UCGE Reports
Number 20148

Department of Geomatics Engineering

Development of a Distributed Geoprocessing Service Model

(URL: http://www.geomatics.ucalgary.ca/GradTheses.html)

by

Shuxin Yuan

December 2000

UNIVERSITY OF CALGARY

UCGE Reports
Number 20148

Department of Geomatics Engineering

Development of a Distributed Geoprocessing Service Model

(URL: http://www.geomatics.ucalgary.ca/GradTheses.html)

by

Shuxin Yuan

December 2000

UNIVERSITY OF
CALGARY

THE UNIVERSITY OF CALGARY

DEVELOPMENT OF A DISTRIBUTED GEOPROCESSING SERVICE MODEL

by

SHUXIN YUAN

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF GEOMATICS ENGINEERING

CALGARY, ALBERTA

DECEMBER, 2000

ABSTRACT

GIS services can be classified into geodata services and geoprocessing services. Most existing GIS service systems focus more on providing geodata services than geoprocessing services, and offer very limited geoprocessing functions. It has been realized that, in GIS services, distributed geoprocessing is very critical to help users in geodata manipulation, such as format and reference system conversions, editing, and analysis. In this research, a new GIS service model is developed to support distributed geoprocessing. It is expected that this model will contribute to the development of making GIS functions accessible to network users on a per request basis.

The new model is designed to allow geoprocessing components to be distributed anywhere in the Internet and be accessible at client sites whenever they are requested. Some new concepts such as component registration and component meta-information are introduced. A prototype implementation based on Java technologies has been developed to demonstrate and test the model.

ACKNOWLEDGEMENTS

I wish to express my deep gratitude to my supervisor, Dr. C. Vincent Tao, for his advice, guidance, encouragement, and support throughout my graduate studies. I am also grateful to Dr. J. A. R. Blais for his early critiques during my thesis research work. My appreciation also goes to my fellow graduate students, Mr. Chuangyun Fei for his cooperation, discussions and suggestions in the GeoEye™ related software improvements, and Mr. Ronald Ramsaran for his proofreading of the draft of this thesis. I am also grateful to the secretaries, technicians and computer specialists and graduate students in the Remote Sensing and GIS Lab of the Department of Geomatics Engineering, who, in one way or other, helped me during my graduate studies. Finally my appreciation and thanks go to my family, my dearest wife, Guiping Wang, and my lovely son, Hao, for their endless love, devotion and support, which make this thesis possible.

TABLE OF CONTENTS

LIST OF FIGURES

No. **Page**

LIST OF TABLES

TERMS AND ACRONYMS

The following lists some terms and acronyms frequently used in this thesis.

ActiveX A component technology developed by Microsoft. It is developed upon the Object Linking and Embedding (OLE) technology and used in Microsoft Component Object Model (COM) and Distributed Component Object Model (DCOM) architecture.

API Application Program Interface.

Applet A type of java program running in the client web browser.

Bean A type of java component that is inter-operable with other java components.

Client/Server Refers to a traditional network computing model, in which a server has many clients. Resources are centralized in the server, and the server does most computing. The client provides user interfaces to the user.

CGM Computer Graphics Metafile. A standard used to represent vector graphics.

COM/DCOM Component Object Model (COM) and Distributed Component Object Model (DCOM) architecture developed by Microsoft.

CORBA The Common Object Request Broker Architecture developed by Object Management Group

DGIS	Distributed Geographic Information System
DLL	Dynamic Link Library
Geodata	Geographically related data
Geoprocessing	Processing of geographically related data
GIS	Geographic Information System / Geographic Information Services
GUI	Graphic User Interface
HTML	HyperText Makeup Language. A standard used for transferring text and images in the Internet
IT	Information Technology
IIOP	Internet Inter-ORB Protocol used in CORBA. ORB stands for Object Request Brokers.
JDBC/ODBC	Java Database Connectivity / Open Database Connectivity. A technology used in Java to access database information.
LAN	Local Area Network
PC	Personal Computer
RMI	Remote Method Invocation. Architecture used in Java technology for distributed computing.
Servlet	A Java program running in a web server machine, act as middleware between application server and web server.
SVG	Scaleable Vector Graphics. A standard developed to embed in XML for vector graphics.

Thick Client Refers to a client/server computing model, in which the client perform most computing tasks.

Thin Client Refers to a client/server computing model in which the server perform most computing tasks.

WAN Wide Area Network.

Wrapper An interface that is used to connect a component to other components or programs.

WWW World Wide Web. Usually refers to the Internet.

XML Extensible Markup Language. A new standard used for Internet communication. More advanced than HTML.

CHAPTER 1 INTRODUCTION

1.1 Research Background

Geographic information services, or GIS (Geographic Information System) services, are referred to the providing of geographic information to users. In a network environment, especially the Internet, such services can be provided much easier and faster with the adoption of web-enabled GIS. As described in The OpenGIS Guide (Buehler and McKee, 1998), GIS services contain two basic types:

1) access and process the geographic types defined in the Geodata Model, and

2) provide capabilities to share geodata within communities of users who use a common set of geographic feature definitions and translate between different communities of users that use different sets of geographic feature definitions.

The first type of services is referred to as the geoprocessing services, which focuses on providing geoprocessing functionality to users. The second type of services is referred to as the geodata services, which addresses the issues for distributing georeferenced data to consumers.

In the past several years, following the rapid development of Information Technologies (IT), especially with the introducing of the Internet/Intranet, distributed object computing and object database technologies, GIS software development has evolved many phases.

The biggest evolution, in the author's opinion, is that it shifted its paradigm from desktop-centric GIS to network-centric GIS so as to provide the increasing network users with geo-referenced information and geoprocessing tools. Due to the wide accessibility of the Internet, GIS software vendors, information providers and users became more and more interested in Internet GIS services. Great efforts had been given to the geodata services. Now on-line Internet GIS services are no longer a novel idea, but a reality in terms of publishing centralized geodata over the Internet. Many on-line geodata catalog and query services are now available over the Internet. These services are based on different geodata publishing systems produced by different vendors with different technologies. These systems usually use the client-server model, in which clients submit requests and the server processes the requests and returns the results. GIS data can be transmitted and displayed on web browsers.

On the other hand, geoprocessing services have not gained enough attentions. Fewer efforts have been made in this field. In the existing GIS service systems, GIS functionality provided to the clients is limited to a very narrow range, typically data query and data display. However, on-line geoprocessing services are also of very significant value to the users. It plays an important role in many applications. A typical application scenario is that the client has his own geodata sets and only wants to “rent” some GIS processing tools to process them, such as data format conversion, reference system transformation, data editing, data analysis and modeling. Expanding this use case to a more complicated scenario, the client may combine geodata from several different places and process them by using geoprocessing tools available in the network and save the results in the local site. A typical geodata publishing system cannot solve this problem. To tackle this problem, different technologies should be applied such as distributed object technologies. A distributed GIS model is required. The difference between a distributed GIS and a traditional client-server GIS is shown in Figure 1.

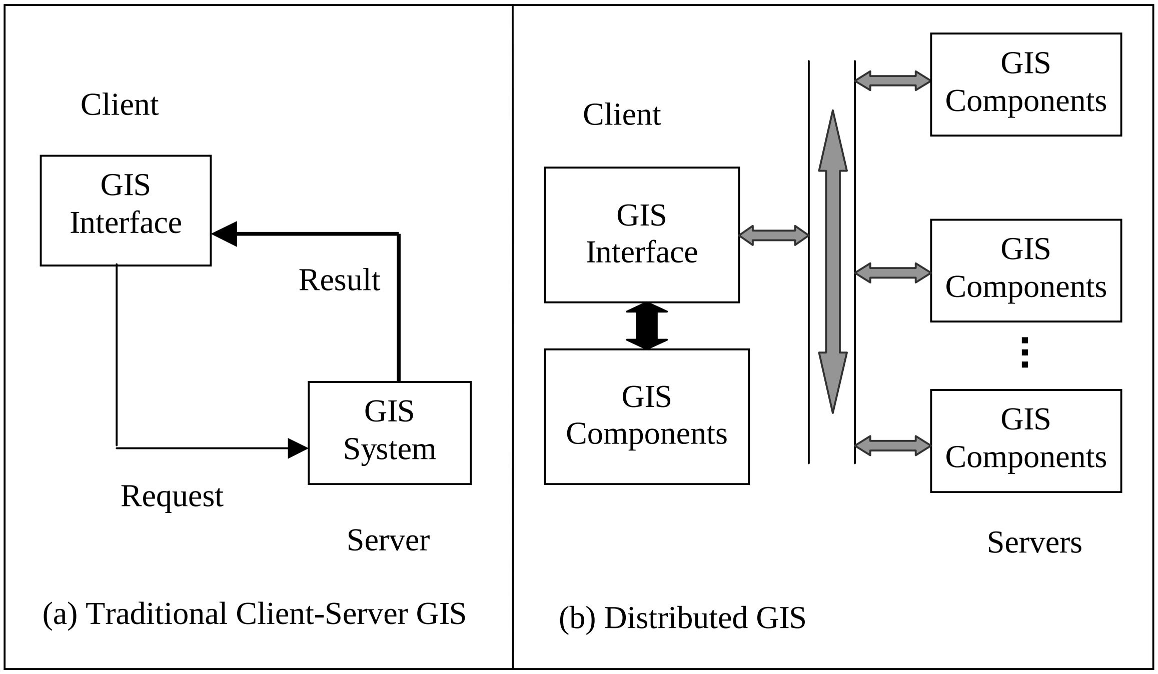


Figure 1: Different Structures of Traditional Client-Server GIS and Distributed GIS

Distributed GIS has become the trend of GIS software development. Unlike the traditional client-server architecture used in most geodata service systems, distributed GIS allows GIS components to be distributed in different locations of the entire network, not only data components, but also GIS functional objects. GIS software is no longer a large single integrated system. It can be decomposed into many interoperable functional components using componentware technology. In a typical client/server system, both geodata sets and geoprocessing tools are located at the server site. The client only sends requests to the server. The server receives the request, conducts the corresponding computation and sends the result to the client. In this architecture, every GIS computation, including the basic ZOOM and PAN functions, is performed on the server. In this case, the server is referred to as thick-server and the client is referred to as thin-client. Since the result have to be transmitted back to the client every time when the server finishes the computation, the network traffic is very heavy, especially when the result is transmitted in the form of raster images. The client-server model may be efficient in terms of providing centralized geodata services, but it does not have the ability to process the data on other sites. A distributed GIS model may change this situation. In a distributed model, a client may use GIS resources distributed anywhere in the network, not just from the server in the client-server model. The role of a client is not only sending the requests to the server, but also performing some basic GIS operations, such as display operations (PAN, ZOOM), etc.

Distributed geoprocessing service model study is an important part of distributed GIS research. To provide GIS functionality over a heterogeneous network environment, distributed object technologies will be involved in the development. The object distribution architecture and the GIS data model are the key parts of the model study that affect the structure and the performance of the service model. A distributed geoprocessing model should be flexible for taking advantages of both geodata and GIS functional resources distributed on the network. It may also balance the computation burden between the client and servers. Neither a thin-client structure (in which a server do everything) nor a thick-client structure (in which the client do everything) meets this requirement.

