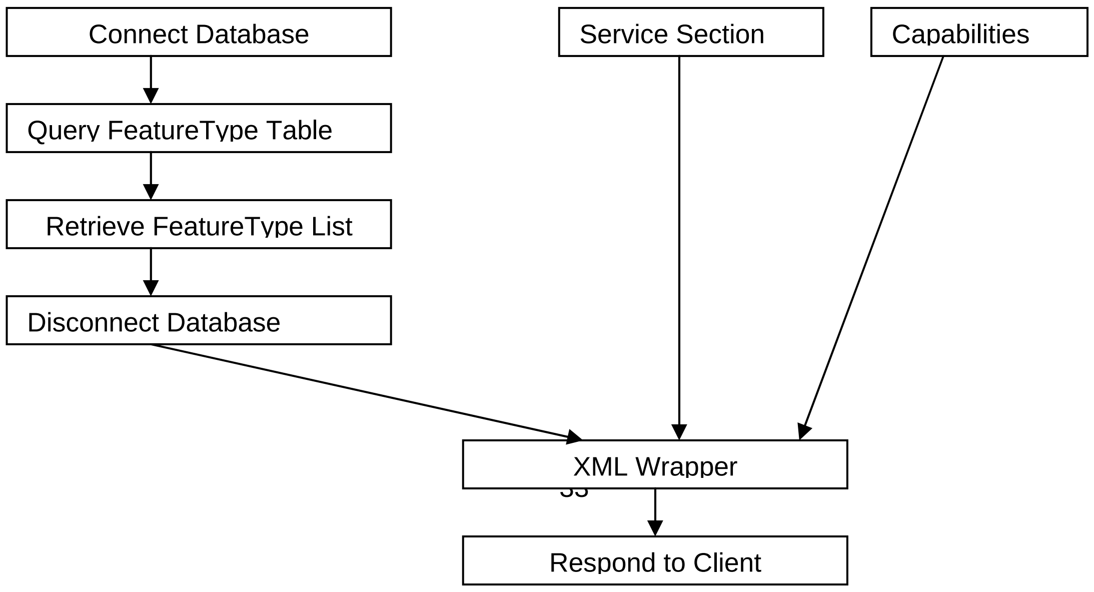
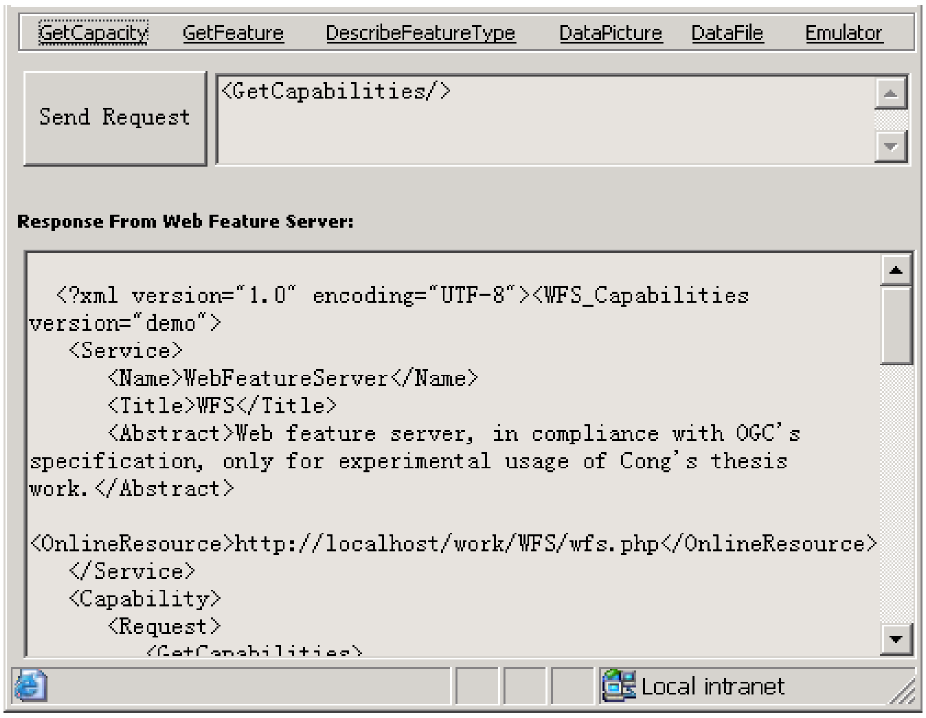