Among the several alternative solutions of the distributed object model, the Java technology is the fastest and easiest way for distributed system implementation. Due to the Windows platform limitation of Microsoft’s Distributed Component Object Model (DCOM), and the implementation complication of Object Management Group’s (OMG) the Common Object Request Broker Architecture (CORBA) (Alberson 1998), Java technology are more widely used in the Internet GIS applications. The Java family provides a series of ease-of-use solutions for platform independent distributed systems. Java Applet provides the clients with user interfaces and shared classes through the Java enabled web browsers. Java Servlet provides an efficient way of communication between application server and web server. Java Bean provides an interoperable way to produce interoperable components. Java Database Connectivity (JDBC) or JDBC-ODBC (Open Database Connectivity) provides a universal means of connection between the internal objects and the external database. Java RMI enables the remote object method invocation, which is the basis of Java distributed computing.

In this research, a distributed geoprocessing service model is proposed. This new model is designed and implemented based upon pure Java technologies, which allows the geoprocessing components to be distributed anywhere and transferred to the client whenever the geoprocessing service is request. Geodata at either client or server site can be processed via a Java enabled Internet browser and the processed result can be saved in the client machine. The model proposed is intended to improve the performance of client-server communication and to make the entire geoprocessing functions accessible over the Internet.

1.2 Objectives and Limitations

The major objective of this research is to develop an efficient distributed object model for providing geoprocessing services to GIS clients over the Internet. The focus of the research is on the system architecture design, implementation and testing, methodology study for geoprocessing component distribution and registration, and client interfaces implementation for remote object invocation. The following lists the objectives of the thesis research:

- Study the distributed object technology. Propose a novel distributed geoprocessing model that can distribute geoprocessing components on different sites of the Internet and use both geodata and geoprocessing resources available in the entire Internet.
- Use Java technologies in design and implementation of the model.
- Design and implement a simplified geodata model and shared classes in pure Java. Since there are no existing GIS componentware or classes available in Java, every class and method have to be implemented from the ground up.

- Use the OpenGIS specifications in the model design and implementation.
- Implement a prototype geoprocessing service system to test the model design and demonstrate the performance of the model.
- Develop geoprocessing service application examples to demonstrate and test the model.

Distributed GIS is a large topic that has many issues to be considered. However, due to the time limitation of the research, many of these issues are not included into this research scope. The followings are some of the limitations considered towards the research.

- This research does not intend to develop a complete GIS system although a simplified geodata model is developed in the prototype system. A complete GIS is a complicated software system that comprises many components and functions.
- This research focuses on the geoprocessing services instead of geodata services. Many issues in geodata services, such as metadata, geodata catalog and publishing, etc. are not discussed in the thesis.
- This research focuses on the Internet on-line geoprocessing service model design and prototype implementations. Specific issues and functions, such as distributed data transaction, security issues, etc. are not included in the current research.

1.3 Outline

The thesis consists of five chapters, including this introductory chapter.

Chapter Two is a review of current GIS development status and trends, in both industrial and academic aspect, and focuses on the technologies used in web-enabled GIS. Distributed object technologies, its use in GIS, as well as Open GIS Specifications are introduced and discussed. Current Internet on-line GIS service applications, architectures, technologies and problems are analyzed in this chapter.

Chapter Three introduces the design of the geoprocessing service model. This model design includes a conceptual model design, geodata model design, and system architecture design. Geodata model covers GIS geometry, geodata structure and reference system. The system architecture contains client-server communication, geoprocessing component distribution and registration.

Chapter Four introduces the implementation of the model designed shown in Chapter three. In this chapter, Java implementation strategy of client and server side programs, component registration system, user interfaces, and the results achieved are introduced. Some component examples that test and demonstrate how the model works are also introduced. The potential uses of the model are discussed at the end of this chapter.

Chapter Five concludes the research. Some conclusions are drawn according to the implementation and test of the prototype system. Also recommendations for the continuing research are given.

CHAPTER 2 DISTRIBUTED GIS SERVICES

2.1 GIS Software Development

A GIS is an organized collection of computer hardware, software, geographic data, and personnel designed to efficiently capture, store, update, manipulate, analyze, and display all forms of geographically referenced information (ESRI 1990). From a software engineering point of view, GIS is a computer system designed to allow users to collect, manage, and analyze large volumes of geographically referenced spatial data and associated attribute data. The major components of a GIS are the geographically referenced data and the geoprocessing functionality. The major software components of a GIS include user interfaces (UI), spatial data manipulation, spatial analysis, database creation, data-entry and database management capability, and data visualization functions.

Initiated in 1960's (Coppock and Rhind, 1995), GIS software experienced several development phases and gradually evolved into the mainstream of Information Technology. With the rapid development of computer network technology and the fact that more and more individuals and organizations adopted GIS as their solutions, the demands for an open, network-centric distributed GIS became the common concern of GIS software vendors, geographic information providers and GIS users. As more and more people tend to use geographically referenced information and GIS software, GIS services that provide geographical information to broader users became a hot issue in the recent years, especially GIS services over the Internet. Driven by both technology development and GIS market, GIS software has changed its paradigm several times: From the initial GIS functional packages to the integrated huge system and from modular GIS to component GIS, as shown in Figure 2. From desktop GIS to network-centric GIS, and now from traditional client/server GIS to Distributed GIS (DGIS), as shown in Figure 3. Each of these changes marked a big progress in the history of GIS development.

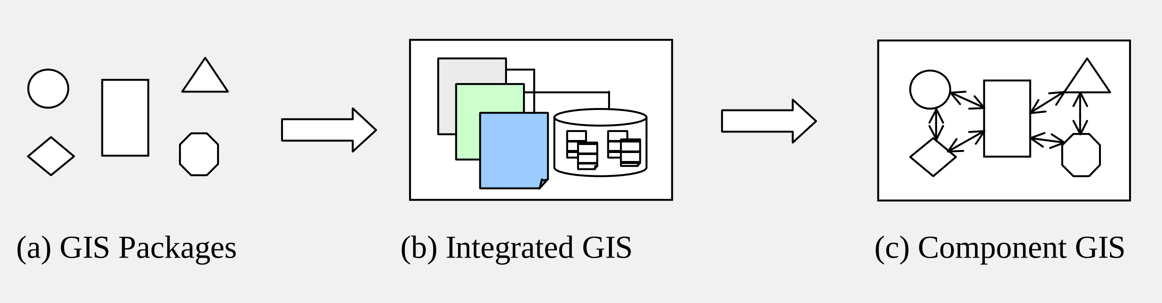


Figure 2: GIS Movement Towards Component GIS

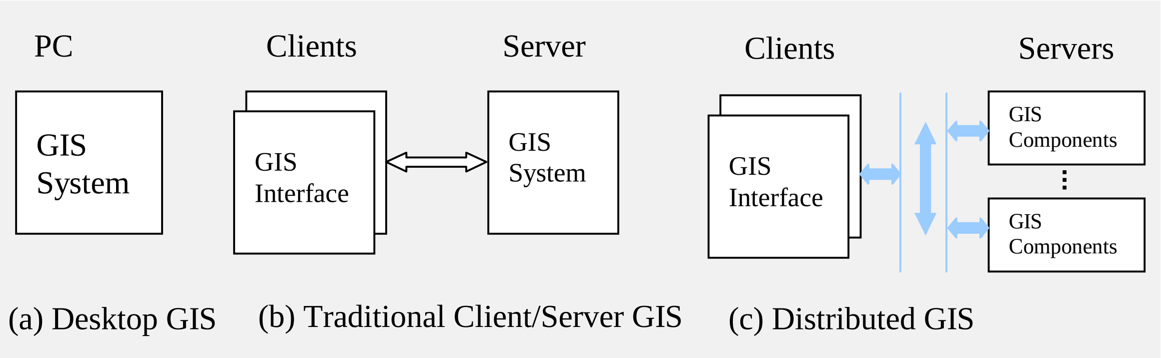


Figure 3: Desktop, Traditional Client/Server GIS and Distributed GIS

2.1.1 Historic Stages in GIS Development

In the history of GIS software development, GIS software experienced the following development stages: GIS functional packages, Integrated GIS, Modular GIS, Core GIS, and Component GIS. The general trend of GIS is towards the Open GIS in a heterogeneous network environment.

In the early development stage of GIS software, constrained by technology limitations, GIS software actually is only some geoprocessing functional packages. These individual packages could not efficiently cooperate and an integrated system had not been formed. Later, with the maturity of the computer technologies, GIS packages naturally integrated into large systems. The advantage of an integrated GIS was that it integrated various GIS functions into an independent system, making it easier to operate and conduct more complicated GIS tasks. The disadvantage of it is that the system tends to be too large, so it is costly, hard to maintain and difficult to integrate with other systems. The integrated GIS is an important benchmark in GIS history, since it made GIS a real system instead of some software packages. A good representative of the integrated GIS would be ESRI's Arc/Info (ESRI 1995).

Another development of GIS software that emerged later is the Modular GIS, represented by Intergraph's MGE (Intergraph 1995). The basic idea of the Modular GIS is to divide GIS into a series of functional modules executed on a fundamental environment (such as

MicroStation). Different from an Integrated GIS, which also may be considered as formed by several modules, Modular GIS planed its modules in a more detailed style. This is very useful for the development of a complicated GIS and makes the software maintenance easier. Modular GIS overcame some disadvantages from integrated GIS, however, it is still hard to integrate with other applications.

To overcome these disadvantages of GIS, the Core GIS concept was proposed (Zhang 1995). The core GIS was designed as an extension to the operation system. In Windows platform, it is a collection of Dynamic Linking Library (DLL). In a specific GIS application, developers access the GIS function library through Application Program Interface (API). The GIS functions in DLL can be reuse in different applications, which provides great flexibility for the system integration. However, the core GIS requires developers to be knowledgeable of DLL functionality. The application development is somehow difficult and does not fit the visual programming requirements.

Component technology is an important innovation in software development, which changed the trace of GIS development. Component GIS has become a strong trend in GIS software development. In component GIS, GIS is again partitioned into many lightweight standalone software pieces. However, these software pieces are not the traditional software packages or modules, but vivid, inter-operable software components that can be freely "plug and play" in GIS applications. Based on a certain component technology and standard such as ActiveX in COM/DCOM architecture and Java Bean, a component can communicate with other components through its visual interface to the outside. This feature makes the regrouping of GIS components flexible enough in system integration. Currently, most GIS software venders realized the importance of the component GIS and its implications. Developing component GIS software became an important strategy of GIS products. Some examples of component GIS products are Intergraph's GeoMedia, Esri's MapObjects, etc., which are adopted in many GIS applications, ranged from simple desktop-centric GIS applications to network-centric GIS application over the Internet (Engen 1997 and Intergraph 1999).

As clear as the trace of GIS software development from the initial GIS functional packages to Integrated GIS and to GIS components, GIS software made another deep trace in its general architecture developments. That is from host-centric systems to desktop-centric applications and further to the network-centric GIS.

In its initial days when the integrated GIS came out, GIS turned out to be very complicated system that had very high performance requirements to the computer that loaded it. Only can high performance mainframe computers afford these requirements due to the status of computer development at that time. Host-centric architecture was characterized by centrally located host or mainframe computers that performed all the computations, data storage and management, and screen interfaces generation and interaction. End user interacted with and shared the resources of the host via terminals, which had no capability to save information locally, perform local processing, or directly control the user interfaces.

Subsequently, with the rapid improvement in computer processor speed and the increase in computer memory, Personal Computer (PC) was introduced and broadly accepted in early 1980's. GIS application architectures shifted to desktop-centric, with a large number of applications residing on the end user’s desktop machine, which may be connected to a local-area network for file and peripheral device sharing. Compare to the host-centric GIS architecture, desktop-centric GIS replicate all application processing, data management, and user interfaces and interaction functions on each desktop computer. While the adoption of local-area networks provides a way to share data and peripherals devices, application architectures remain very desktop heavy (Autodesk 1997).

As a result of the broad adoption and acceptance of network and application standards introduced and supported by the Internet, the world of information systems is entering into a third era of computing, an era based on network-centric application architectures. Network-centric architecture has dramatically impacted GIS industry. For the first time, cost-effective, enterprise-wide mapping and geographically enabled applications can be deployed to large numbers of end users. For the development and deployment of GIS applications to multiple users, network-centric GIS show significant, measurable benefits combined with high returns on investment. Network-centric GIS applications exhibit dramatically lower initial implementation and ongoing support costs, more efficient use of network computing resources, and enhanced access to corporate data resources. Applications that are network-centric also display higher levels of application performances, feature new distributed-processing features, and justify the extension of specific technologies such as GIS to a broader audience of end users (Autodesk 1997).

Client/Server architecture was adopted as the typical model of network-centric applications. In this model, many clients linked to a centric application server that contains application logic and have the connectivity to the centralized database. The concepts of "thin client" and "thick client" came out in this period in terms of the distribution of computing burdens between the client and the server. If the server does most of the processing, then it is a thin-client architecture. Conversely, if the client does most of the job, it is a thick-client application. Either thin-client or thick client architecture has many drawbacks in terms of taking advantage of network computing resources, such as data and processing tools.

With the expanding of networks, especially the widely adopted Internet, the computing resources are distributed world wide in the network. How to take good advantage of these resources becomes more and more important. The typical client/server architecture cannot solve this problem. Only recently, distributed computing technologies with open infrastructures made it possible. In the GIS community, Distributed GIS became a hot issue, open GIS architecture became the focus in both academic research and GIS industry.

2.1.2 GIS Trends

As component GIS became the standard in the GIS industry, network-centric GIS became a common concern of GIS software vendors, information providers and GIS users. GIS applications shows a very strong trend from proprietary systems to open systems. As shown in Figure 4, as GIS becomes more open, GIS user range expands from the GIS specialist to GIS professionals and further to the general public.

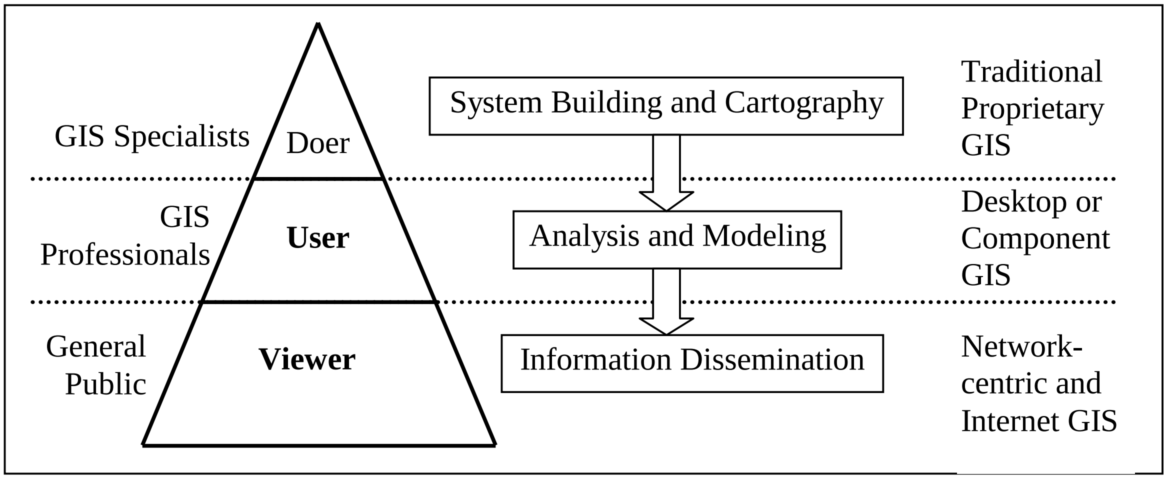

GIS Specialists
Doer
System Building and Cartography
Traditional Proprietary GIS
GIS Professionals
User
Analysis and Modeling
Desktop or Component GIS
General Public
Viewer
Information Dissemination
Network-centric and Internet GIS

Figure 4: Different Levels of GIS Users in the Progress of GIS toward Open

In the early stage of GIS development, GIS applications were often niche systems for certain GIS applications, and they were GIS specialist-oriented. The user group was very narrow and knowledgeable of GIS concepts. Since the main purpose of the application was to establish a certain application system, usually digitize paper maps, organize the data and output map results in the cartography style, this group of users were referred to as GIS Doers (Intergraph 1997). GIS functionality was focused on the data input, data processing, data management, and data presentation and output, such as digitizing, editing, topology building, map plotting, etc. Later when GIS system became more general to fit more applications, the request of GIS knowledge for the doers became even heavier if a good outcome was expected.

When GIS shifted its paradigm from mainframe computers to desktop computers, GIS users expanded to a larger group, the GIS professionals, which used GIS to solve their specific problems, such as data analysis and modeling. This group of users is often referred to as GIS user. System functions also expanded from system building to spatial analysis and modeling. As component GIS was widely adopted, GIS became much easier for the user to integrate GIS with other systems.

Today, the network-centric GIS, especially the Internet GIS, brings large group of users to GIS technology. These users are the general public that is usually referred to as the GIS Viewers. They have little or no knowledge about GIS technology, but can use GIS. For the viewers, the interests are in getting geo-referenced information and doing some analysis. As shown in Figure 4, this group of users forms the largest percentage of GIS users, so it is located at the bottom of the user pyramid.

The expansion of GIS end user not only brought a great market prospect but also triggered many technical challenges to the GIS industry. Many important issues are involved in the progress of becoming open. The top challenge is the interoperability of both geodata and geoprocessing software components (Buehler and McKee, 1998). The network-centric GIS architecture and distributed GIS services model are among the most important research topics (Smith 1996).

2.2 Network-centric GIS

The network-centric GIS introduced a new paradigm that enabled a broader variety of GIS applications and expanded GIS users. The new and innovative architecture delivered GIS for wide distribution across enterprises and organizations and made GIS accessible over the ubiquitous Internet.

2.2.1 Network-centric GIS architectures

The development of network technology facilitated the applications of network-centric GIS in two network levels: the Enterprise level and the Internet level. An enterprise network might be either the Local Area Network (LAN) or the Wide Area Network (WAN) and the Internet is also referred to as the World Wide Web (WWW). More specifically, GIS software exists in the following forms of network:

- Host-Terminal network: This is the old model of networking in which a mainframe computer acts as the host and many terminals are used to access the data and GIS functions. Since every computation is calculated in the host and the terminals are only used for display and interactions, this model has very high performance requirements to the host. The major problem of this model is its slow response speed, high cost, and difficulty of development. Current networked GIS discarded this model.
- Client-Server network: This model of network is widely exist within enterprises, in which some computers act as servers as well as others act as clients. The server computers usually have more power than the client and manage the centralized resources. Different from the old host-terminal model, the client machines in this model also have some resources and computational power that might be used to relieve the load of servers. This characteristic of client-server network made it faster, more flexible and less costly than the host-terminal network. Actually, the client-server network is the major form of network in the enterprises currently.
- The Internet: It is the largest network over the world that links many smaller networks together. The major characteristic of it is its wide adoption and good connectivity and accessibility. The Internet is the ideal network for information dissemination.

In GIS systems, like in other information systems, the major architecture adopted is the typical three-tier architecture in which a GIS application are partitioned into three tiers and distributed in different locations (Larman 1998 and Charles, et al, 1999), as shown in Figure 5:

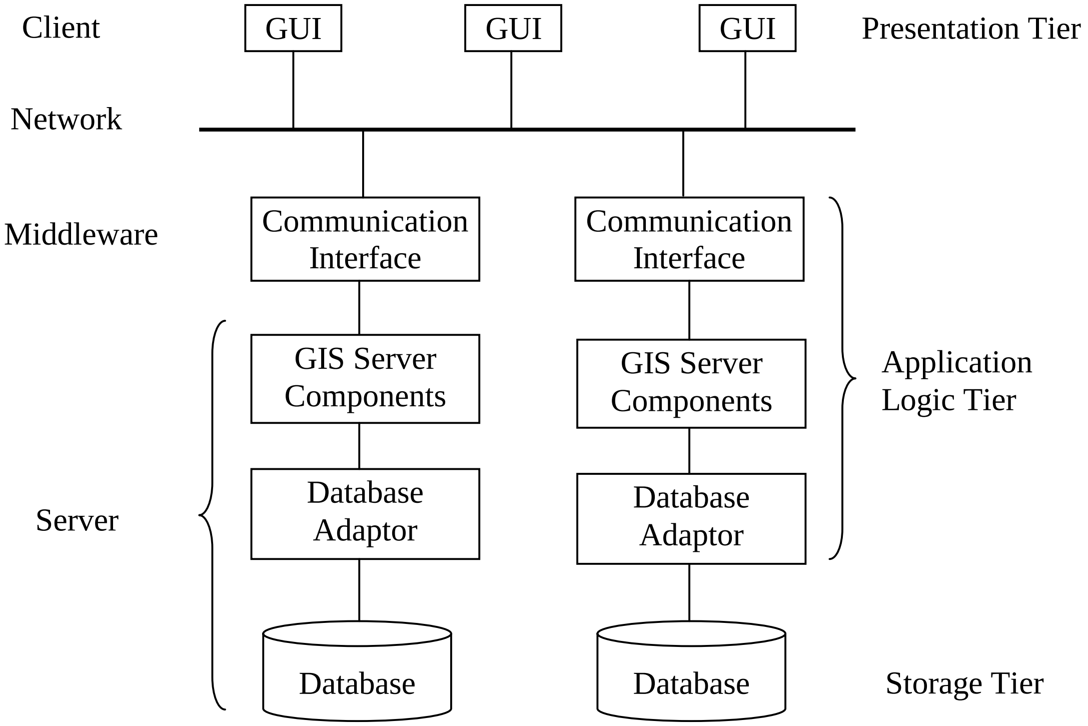


Figure 5: Typical Multi-Tiers System Architecture

- The Presentation Tier: contains the system User Interfaces for system operations. According to model-view separation principle in object oriented analysis and design (Larman 1998), the presentation objects should be invisible to application domain objects. In a network-centric GIS, the presentation tier should be located in the client side.
- The Application Logic Tier: contains the GIS domain components that model and process GIS data. This tier is the central part of a GIS system that links the presentation and the storage tier and performs the geoprocessing. It may be further

divided into many sub-tiers that form a multi-tier system. These sub-tiers might be network communication middleware, database interface, domain objects, and so on. In a network-centric GIS, the application logic tier is located in the application server in the server side.

- The Storage Tier: contains the databases that store the GIS data in data servers. This tier is linked to the application logic tier via the database connection objects in the application tier.

In a networked environment, a middleware that is responsible for the information communication between clients and servers is usually used in the network-centric GIS (see Figure 5). The client-middleware-server architecture is good to increase the interoperability of the proprietary systems, especially when the system is designed to use in a heterogeneous network environment (Wiederhold 1999), such as the Internet.

Some network-centric GIS systems are extended from traditional proprietary systems, which simply have the proprietary GIS running at the server site, and add a client interface at the client side and a middleware at the server side to communicate between the client and the proprietary GIS software. The thin client architecture is used in this case. In a thin client system, the clients only have user interfaces to communicate with the server and display the results. All the processing is done on the server.