Figure 6.3

6.4.5 Example

Figure 6.4 is the screenshot of this operation.



```
<GetCapabilities/>
```

```
  <?xml version="1.0" encoding="UTF-8"><WFS_Capabilities
version="demo">
   <Service>
      <Name>WebFeatureServer</Name>
      <Title>WFS</Title>
      <Abstract>Web feature server, in compliance with OGC's
specification, only for experimental usage of Cong's thesis
work.</Abstract>

<OnlineResource>http://localhost/work/WFS/wfs.php</OnlineResource>
   </Service>
   <Capability>
      <Request>
```

Figure 6.4

This demo service would generate the following code for a request of <GetCapabilities/>.

```
<?xml version="1.0" encoding="UTF-8"><WFS_Capabilities version="demo">
  <Service>
    <Name>WebFeatureServer</Name>
    <Title>WFS</Title>
    <Abstract>Web feature server, in compliance with OGC's specification, only for experimental
usage of Cong's thesis work.</Abstract>
    <OnlineResource>http://localhost/WFS/wfs.php</OnlineResource>
  </Service>
  <Capability>
    <Request>
      <GetCapabilities>
        <DCPType>
          <HTTP>
            <Post onlineResource="http://localhost/WFS/wfs.php"/>
          </HTTP>
        </DCPType>
      </GetCapabilities>
      <DescribeFeatureType>
        <SchemaDescriptionLanguage>
```

```
          <XMLSCHEMA/>
        </SchemaDescriptionLanguage>
        <DCPType>
          <HTTP>
            <Post onlineResource="http://localhost/WFS/wfs.php"/>
          </HTTP>
        </DCPType>
      </DescribeFeatureType>
      <GetFeature>
        <ResultFormat>
          <GML2/>
        </ResultFormat>
        <DCPType>
          <HTTP>
            <Post onlineResource="http://localhost/WFS/wfs.php"/>
          </HTTP>
        </DCPType>
      </GetFeature>
    </Request>
  </Capability>
   <FeatureTypeList>
    <FeatureType>
      <Name>Country</Name>
    </FeatureType>
    <FeatureType>
      <Name>Mjrivers</Name>
    </FeatureType>
    <FeatureType>
      <Name>Cities</Name>
    </FeatureType>
   </FeatureTypeList>
</WFS_Capabilities>
```

6.5 DescribeFeatureType Operation

6.5.1 Introduction

The function of the DescribeFeatureType operation is to generate a schema description of feature types serviced by a WFS Service. The schema descriptions define how a WFS service expects feature instances to be encoded on input and how feature instances will be generated on output.

6.5.2 Request

The DescribeFeatureType element contains zero or more TypeName elements that encode the names of feature types that are to be described. If the content of the DescribeFeatureType element is empty, then that shall be interpreted as requesting a description of all feature types that a WFS can service. The XML encoding of a DescribeFeatureType request is defined by the following XML Schema fragment:

```
<xsd:elementname="DescribeFeatureType" type="wfs:DescribeFeatureTypeType"/>
<xsd:complexTypename="DescribeFeatureTypeType">
        <xsd:sequence>
                <xsd:elementname="TypeName" type="xsd:QName" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attributename="version" type="xsd:string" use="required" fixed="1.0.0"/>
        <xsd:attributename="service" type="xsd:string" use="required" fixed="WFS"/>
        <xsd:attributename="outputFormat" type="xsd:string"use="optional"
        default="XMLSCHEMA"/>
</xsd:complexType>
```

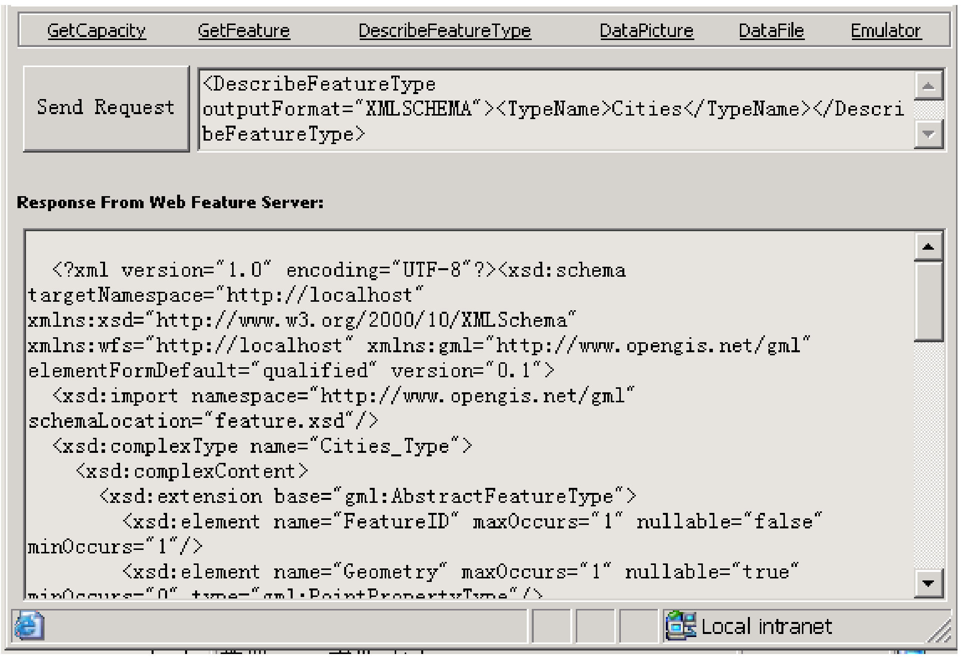
The outputFormat attribute, is used to indicate the schema description language that should be used to describe feature type schemas. The only mandatory output format in response to a DescribeFeatureType operation is XML Schema, denoted by the value XMLSCHEMA for the outputFormat attribute.

6.5.3 Response

In response to a DescribeFeatureType request, where the value of the outputFormat attribute has been set to XMLSCHEMA, a WFS service must be able to present an XML Schema document that is a valid GML application schema and defines the schema of the feature types listed in the request. The document(s) presented by the DescribeFeatureType request may be used to validate feature instances generated by the WFS in the form of feature collections on output or feature instances specified as input for transaction operations.

6.5.4 Example

Figure 6.5 is the screenshot of this operation.



```
<DescribeFeatureType
outputFormat="XMLSCHEMA"><TypeName>Cities</TypeName></Descri
beFeatureType>
```

Response From Web Feature Server:

```
  <?xml version="1.0" encoding="UTF-8"?><xsd:schema
targetNamespace="http://localhost"
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
xmlns:wfs="http://localhost" xmlns:gml="http://www.opengis.net/gml"
elementFormDefault="qualified" version="0.1">
  <xsd:import namespace="http://www.opengis.net/gml"
schemaLocation="feature.xsd"/>
  <xsd:complexType name="Cities_Type">
    <xsd:complexContent>
      <xsd:extension base="gml:AbstractFeatureType">
        <xsd:element name="FeatureID" maxOccurs="1" nullable="false"
minOccurs="1"/>
        <xsd:element name="Geometry" maxOccurs="1" nullable="true"
minOccurs="0" type="gml:PointPropertyType"/>
```

Figure 6.5

In response to this deliberately simplified DescribeFeatureType request:

```
<?xmlversion="1.0"?>
<DescribeFeatureType
xmlns=http://www.opengis.net/wfs
xmlns:myns=http://www.myserver.com/myns
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
outputFormat="XMLSCHEMA">
        <TypeName>Cities</TypeName>
</DescribeFeatureType>
```