Recent development in object oriented programming make it possible to produce software components and send them to the client before the running in the client machine, such as Java classes, ActiveX components and plug-ins. This comes out the thick client GIS. Java applet, Netscape plug-ins and Microsoft ActiveX component technology were involved in the structure. In the thick client architecture, the client machine does most processing work locally.

Both thin and thick client systems have some advantages and drawbacks, but they are not the best solution in terms of taking advantage of network resources. An ideal architecture should be a distributed architecture which can distributed both data and processing components anywhere within the entire network. We will discuss this in session 2.3 about Distributed GIS.

2.2.2 Web GIS Technologies

The rapid expansion of the Internet provides GIS communities with a new technology for disseminating Geographical Information (GI) to the general public. Web GIS or the Internet GIS is a new phenomenon in the recent years and become more and more popular (Jaakko, et al, 1999). A lot of approaches were introduced into GIS mainly to provide geographical information (geodata) in the Internet. The following depicts some of these techniques:

I. Common Gateway Interface (CGI)

CGI is the first generation technique applied to the Internet GIS. It was the standard way that connected the GIS application and the HyperText Mackup Language (HTML) in the web server to provide GIS information to the clients (Xia and Chao, 1995). HTML is a static text file, and it is designed for displaying text and image data in web browsers instead of processing the data, so it cannot produce dynamic information, such as GIS data processing.

As shown in Figure 6, CGI actually acts as the middleware between the web server and the GIS server. It accepts user's requests from HTML and then passes them to the GIS application. When the application finish the processing, the result is passed back to CGI. CGI dynamically generates HTML file that attaches the map on it and sends it to the web server. The web server then sends the HTML to the clients.

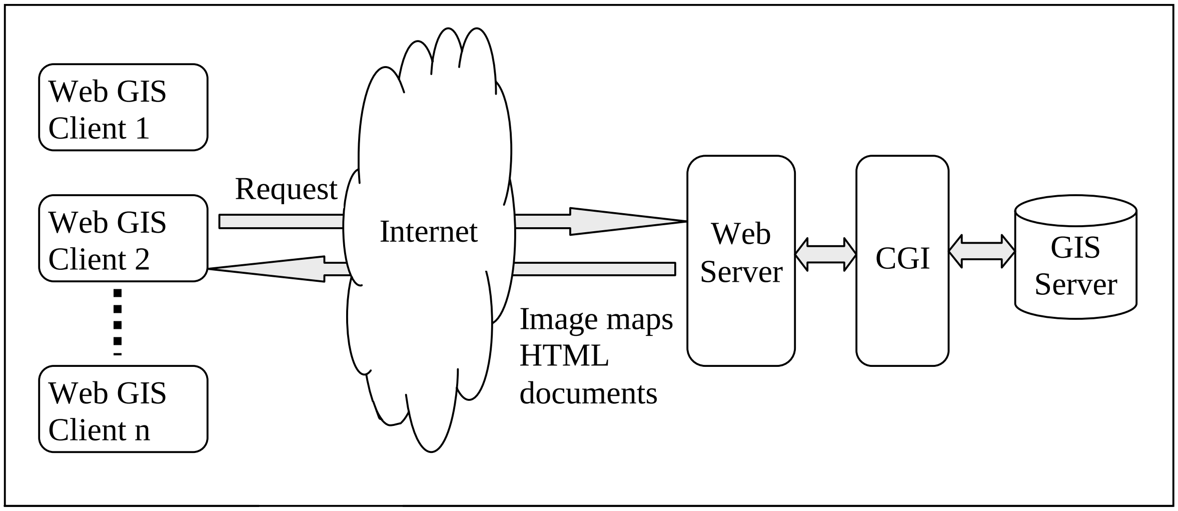


Figure 6: CGI approach in Web GIS

Since HTML only supports text information and static images, maps can only be transferred to the clients in image form.

The biggest advantage of CGI approach is its browser independence due to the standard HTML. Also the exist proprietary GIS systems can be easily extended to be GIS servers. However, the use of static maps in raster form, displayed in a web browser not providing any analysis functions, severely limits the facilities available to the end users (Jaakko, et al, 1999). The inability of CGI approach at providing vector data and GIS functionality directly to the clients makes it hardly meet the requirement of Web GIS development. New generation approaches that may achieve the same goal but has more flexibility were introduced in terms of providing functionality and variant data types to the clients, such as Java Servlet.

II. Plug-ins

Internet GIS based on the plug-ins method is the second generation web GIS (Xia and Chao, 1999). CGI based systems only provide clients with very limited GIS functionality, basically map navigation and viewing but lacks geodata processing capability. All of GIS processing including the simple ZOOM and PAN functions have to be done at the server side and the information provided to the clients is only static images. To solve this problem, some GIS functions should be moved to the client side and the basic vector data format should be supported. This functionality location change from the server to the client will greatly improve the response speed of the system and decrease the network traffic. One of the methods called "plug-ins" implements the principle by installing GIS software extensions to the web browser and exchanging information with the browser during user's operations. Plug-ins is a kind of Application Program Interface (API) provided by Netscape Communications Corporation to extend the browser's functions. GIS functionality plugged into a browser lets the browser has the ability of recognizing vector GIS data, and make it possible to process the geodata locally. Figure 7 illustrates the general process of plug-ins based web GIS systems.

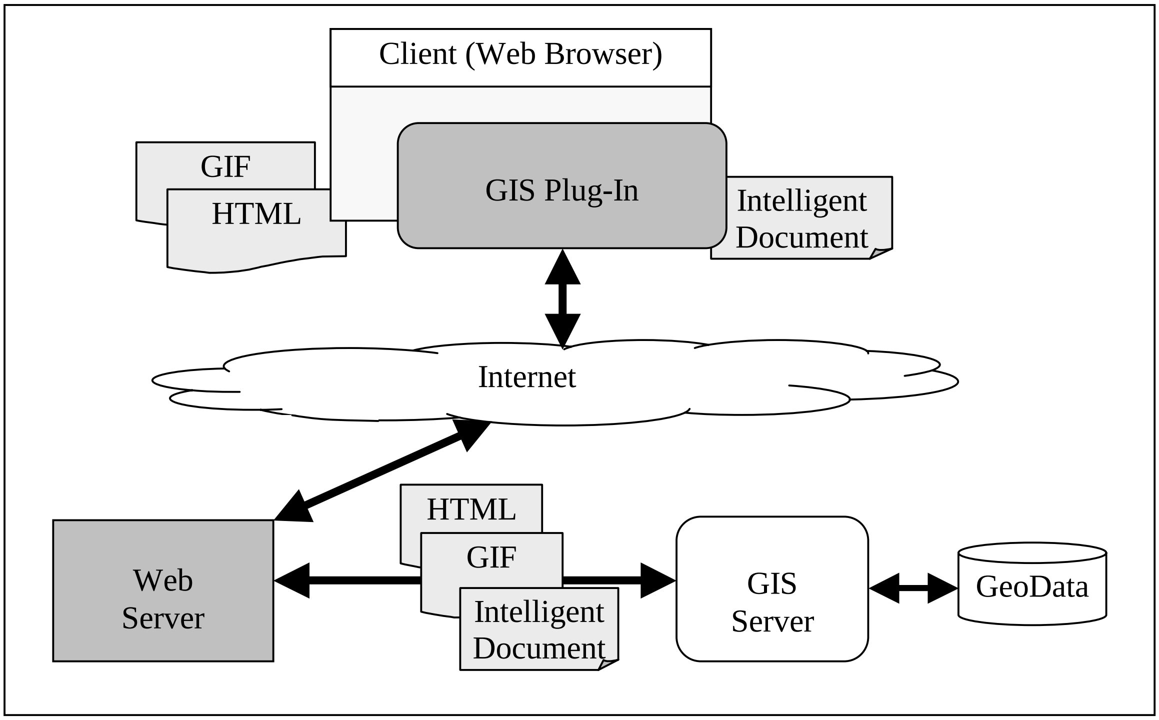


Figure 7: Plug-ins and Intelligent Graphics

Different from CGI approach, plug-ins needs to be installed locally. The compatibility in different platforms and version changes make it a problem, and this causes the inconvenience in system update.

III. Intelligent document

To date, most Internet GIS systems are focused on geodata catalog and geodata providing services. The main purpose of these systems is to let the user browse and use their georeferenced data, including both spatial and non-spatial. The large volume of spatial data and the lack of GIS functionality support of web browser were considered as the bottleneck in the Internet GIS services. Lots of efforts have been made to overcome the bottleneck problem and at last a compressed, vector data formatted, and self-contained intelligent document became the solution (Autodesk 1997, Intergraph 2000). These intelligent document is designed to be efficient for network transportation and it may comprise all necessary information, including general map properties, security information, map layer properties, raw map data, and user interface specifications. By expanding standard browser's functionality, map document can be intelligent in terms of containing self-processing tools and relevant information.

Maintaining all the necessary information in one file – an intelligent document – makes it much easier to support multiple users, since a single file update or refresh can update both a complete application and its associated data. This intelligent document would contain application-specific "meta-infomation" in addition to application data, including information on how the client application that loads the document behaves during the presentation of the document (Autodesk 1997).

WebCGM is an intelligent graphics profile of the Computer Graphics Metafile (CGM) standard, tailored to the requirements for scalable 2D vector graphics in electronic documents on the World Wide Web (Gebharlt and Henderson, 1999). It can be used to represent GIS information within Web documents. WebCGM is an "intelligent graphics" profile, which means that in addition to spatial content, the profile includes non-spatial contents. The non-spatial content allows the definition of hierarchies of application objects, as well as the association of metadata, such as link specifications and layer definitions, with the objects.

So far plug-ins is needed to let the standard browsers understand the intelligent file and provide GIS functionality support, see Figure 7. Many web GIS vendors have their own WebCGM based formats and plug-ins solutions, such as Autodesk MapGuide has MWF (Map Window File) and MapGuide plug-ins (Autodesk 1997), Intergraph GeoMedia Web Map has the Map Definition File (MDF) and ActiveCGM™ (Intergraph 2000), etc.

The adoption of another standard intelligent graphics document, the Scalable Vector Graphics (SVG) in the next generation browser file, the Extensible Markup Language (XML), will make the plug-ins that interprets the document unnecessary, since the next generation browser will understand the SVG file. Similar to WebCGM, SVG graphics can be dynamic and interactive.

Intelligent document can be efficient in terms of geodata publishing, but the GIS functionality supported is still limited. The most frequently used functions could be supported by browsers, however, we cannot count on browsers to support all geoprocessing functions. In this case, other approaches will be needed.

IV. Internet Programming Languages

Plug-ins can extend browser's functionality to process spatial data efficiently, however, this method have many drawbacks. Firstly, it will greatly increase the burden of the client and let the client become very fat (fat client), because most software vendors will have their own plug-ins to make Internet-enabled products. This is not compatible with the principle in the design of the standard browser, which is intended to be a thin client. Secondly, the management of too many plug-ins will become a big issue, as everyone can install the new version of plug-ins through the Internet. Internet programming language can be used to solve these problems.

The most popular Internet programming language is the Java language developed by Sun Corporation. Java is a new generation object-oriented (OO) language. It inherited the advantages of other object-oriented language such as C++ and Smalltalk but discarded the weak points of them, such as complexity, security problems, etc. Instead of producing binary codes for a specific platform, Java compiler produce byte codes that can run on any Java Virtual Machine (JVM). This makes Java platform independent, which is one of the most important issues in the heterogeneous Internet. Another important issue is the network security management, which Java controlled very well compare with other languages.