My demo WFS service might generate an XML Schema document that looks like:

```
<?xml version="1.0" encoding="UTF-8"?><xsd:schema
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema" xmlns:wfs="http://localhost"
xmlns:gml="http://www.opengis.net/gml">
  <xsd:import namespace="http://www.opengis.net/gml" schemaLocation="feature.xsd"/>
  <xsd:complexType name="Cities_Type">
    <xsd:complexContent>
      <xsd:extension base="gml:AbstractFeatureType">
        <xsd:element name="FeatureID" maxOccurs="1" nullable="false" minOccurs="1"/>
        <xsd:element    name="Geometry"    maxOccurs="1"    nullable="true"    minOccurs="0"
type="gml:PointPropertyType"/>
        <xsd:element name="XMin" maxOccurs="1" nullable="true" minOccurs="0"/>
        <xsd:element name="XMax" maxOccurs="1" nullable="true" minOccurs="0"/>
        <xsd:element name="YMin" maxOccurs="1" nullable="true" minOccurs="0"/>
        <xsd:element name="YMax" maxOccurs="1" nullable="true" minOccurs="0"/>
        <xsd:element name="City_name" maxOccurs="1" nullable="true" minOccurs="0"/>
        <xsd:element name="Admin_name" maxOccurs="1" nullable="true" minOccurs="0"/>
        <xsd:element name="Cntry_name" maxOccurs="1" nullable="true" minOccurs="0"/>
        <xsd:element name="Status" maxOccurs="1" nullable="true" minOccurs="0"/>
        <xsd:element name="Pop_rank" maxOccurs="1" nullable="true" minOccurs="0"/>
        <xsd:element name="Pop_class" maxOccurs="1" nullable="true" minOccurs="0"/>
        <xsd:element name="Port_id" maxOccurs="1" nullable="true" minOccurs="0"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```

6.6 GetFeature operation

6.6.1 Introduction

The GetFeature operation allows retrieval of features from a web feature service. A GetFeature request is processed by a WFS and an XML document, containing the result set, is returned to the client.

6.6.2 Request

The XML encoding of a GetFeature request is defined by the following XML Schema fragment:

```
<xsd:elementname="GetFeature"type="wfs:GetFeatureType"/>
<xsd:complexTypename="GetFeatureType">
<xsd:sequence>
<xsd:elementref="wfs:Query"maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attributename="version" type="xsd:string"use="required"fixed="1.0.0"/>
<xsd:attributename="service" type="xsd:string"use="required"fixed="WFS"/>
<xsd:attributename="handle" type="xsd:string"use="optional"/>
<xsd:attributename="outputFormat" type="xsd:string"use="optional"default="GML2"/>
</xsd:attribute>
<xsd:attributename="maxFeatures"type="xsd:positiveInteger" use="optional"/>
</xsd:complexType>
<xsd:elementname="Query"type="wfs:QueryType"/>
<xsd:complexTypename="QueryType">
<xsd:sequence>
<xsd:elementref="ogc:PropertyName"minOccurs="0"maxOccurs="unbounded"/>
<xsd:elementref="ogc:Filter"minOccurs="0"maxOccurs="1"/>
</xsd:sequence>
<xsd:attributename="handle" type="xsd:string"use="optional"/>
<xsd:attributename="typeName" type="xsd:QName"use="required"/>
<xsd:attributename="featureVersion" type="xsd:string"use="optional"/>
```