Java provides a series of solutions to replace old technologies: At the client side, it has Java Applet that can be downloaded to the client machine at run time. At the Server side, it has Servlet that can be used to replace CGI interface. In the client-server communication, it provides Java Remote Methods Invocation (RMI), which can be used to implement distributed computing tasks.

Another Internet programming technique is Microsoft's ActiveX, which is developed upon the OLE control (OCX). ActiveX can extend standard browser's functionality by loading the ActiveX components at run time. For example, GIS tools can be implemented in ActiveX components and embedded in the HTML file. The GIS tools can be loaded in the client machine when the client visits the web page. The functionality-expanded browser then can display and process the spatial geodata.

Figure 8 shows the major difference between Internet programming language method and other approaches described above is that, it allows the executable program to be transferred and run in the client machine at run time. The traditional software installation at the client side is ignored. This big change has a lot of advantages. It makes the deployment and version management of the software easier. Since the program is running on the client machine, the large volume spatial data is transferred only when the client needs new data from the server. This not only improved the response time greatly but also decreased the network traffic, which would be very heavy in other cases. Another important advantage of the Internet Programming Language method is that the whole structure of the program can be very flexible and scalable, this is very useful for developing distributed GIS models.

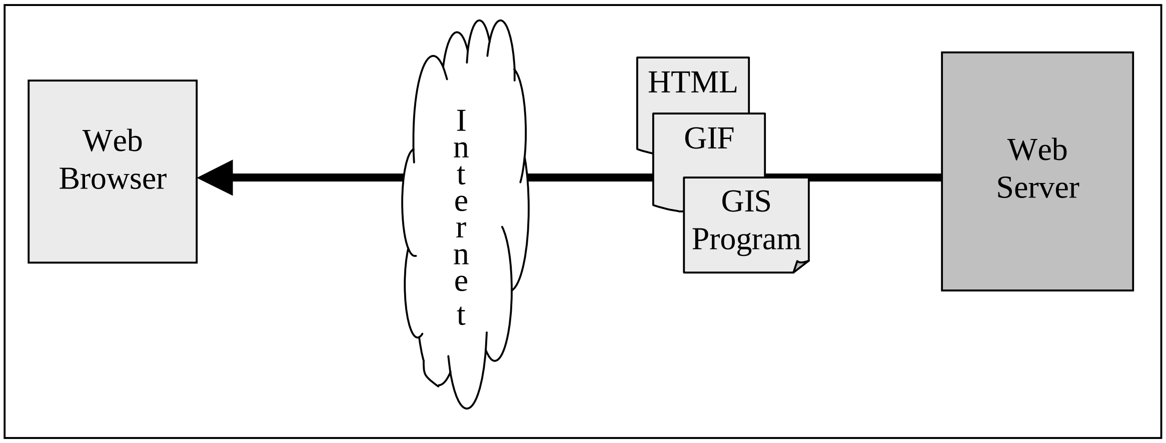


Figure 8: Internet Program Downloading

2.2.3 Existing Problems

By 1996, on-line web mapping tools were becoming available from virtually all of major GIS vendors, and they were a hot item (Thoen 1999). Unfortunately, almost all of current web GIS products are still only for geodata publishing purposes. Geoprocessing tools keeps as a weak point. It is expected that during the next decades, the market place will expand, bring many new tools to more users, and extend to the edge of corporate environments. All enterprise users will have access to geospatial data via web-enabled GIS tools (Asbury 1999).

The two types of GIS services, geodata services and geoprocessing services, should not be separated. The weakness in geoprocessing services will definitely have negative impact upon the geodata services. The geoprocessing services are of the equal importance with the geodata services. Currently, most Internet GIS products and applications are focused on geodata services: geodata catalog and geodata publishing. The approaches depicted above may be efficient in terms of providing centralized geodata services, however, they are not good enough for providing geoprocessing services. Even with hybrid method that combines CGI, plug-ins, intelligent graphics documents, and Internet Programming Languages, without a new distributed object model, only very limited GIS functionality can be provided to clients. The current client/server model, either a thin client or a thick client model, cannot meet the requirements of geoprocessing services, in which the interoperability of GIS components is the key issue. An open and distributed Internet GIS model needs to be developed.

Distributed object technologies can help in building an open architecture for GIS, in which a proprietary large GIS system will be split into many small interoperable GIS components. These components can be distributed in arbitrary sites within the network and will be able to process geodata distributed in the entire network. A "geodata anywhere, geoprocessing anywhere" model will be ideal for the providing GIS services, in which third party geoprocessing software components and geodata can be easily added into the GIS services (Tao and Yuan 2000a and 2000b).

2.3 Distributed GIS

Recent developments in information technology have resulted in a number of distributed object computing architectures that provide the framework required for building distributed applications that use distributed objects. The framework also supports a large number of servers and applications running concurrently. Many of such frameworks provide natural mechanism for interoperability (Kafatos, et al, 1999). For example, COM/DCOM architecture in windows platform, CORBA in a completely open environment and Java RMI in Java Virtual Machine (JVM) are the most popular protocols that are used in different cases. These architectures can be applied to GIS to improve the traditional client/server GIS model and develop a scalable distributed GIS model. Some attempts have been made in the academic area (Zhang 1998, Evans 1999, Charles, et al, 1999). The following sections will discuss these architectures in more detail.

2.3.1 Distributed Object Technologies

Distributed object computing extends an object-oriented programming system by allowing objects to be distributed across a heterogeneous network. Each of these distributed object components inter-operates as a unified whole. Currently, the three most popular industry standards for fulfilling distributed computing tasks are Microsoft's Distributed Component Object Model (DCOM), Object Management Group's (OMG's) Common Object Request Broker Architecture (CORBA) and Sun's Java Remote Method Invocation (Java RMI).

DCOM is an extension of the Component Object Model (COM) to the networked environments. It supports remote objects by running on a protocol called the Object Remote Procedure Call (ORPC). This ORPC layer is built on top of RPC and interacts with COM's run-time services (Gopalan HTML). The key features engineered into the COM/DCOM architecture comprise language independent and static/dynamic invocation of objects (Albertson 1998). Since the COM specification is at the binary level, it allows DCOM server components to be written in diverse programming languages like C++, Java, Visual Basic and so on. As long as a platform supports COM services, DCOM can be used on that platform. DCOM is now heavily used on the Windows platform (Gopalan HTML) and many practical enterprise level systems are based on this architecture. Unfortunately, it is still limited within the Windows platform currently and hardly used in the heterogeneous Internet.

CORBA, based on the Object Oriented (OO) technology from its very beginning, may be the most effective tool, to-date, for developing a large-scale, open-architecture, heterogeneous distributed systems. A CORBA implementation employs Object Request Brokers (ORBs), located on both the client and the server, to create and manage client-server communications between objects (Albertson 1998). CORBA replies on Internet Inter-ORB Protocol (IIOP) to inter-operate with objects. The ORB acts as a central Object Bus over which each CORBA object interacts transparently with other objects (Remote or Local). A CORBA object has an interface and supports implement language, operation system as well as platform independence. The problem with this architecture is that it is relatively new and probably too complicated to implement. Fewer practical systems have come out.

On the other hand, Java technology provides an easy way of platform independent. Java RMI may be the simplest and fastest way to implement a distributed object architecture, due to its easy-to-use native Java model. Therefore, it is a good choice for Rapid Application Development (RAD) and small-size prototype application (Albertson 1998). Java RMI relies on Java Remote Method Protocol (JRMP) and Java Object Serialization, which allows objects to be marshaled as a stream. Since Java Object Serialization is specific to Java, both the Java RMI server objects and the client objects have to be written in Java. This language limitation of Java RMI applications makes it difficult to communicate with existing legacy applications. Efforts are still being made to build bridges to solve the problem and there are some good results coming out. For instance, the Java RMI components can shift to CORBA components through certain bridges.

The three distributed object architectures that use different methods implement the same idea: Distributing objects are inter-operable across the network. Table 1 gives a brief comparison among the characteristics of the three architectures:

Table 1: A Comparison of CORBA, DCOM, and Java RMI Characteristics

	DCOM	**CORBA**	**Java RMI**
Protocol	ORPC	IIOP	JRMP
Language	independent	independent	Java
Platform	Windows	independent	independent
Tech. basis	COM	ORB	Object Serialization
Popularity	Windows Applications	Enterprise Applications Large Scale Applications	Rapid Applications Small Applications
Performance	High	Low	Medium

In choosing the "right" architecture for the distributed GIS model development, the performance of these architectures is another important factor to support the decision-making. An experiment to examine the network communication efficiencies among the two platform independent architectures, CORBA and Java RMI, as well as light-weight, but more primitive communication solutions such as socket and ftp, had been made by the Center for Earth Observing and Space Research. The experiment result shows that CORBA seems to be four times slower than RMI, 10 times slower than sockets, and 40 times than ftp in a LAN environment. Due to the large overhead over the Internet, the performance gap between these technologies becomes smaller. However, as understood from the LAN experiment, The performance of CORBA is much slower than RMI. It is possible to infer that CORBA's performance is perhaps three to four times worse than RMI (Kafatos 1999).

2.3.2 Open GIS Framework and Specifications

As network-centric GIS booms, many Internet GIS service systems, primarily geodata service systems, became available (Limp 1999). When clients need to get GIS resources from different sites distributed all over the world, the interoperability of geodata and geoprocessing then becomes an important issue. The objective of interoperation is to increase the value of information when information from multiple sources is accessed, related, and combined (Wielderhold 1999). From geodata point of view, the interoperation of information solves the integration of geodata from multiple sources, which overlaps with the topic of data warehousing or geodata services. From geoprocessing point of view, the interoperability involves the integration of software components that come from different sites, which is the key issue of distributed geoprocessing services in the network.

The Open GIS Consortium, OGC, was founded in August 1994 when the component technology was the major innovation. Since that time, the world has been exposed to another paradigm shifting technology: the Internet. From the perspective of end-users, this is far more profound a development than the ability of components that make the task of building software easier (Cuthbert 1999). The major objective of OGC is to develop a series of specifications to increase the interoperability of geospatial data and geoprocessing software components. The OGC provides an open process for defining specifications designed to help promote interoperability between geospatial data and geoprocessing and allow the development of multi-tier solutions. The concept of open refers to the fact that these tiers need not all be implemented by the same vendor/developer. The GIS Service was proposed in the Open GIS Guide (Buehler and McKee, 1998) as an important concept for the open GIS architecture in an open network environment, such as the Internet. GIS services become so popular in both GIS industry and academic research that some scholars even consider that GIS has come into a brand new era: From the Geographic Information Systems (GISystems) to the Geographic Information Services (GIServices) (Müller 1999, Tao and Yuan 2000a). This consideration is reasonable since GIS is undergoing a series of evolutions in this Internet era: large GIS systems are spitting up into smaller components, Internet GIS, Distributed GIS Service systems are booming.

Open GIS means open and interoperable geoprocessing, or the ability to share heterogeneous geodata and geoprocessing resources transparently in a network environment (Schell 1999). The OpenGIS specifications will cover all of important aspects of GIS towards open. From abstract specifications to implementation specification, from semantics to software components, from common geodata model to common service model, it defines a series of interfaces in different levels. From software engineering perspective, these interfaces define the middlewares between data sources, GIS application components, and user interface. Figure 9 illustrates the role OpenGIS interfaces will play in an open GIS system. GIS services systems should follow the OpenGIS concepts and interfaces to increase the interoperability of geodata and geoprocessing components.