```
</xsd:complexType>
```

The <GetFeature> element contains one or more <Query> elements, each of which in turn contain the description of a query. The results of all queries contained in a GetFeature request are concatenated to produce the result set.

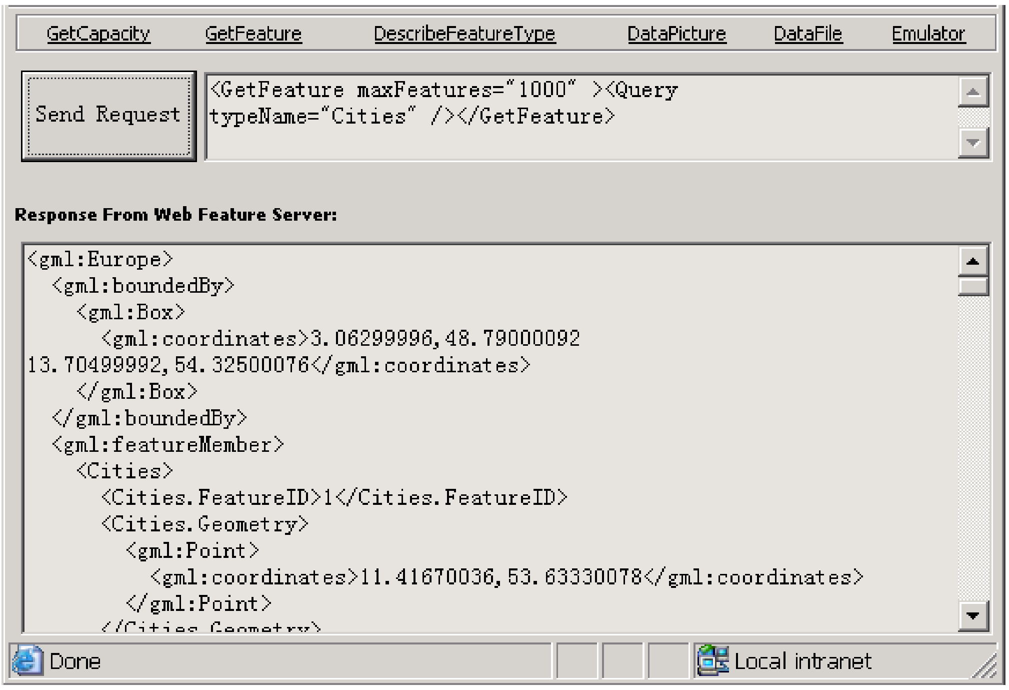
The outputFormat attribute defines the format to use to generate the result set. The default value is GML2 indicating that GML [2] shall be used. Vendor specific formats (including non-XML and binary formats), declared in the capabilities document are also possible.

This is a rather sophisticated operation, further elaboration of it could be found at OGC official website.

6.6.3 Response

The format of the response to a GetFeature request is controlled by the outputFormat attribute. The default value for the outputFormat attribute shall be GML2. This will indicate that a WFS must generate a GML document of the result set that conforms to the OpenGIS Geography Markup Language Implementation Specification, version 2.1.1 [2], and more specifically, the output must validate against the GML application schema generated by the DescribeFeatureType operation.

Figure 6.6 is the screenshot of this operation.



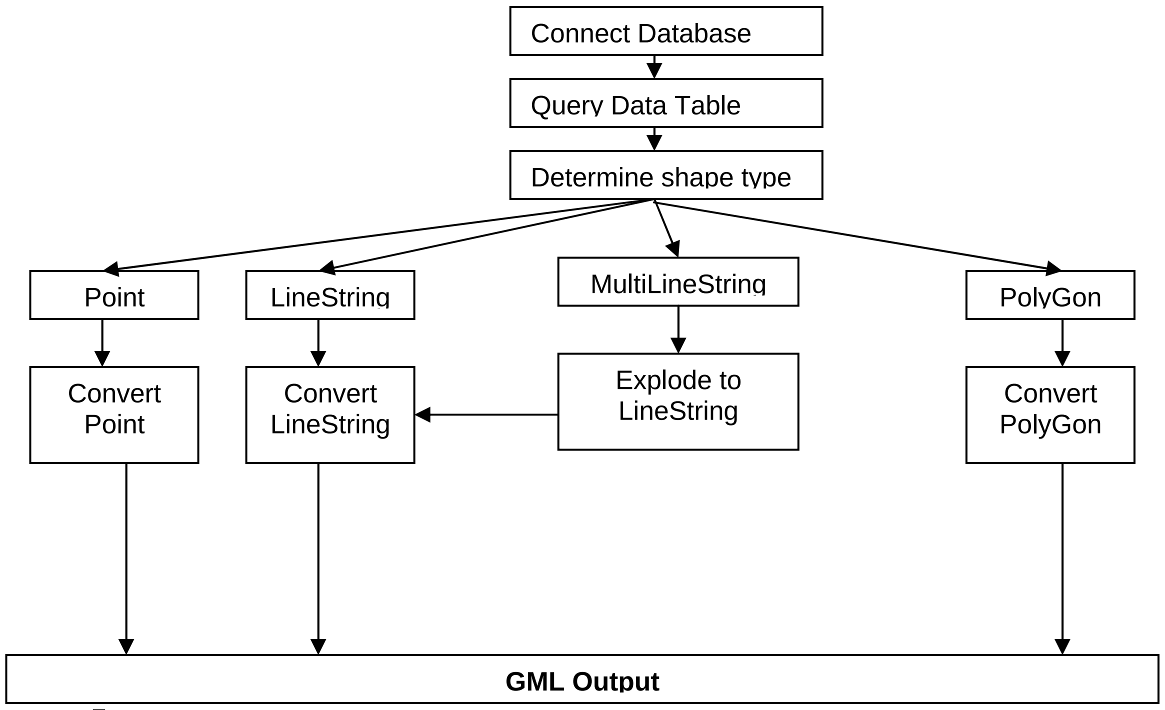
```
<GetFeature maxFeatures="1000" ><Query
typeName="Cities" /></GetFeature>
```

Response From Web Feature Server:

```
<gml:Europe>
  <gml:boundedBy>
    <gml:Box>
      <gml:coordinates>3.06299996,48.79000092
13.70499992,54.32500076</gml:coordinates>
    </gml:Box>
  </gml:boundedBy>
  <gml:featureMember>
    <Cities>
      <Cities.FeatureID>1</Cities.FeatureID>
      <Cities.Geometry>
        <gml:Point>
          <gml:coordinates>11.41670036,53.63330078</gml:coordinates>
        </gml:Point>
```

Figure 6.6

6.6.3 Data flow



7

Figure 6.7

Figure 6.7 depicts the programming model.

Chapter 7 Discussions

7.1 Discussions

OGC declares their mission is to deliver spatial interface specifications that are openly available for global use. Adhering to their mission, a suite of implementation specifications have been developing till now.

WFS certainly is only a rather small part of OGC specifications. From my prototype system, we can see the functionality is quite limited. But some light have been thrown on the brilliant vista that make geographic information and services available across any network, application or platform. It won't be hard to imagine how powerful and versatile it would be if we intermingled and integrated all web service defined in OGC specifications.

XML is designed to be self-explanatory and self-descriptive, as well as separate information content from presentation. Those genetic advantages allow various services mutually understandable and comprehensible. Based on this, we can foresee a significant automation in the process of data retrieval in a not far future, as OGC specifications is rapidly becoming a benchmark in this industry.

Traditionally, LBS user subscribe or buy the geo service from a certain service provider, or their LBS devices or applications only recognize or accept a certain data format or service interface. But OGC specifications turn this situation around. With the unifying service interface, user's client LBS devices or application will be capable to perform discovery, browse and query from multi source or heterogeneous datasets, without knowing who is the data provider behind it. This is going to tremendously expand the scope and capacity of LBS, by removing the boundary and hurdle among all those providers. I envision that this geo data global access will be taking off soon.

I think there are couple of key issues that are worth discussing, like the data repository of my prototype system and some pitfalls

7.2 Data repository

This niche market is flourishing with many pre-eminent database products. If you are seeking uncompromising stability and brilliant performance, the Oracle Spatial is indisputably your first option. However, the painful installation and esoteric operations manual could be as frustrating as you could expect. Sometime while we are testing our serendipity on the Internet, there might be some felicitous findings. That exactly how I got the Mysql Spatial Extension, which is a free and open source spatial database.

In 1997, the Open GIS Consortium published the OGC Simple Features Specifications For SQL, a document that proposes several conceptual ways for extending an SQL RDBMS to support spatial data. This specification is available from the Open GIS web site at http://www.opengis.org/techno/implementation.htm.

MySQL implements a subset of the SQL with Geometry Types environment proposed by OGC. This term refers to an SQL environment that has been extended with a set of geometry types. A geometry-valued SQL column is implemented as a column that has a geometry type. The specifications describe a set of SQL geometry types, as well as functions on those types to create and analyze geometry values.

Mysql supports standard spatial data formats that are used to represent geometry objects in queries. They are Well-Known Text (WKT) format and Well-Known Binary (WKB) format. In my system, I only deal with WKT, thus the introduction of WKB will be left out.

The Well-Known Text (WKT) representation of Geometry is designed to exchange geometry data in ASCII form.

Examples of WKT representations of geometry objects are:

A Point:
POINT(15 20)
Note that point coordinates are specified with no separating comma.
A LineString with four points:
LINESTRING(0 0, 10 10, 20 25, 50 60)
A Polygon with one exterior ring and one interior ring:
POLYGON((0 0,10 0,10 10,0 10,0 0),(5 5,7 5,7 7,5 7, 5 5))
A MultiPoint with three Point values:
MULTIPOINT(0 0, 20 20, 60 60)
A MultiLineString with two LineString values:
MULTILINESTRING((10 10, 20 20), (15 15, 30 15))
A MultiPolygon with two Polygon values:
MULTIPOLYGON(((0 0,10 0,10 10,0 10,0 0)),((5 5,7 5,7 7,5 7, 5 5)))
A GeometryCollection consisting of two Point values and one LineString:
GEOMETRYCOLLECTION(POINT(10 10), POINT(30 30), LINESTRING(15 15, 20 20))

In this system, I exported a part of Europe map in format of Shapefile, from Arcview to WKT file. And then populated it to Mysql data table. I deliberately chose 3 typical geometry object as 3 layers stored in different type data table: point, multilinestring, multipolygon.

Figure 7.1 is the visualization in ArcView.

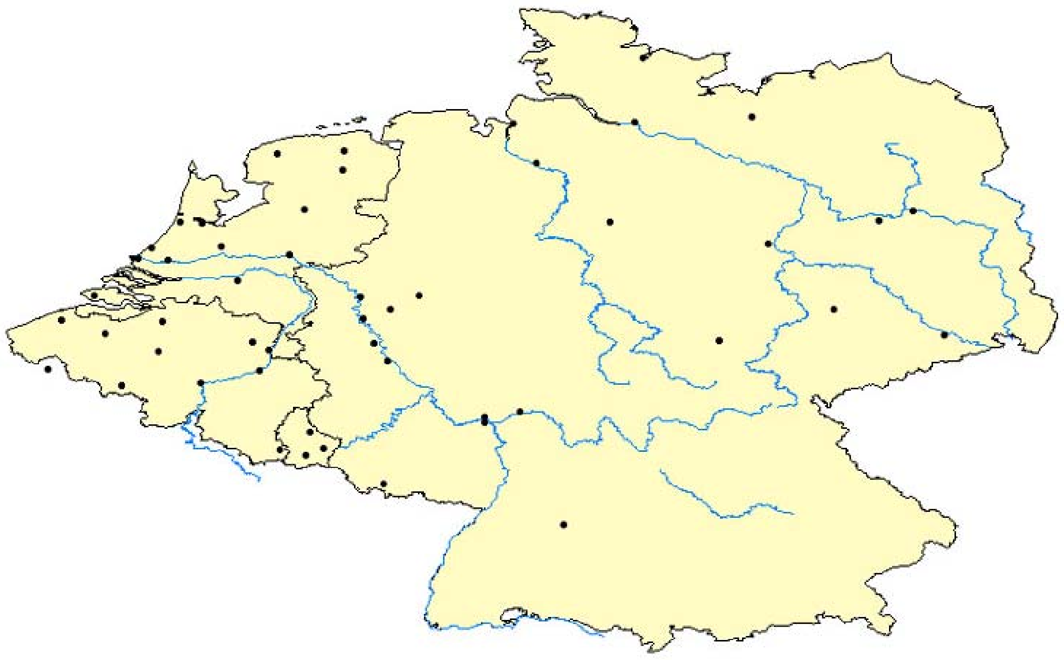


Figure 7.1

7.3 Limitations and pitfall

This demo system was merely built for usage of demonstrating my thesis work, and it is definitely not a perfect or flawless system. In order to put it in practice, there are certainly many limitations to overcome.

7.3.1 Data acquisition

In this case, I chose Mysql as my data repository. Mysql is well-known and widely recognized database software, which is further enhanced by its spatial extension. Especially, it's free and open source. Everything is so good except I didn't fine a good way to migrate geo-data from other resources to Mysql datatable.

The data I am making use of in this system is exported from Arcview. I installed an Arcview extension, which allows me export the shape file to Mysql datatable. However, the drawback of this process is if there were a big shapefile that we are going to export, the system would hang up or just crush. I am looking for a method to transfer or convert data with higher performance and stability.

7.3.2 Transaction

At current version of WFS specification, a notable operation was added: transaction. The Transaction operation is optional and a WFS implementation does not need to support it to conform to this specification. If the Transaction operation is supported then this fact must be advertised on the capabilities document as I aforementioned.

The Transaction operation is used to describe data transformation operations that are to be applied to web accessible feature instances. A web feature service may process a Transaction operation directly or possibly translate it into the language of a target datastore to which it is connected and then have the datastore execute the transaction. When the transaction has been completed, a web feature service will generate an XML response document indicating the completion status of the transaction.

I didn't implement the Transaction operation in my demo system, due to time constrain and some technique difficulties. As we can see, the functionality of the service would be dramatically expanded if this operation were implanted.

7.3.3 Security

Security is always a big issue under any circumstances. Throughout of this demo system, there is no any user authentication. Even I deliberately overlooked verification mechanism for data's validity and reliability. Obviously there are still so many works to do ahead.

In practice, the system is expected to be secure in two ways:

- it must not allow users to see any restricted data, and
- it must guard against requests for too much data that, if processed blindly, would result in loss of or degradation in service.

Since the Internet can be a very hostile environment, there must be a layer of software between the underlying database and users, ensuring that users cannot find ways to sensitive data. If the system detects any attempts to thwart the security, then it should log this information with as much user and/or IP information as possible, and notify system operators.

The system must also be capable of handling requests for too much data. For example if there is a theme named "Roads" that contains all the roads in the continental United States, the system should guard against a misinformed or hostile user that requests all the roads for a particular state or for the whole country. This is too much data for a real-time request and processing such a request would greatly degrade the system performance.