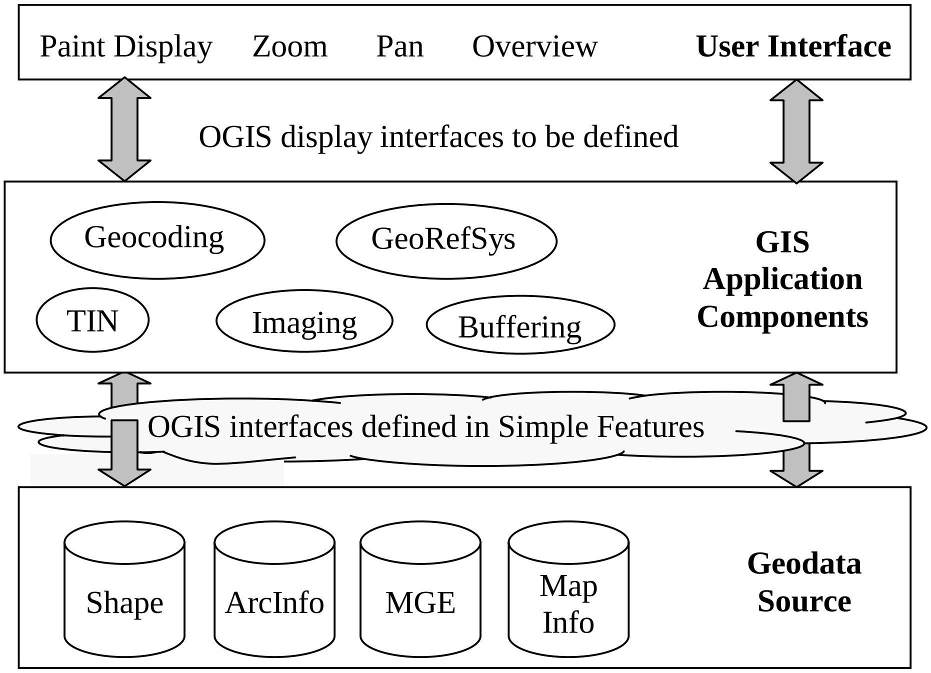


Figure 9: OpenGIS Interfaces in a GIS (Partial: Buehler 1998)

2.3.3 Technical Issues for Distributed Geoprocessing Services

The technologies underlying Distributed GIS Services can be grouped into four categories: user interface, communication, connectivity, and service infrastructure. Combination of these four categories defines a GIS service model. For example, if one takes a web browser as user interface, HTTP as communication protocol, CGI for connectivity, and geodata catalog as service infrastructure, then one obtains the typical geodata service model.

User interfaces for Internet services include simple web browser, program interfaces embedded in a web browser, or stand-alone programs that embedded communication protocols for remote services. Among the most important technologies for browser-embedded GUIs are makeup document languages (HTML, HTML forms, and XML), JavaScript, plug-ins, ActiveX components, and Java Applets.

Communication refers to the protocol or method used in client-server communication. Examples of communication include stream socket protocols, Internet application protocols (HTTP, FTP, SMTP), remote function calls (RPC, DCE) and distributed object environment (RMI, CORBA, COM/DCOM).

The term connectivity refers to the different ways of wrapping existing software in order to make it accessible via one of the communication protocols, such as CGI script which connects GIS and web server, callable libraries which invoke via a daemon, object wrappers that give the software an OO interface. The service infrastructure refers to service framework such as repositories, catalog, service registration and execution planning (Müller 1999).

Figure 10 summarizes the different levels of solutions for user interfaces, communication and connectivity conbinations. Increasing distance from origin represents higher level complexity and functionality. As shown in the figure, solutions formed from the same level are more feasible and efficient than those formed from different levels.

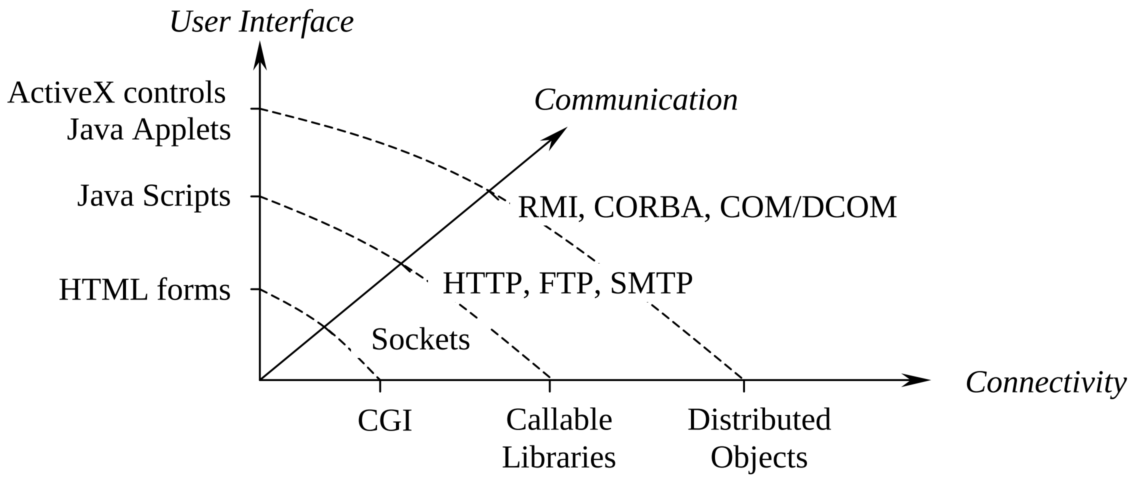


Figure 10: Technologies for Distributed GIS Services (Sources: Müller 1999)

For the development of a distributed geoprocessing service model, important issues include:

- The choice of distributed object technologies that will be applied to the model;
- The method used for user interface;

- The way of defining geospatial components;
- The service infrastructure that will be used to support the open access of both geodata and geoprocessing components.

In the next chapter, the solutions for these issues will be given and a distributed geoprocessing service model will be established as the result.

CHAPTER 3 DISTRIBUTED GEOPROCESSING SERVICE MODEL DESIGN

3.1 General Design

This section depicts the general considerations of the systems design: an ideal conceptual model, possibility analysis and solutions, and general architecture.

3.1.1 An Ideal Conceptual Model for Distributed Geoprocessing

The general idea of the distributed GIS service model is that a client program, either in an Internet browser or an independent application, should be able to access the resources distributed in the entire network. The resources here refer to both geodata and geoprocessing components available in the network. The client and the server in this context do not refer to a specific machine. Any machine, when it requests the remote resources during the processing, is a client, and any machine that provides such resources is a server. In a specific program, a client may connect to several servers if needed and a specific machine may be the client at one time and the server at another time.

An ideal distributed GIS service model should be a "geodata anywhere, geoprocessing anywhere" model, which means the geodata and geoprocessing tools could be distributed with the largest flexibility – virtually anywhere in the network. The geodata and geoprocessing components do not have to be in the same site, but they should be able to cooperate or integrate whenever they are needed to finish a specific task. Figure 11 illustrates this conceptual model.

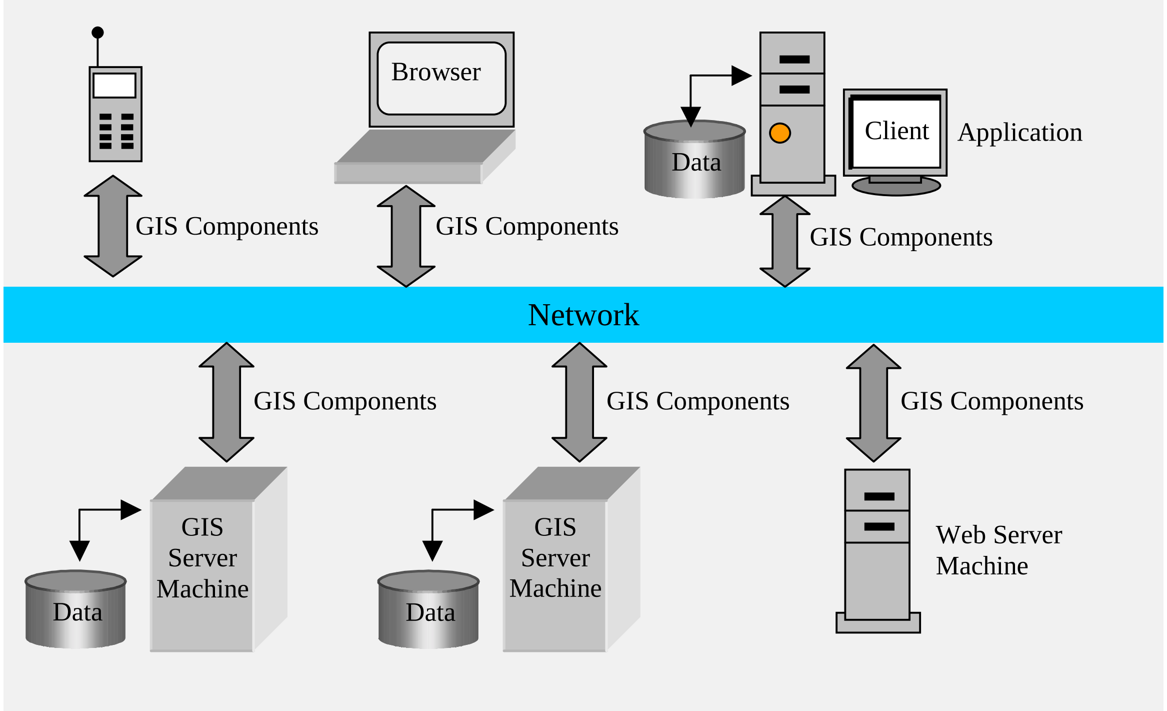


Figure 11: Conceptual Model of Distributed Geoprocessing

In Figure 11, GIS components are transferred across the network, which may be geodata components, geoprocessing components, or both. According to the actual needs in a task, these components could be as simple as basic data primitives such as integers or doubles, or be as complicated as a whole map, be as small as a single class, or be as large as a whole package or system. In this conceptual model, there is no GIS System concept, but instead GIS Service concept. A GIS system is splitted up into many interoperable components and may be distributed in different sites. Distributed geodata sets can be accessed by geoprocessing components that contain behaviors to manipulate the geodata sets depending on the actual data model. This is a complete distributed model in which any physical machine can communicate with other machines in the network.

3.1.2 General Analysis and Solutions

The design of the distributed geoprocessing service model is based on the following fundamental considerations:

I. Possibilities of Geodata and Geoprocessing Distribution

According to the ideal conceptual model that support "geodata anywhere, geoprocessing anywhere", the geodata and geoprocessing functions could be distributed in different sites (Tao and Yuan, 2000b). The following gives the possibilities of distribution (***P***) of the geodata and geoprocessing tools in one location, no matter it is at a client site or at a server site. Let ***D*** represent geodata, and ***F*** indicate geoprocessing tools (Functions). Using the following notations:

D_a : all of geodata, F_a : all of geoprocessing tools,

D_p : part of geodata, F_p : part of geoprocessing tools, and

null: none of geodata or geoprocessing tools.

Then, we have:

$\boldsymbol{D} = \{ D_a ,\ D_p ,\ null \}$ and $\boldsymbol{F} = \{ F_a ,\ F_p ,\ null \}$

$$\boldsymbol{P} = \boldsymbol{D} \times \boldsymbol{F} = \{ \quad \begin{array}{lll} (D_a,\ F_a) & (D_p,\ F_a) & (null,\ F_a) \\ (D_a,\ F_p) & (D_p,\ F_p) & (null,\ F_p) \\ (D_a,\ null) & (D_p,\ null) & (null,\ null) \end{array} \quad \} \qquad (3\text{-}1)$$

Table 2 gives a detailed description of these possibilities:

Table 2: Possibility of Geodata and Geoprocessing Distribution

$(null, null)$	Only exist at the client side, which means that neither geodata nor geoprocessing tools exist at the client side. The client in the typical client/server GIS model fit this case. In this case, the client is a thin client.
(D_a, F_a)	Typically exist at the server side in an typical client/server GIS architecture, which means that all geodata and geoprocessing functions are provided by the server. Since the server has everything, its clients should be thin clients (see the (*null, null*) case). Standalone GIS systems also fit this case.
(D_p, F_a)	A site has partial geodata but all geoprocessing components. It could be a GIS server that does everything for its clients or could be a GIS client that is very thick.
(D_a, F_p)	The site has part of geoprocessing components but all of the geodata. It could be a client or a server in a distributed system.
(D_p, F_p)	The site has partial geodata and partial geoprocessing components. This could be a typical case for both client and server in a distributed system.

(*null*, F_a)	The site has all geoprocessing components but none of geodata. The geodata are separated from the geoprocessing and distributed in other places. Similar to the case (D_p, F_a) and (D_a, F_a), since the functional components are centralized only at one site (client or server), the client will be either a thin client or a thick client.
(*null*, F_p)	Typical case in a distributed GIS system. The site does not have any geodata, but has some of geoprocessing components.
(D_a, *null*)	This may probably exist as a data center. Geoprocessing may be provided by other sites.
(D_p, *null*)	Partial geodata but no geoprocessing tools.

As in case (*null*, F_a), (*null*, F_p), (D_a, *null*), and (D_p, *null*), it is quite clear that the geodata and the geoprocessing tools are not always located in the same location. Both geodata and geoprocessing components could be distributed. A distributed GIS service model should be able to deal with distributed geodata and geoprocessing components.

II. Three-tier Location Adjustment

In the typical three-tier architecture (see Figure 12), the application logic objects only exist in the middle tier. The presentation tier and storage tier do not have application domain objects. In a typical client/server system, the presentation tier is located on the client machine and the application logic and storage tiers are located at the server side.

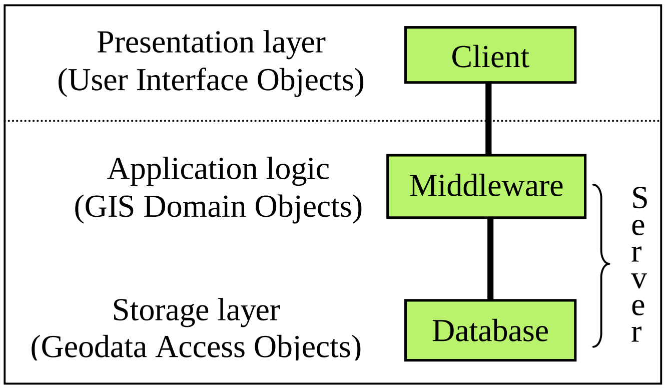


Figure 12: Typical 3-tier Object Distribution

In a distributed GIS system, locations of geodata and the geoprocessing components need to be flexible: the geodata storage and application tiers will be split up and distributed at different sites. For the client, the presentation tier is always located there, but it may also contain some application domain objects and data access objects. It is possible for a client to have all the three tiers. Figure 13 illustrate the redistribution of the three-tier objects in the distributed GIS:

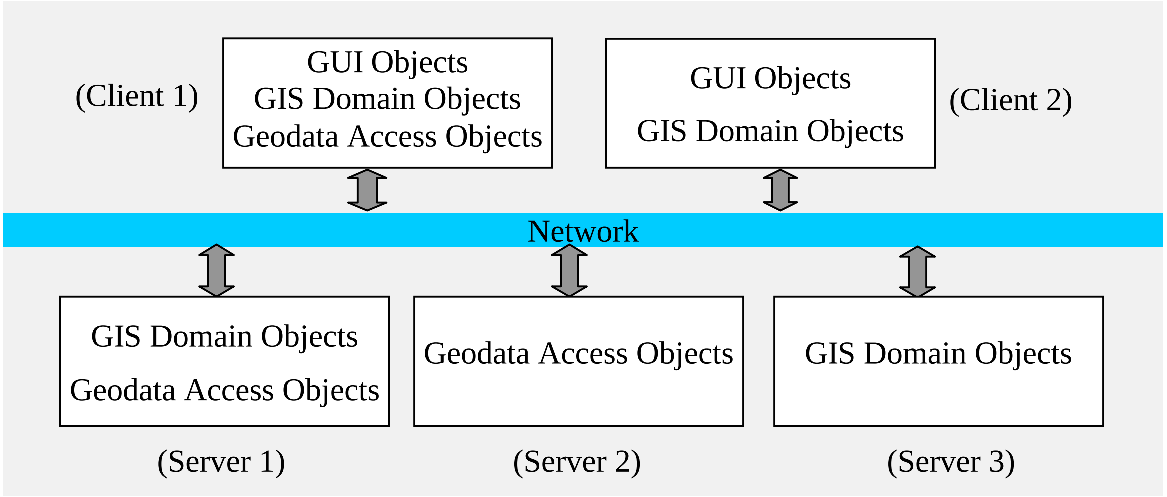


Figure 13: 3-Tier Objects Distribution in Distributed GIS Model

III. Distributed Object Technologies

Distributed geoprocessing model depends largely upon distributed object technologies. As depicted in section 2.3.1, since Java RMI provides an ease-of-use and relatively efficient way for distributed object computing (compare with CORBA), Java RMI is a good option for the solution of remote object communication. In fact, as a new generation object oriented programming language, Java is widely known as the Internet language due to its platform independent and good network security features (Gopalan HTML). In this development of distributed geoprocessing model, we chose Java technologies to provide the fundamental support to the model.

The Java family provides a series of ease-of-use solutions for distributed computing. Java Applet can provide user interface and application classes to the client through the Java-enabled browsers. Java Bean provides a way of producing interoperable components. Java RMI enables the remote object invocation. JDBC or JDBC-ODBC provides a universal means of connection between the internal objects and the external databases (see Figure 15).

Java RMI is a pure Java distributed computing solution. In its architecture, code on a client computer invokes a method of an object on a server. The client-server terminology applies to a single method call only. The machine that calls the remote method is the client and the computer that hosts the objects and processes the call is the server (Cornell and Horstmann, 1997). The client and server are not fixed in roles and it is entirely possible to change their roles in one application. In the Java RMI model, the geoprocessing functions may be distributed in any machine of the network and the resources of the network can be fully utilized.

As shown in Figure 14, Java objects are passed between the client and server in the form of marshaled Stub or Skeleton objects. A typical scenario of remote object passing is as the following: when the client code needs to invoke a remote method in a remote object, the parameter objects are encapsulated into a stub object. This object is then marshaled and passed to the server. On the server side, a skeleton object that makes sense out of the parameters in the marshaled stub object, and passes the parameters to the actual object to execute the remote method. After finishing the calculation, the returned object is then passed to the client in the same way.

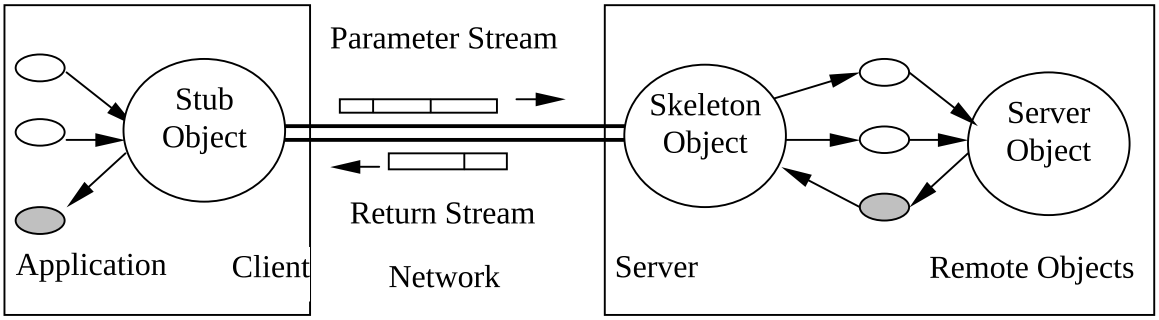


Figure 14: Java RMI object passing procedure

By applying Java technologies to the distributed geoprocessing service model, we can end up with a computing architecture. As shown in Figure 15, Java technologies provide solutions for both client and server sides.

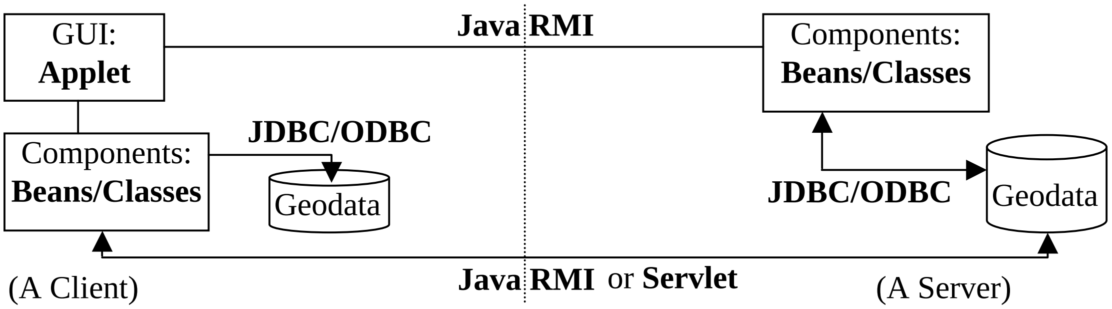


Figure 15: Java Solutions for Client and Server in Distributed Geoprocessing

For the Internet, Java applet is a good choice and used by many service providers to provide user interfaces as well as associated classes to the clients. When a user visits a web page with embedded applets, the user interfaces and the support classes are downloaded into the client machine and run locally. Once the basic application logic and data access objects are transferred to the client machine, the client will have all the 3 tiers and can do many works locally. A client can also access the remote geoprocessing components or data sets that are available in the Internet if he wants to perform a task that the local components do not support. The Java RMI provides a flexible and efficient solution to the remote communication and object transfer. An alternative method of Java RMI for remote data accessing is Java Servlet, in the case that the geodata server and the web server physically locate in the same machine. Java Servlet is a very similar to the CGI method in principle, but it is a more advanced technology in which objects can be transferred directly between the applet and servlet.

IV. DGIS Componentization Issues

GIS Componentization becomes one of the major GIS trends and keeps a hot topic in GIS community in recent years. A lot of articles discussed the importance, implementation methods of designing and building reusable, interoperable GIS components (Kuhns 1998, Tang 1998, and Engen 1997) and different GIS software vendor have their own solutions (ESRI 1999, Intergraph 1999, and MapInfo 1999). For the developing of a distributed GIS service model, the componentization of GIS is a more critical issue than that in building a desktop GIS application. The reusability and interoperability of component must be extended to the entire network.