Ideally, systems administrators should be able to define the size of data that is to be distributed and provides for different levels of service based on the amount of data that is requested. An example of one possible set of different levels is described below:

Real-time Service: This is for small requests. This value is dependent on a number of factors: server bandwidth, client bandwidth, number of expected simultaneous clients, and throughput of server. For these requests, the system processes the request immediately with a turnaround time that would be acceptable for a user waiting at a browser.

E-mail Service: These requests are the next level in size. The server still processes the requests immediately, but it is recognized that the delay is beyond the threshold of a user waiting at a browser. The user is sent an e-mail message with an ftp link that points to the extracted data.

Physical Media Service: This level of service is for data requests that are performed off-line and then put on physical media. These results are deemed to be simply too big to be sent via the communication infrastructure.

Prohibited Service: This is for requests that are deemed too large to process. The request is logged and the client is simply notified that that the data request is too big for the data distribution system

7.3.4 Exception handler

To any robust system architectural design, the exception handler is absolutely indispensable. The user may send any kind of wrong request, which could be wrong format dataset, incompatible parameters and malicious attack code. We could never expect user is well trained and knowledgeable to be aware what they are doing. All possible causes of system exception should be considered at system design stage and handled by the exception handlers. Unfortunately, I didn’t take account of this.

Chapter 8 Conclusion

8.1 Conclusion

This paper started with the pertinent background and what is impeding the spatial data's free transaction. The OpenGIS Consortium was introduced and its specification was brought out. And lot of elaboration was given on XML, GML and Web Service. At last, a prototype system was developed to implement and demonstrate the WFS specification.

Leveraging OGS specification, a boundless and seamless spatial data environment can be built throughout most of geo data operations. Standard and unifying XML-Based web service significantly reinforce the interoperability among data providers, which bestows considerable automation on data retrieval from multiple source and heterogeneous datasets.

8.2 Prospect

The geo data global access is no longer the fantasy, and it will be looming as OGC specifications are conforming this arena. If trickle of standard web service turns into flood, the intriguing and captivating prospect of real data sharing freedom will be just ahead of us, not that far.

References

1. Elliotte Rusty Harold, XML Bible, IDG Books worldwide, Inc

2. Geography Markup Language (GML) 2.0, OpenGIS Implementation Specification, 20 February 2001 OGC Document Number:01-029

3. Scalable Vector Graphics (SVG) 1.0 Specification, W3C recommendation 04 September 2001

4. Environmental Systems Research Institue, Inc, ESEI Shapefile Technical Description, And ESRI White Paper, July 1998

5. Michela Bertolotto, A conceptual Model for Web-Based Heterogeneous GIS

6. Interoperability Piece Together Location-based Services, by Alistair Edwardes

7. Johnson, Bill. October 2, 1997. The NYS GIS Data Sharing Cooperative. Available at: http://www.nysgis.state.ny.us/coop_brf.htm

8. Nedovic-Budic, Zorica, and Tschangho John Kim. 1999. Introduction - Special issue on geo-spatial information sharing and standardization. Annals of Regional Science. 33. 141-143.

9. Reichardt M.: OGC's GML 2.0 – A New Wave of Open Geoprocessing on the Web, eoInformtics, Magazine for Geo-IT Professionals, Issue July/August 2001, p. 18-21.

10. http://www.uml.com.cn/ (Chinese)

11. http://www.opengis.org

12. http://www.mysql.com

13. http://www.digitalearth.net.cn/ (Chinese)

14. http://www.php.com

15. http://www.nsdi.gov.cn/ (Chinese)

16. http://www.xml.com/

17. http://www.open.gov.uk/

18. http://celiang.tongji.edu.cn/ (Chinese)

19. http://www.w3.org