The OpenGIS abstract specifications have a topic on OpenGIS Service Archetecture, in which GIS functionality is split up to many services (OpenGIS consortium 1999a). These services are:

- Geospatial Domain Access Services
- Feature Generalization Services
- Geospatial Information Extraction Services
- Geospatial Coordinate Transformation Services
- Geospatial Annotation Services
- Image Manipulation Services
- Feature Manipulation Services
- Geospatial Analysis Services
- Image Geometry Model Services
- Geospatial Symbol Management
- Image Map Generation Services
- Image Synthesis Services
- Image Understanding Services
- Geospatial Display Services

This classification is somehow in a general sense and is from the potential application perspective. From a software engineering perspective, GIS components may be grouped into two classes: the common objects and service specific objects. The common objects provide a fundamental support for GIS services, as listed above. The service specific objects support specific services only.

The common objects should include basic GIS geometry objects (Point, Polyline, Polygon, etc.), basic GIS data structure objects (such as Map, Abstract Layer, Recordset, etc.) and GIS service interface objects. The abstract layer should be the super class of all concrete layers such as ShapeLayer, ImageLayer, TINLayer, etc. The concrete layer definition should be grouped into the service specific objects that provide the specific service. If the system choose to support a certain type of geodata, the corresponding layer definition and the specific data access objects can also be included into the common objects. For Instance, if the service system choose to support shape file as in its basic part, the ShapeLayer definition and shape file access objects should be included into the common objects of the system.

Some services such as Geospatial Display Services are the most basic service that should be always available to the clients. They are "most basic" because without them, the clients could not do anything. Objects support these services should also be a part of the Common Objects.

Service specific objects should be based upon the common objects, but separated from them. It could be the extension of the system when a client requests the specific service. Different services should be independent but interoperable. The communication between different services should be based on the common objects. For instance, if a client chooses the Geospatial Coordinate Transformation Services, the Geospatial Reference Objects should be accessible to the client by either extending the client objects locally or using objects remotely.

Figure 16 summarizes the above discussions.

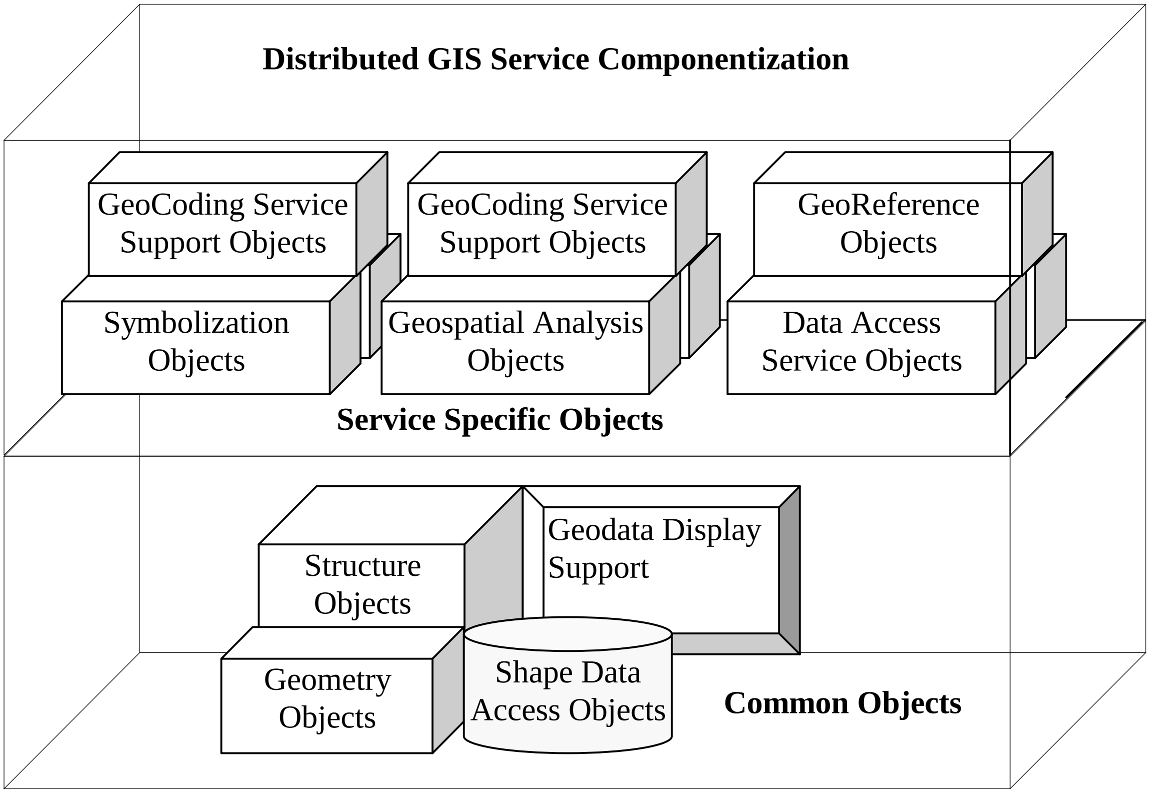

Distributed GIS Service Componentization
GeoCoding Service Support Objects
GeoCoding Service Support Objects
GeoReference Objects
Symbolization Objects
Geospatial Analysis Objects
Data Access Service Objects
Service Specific Objects
Structure Objects
Geodata Display Support
Geometry Objects
Shape Data Access Objects
Common Objects

Figure 16: Componentization of DGIS Service System

3.1.3 Architecture

From those considerations described in the last section, A Java applet and RMI based distributed geoprocessing model is developed. As illustrated in Figure 17, Java applet is used for providing clients with the presentation layer as well as the common GIS domain objects.

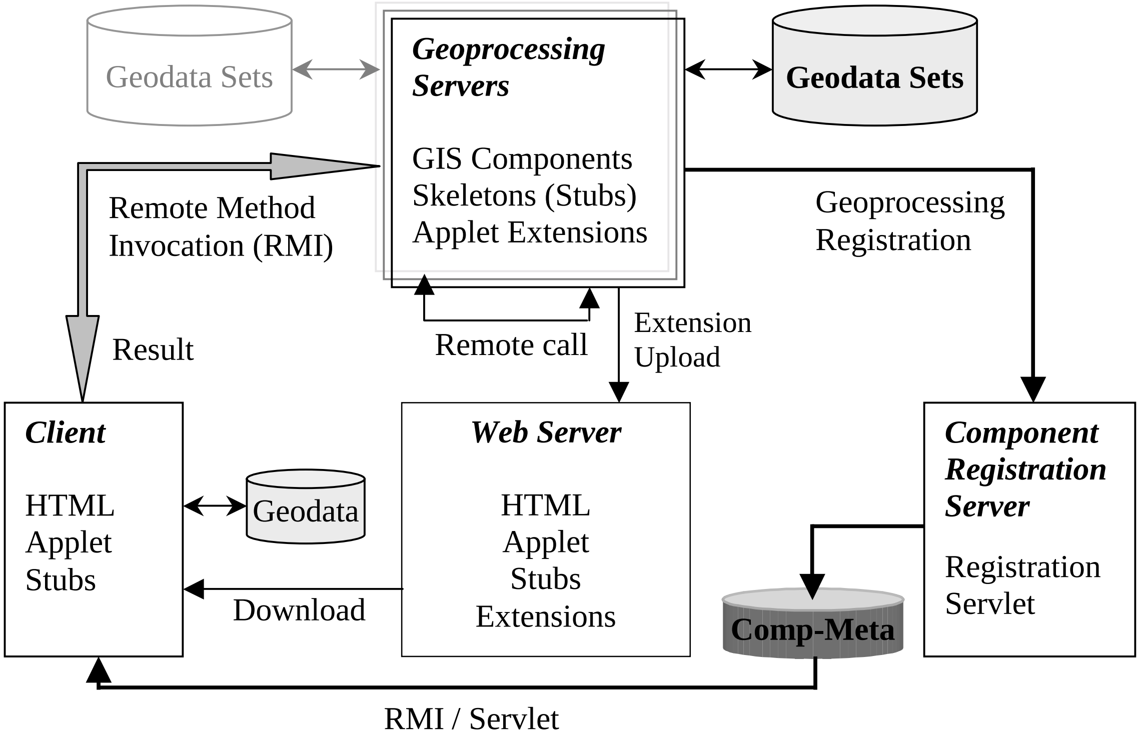


Figure 17: A Distributed Geoprocessing Model Using Java Applet and Java RMI

In Figure 17, the relationships among the web server, the component registration server, a collection of geoprocessing servers, and a web client are given. Possible geodata locations are also shown in the figure. Arrows in this figure represent the directions of data or component flows.

Since an applet must be embedded in a web page, a web server is used in the model. User interfaces, the common GIS classes, applet extensions, and remote object interfaces (stubs) are downloaded to the client machine when the client visits the web page. Multiple geoprocessing servers may be involved to offer the client with different geoprocessing tools via Java Remote Method Invocation (RMI) or the Java Applet Extensions. These tools shall be registered in a geoprocessing component registration server before they can be accessed. Component Meta-Information stored in the component mata-infomation database will be provided to clients either using RMI or Java Servlet. The applet extensions or component interfaces should be uploaded to the web server so that the clients can download them at run time. A geoprocessing server may also make remote calls to other geoprocessing servers for support, so even a single geoprocessing function can be distributed over the network.

In this model, geodata and geoprocessing components are accessible by a client via Java RMI and Java applet. This model is a dynamic model in terms of providing clients with scalable geodata and geoprocessing component accessibility. The applet contains user interfaces and most frequently used geoprocessing tools (provided by the GIS common objects) at the beginning, and it makes the local data access possible and common GIS operations much faster. This is not a thick client because the common objects could be light-weighted and extendable by RMI remote computing.

This model is compatible with the ideal conceptual model in Figure 11. "Geodata anywhere and geoprocessing anywhere" can be implemented by allowing any other server components to be registered into the model. The component registration is the key part of the model and it will be depicted later in more details in section 3.3.

3.2 Geospatial Data Model

A geospatial data model is a computerized representation or abstraction of real-world entities and phenomena. It defines basic GIS data types and their structure, which forms the foundation of a GIS system. Since many users of geographic information view the world in two dimensions, 2D geospatial data models are adopted in most GIS systems. Three major approaches are usually used to model the geospatial data in different systems: feature-based, georelational and object-oriented approach (Gardels HTML). Among them, the georelational approach is mostly used which combines features, maps, images and databases. Essentially, georelational systems manage spatial and non-spatial data separately and every spatial entity is linked to a database record using a unique ID. The object-oriented data model is an emerging alternative for the current georelational model, in which both spatial and non-spatial information elements are considered as attributes of geographic objects. Some researches (Abel 1998) and commercial systems

(ESRI 2000) adopt this approach as their common data model. This thesis is not intent to develop a novel geospatial data model, but practices the existing approaches – the georelational and object-oriented approach – to support the distributed geoprocessing model.

The following sections depict the geospatial data model designed for a prototype system of the above proposed distributed geoprocessing service model, from the abstract data model to the geospatial data organization, from basic geospatial geometry to reference system. Unified Modeling Language (UML) notation, which is described in Appendix A, is used the figures that represent the data models.

3.2.1 Abstract Geospatial Data Model

Basically, the prototype system to be built for this geoprocessing service model follows the typical Map-Layer-Geometry structure. Noticed that most current geodata are organized in georelational formats and the fact that object-oriented data model will be the next generation geospatial data model, in the prototype system, geometry objects are designed to be either separated with their attribute records or integrated into geo-objects according to the specific applications. In a distributed environment, the geodata objects will provide clients and servers with the basic understanding of the geodata during the communication, as the geo-objects are passed in the network as parameters. The following conceptual data model defines the basic geospatial types and their relationships in the prototype system, as illustrated in Figure 18.

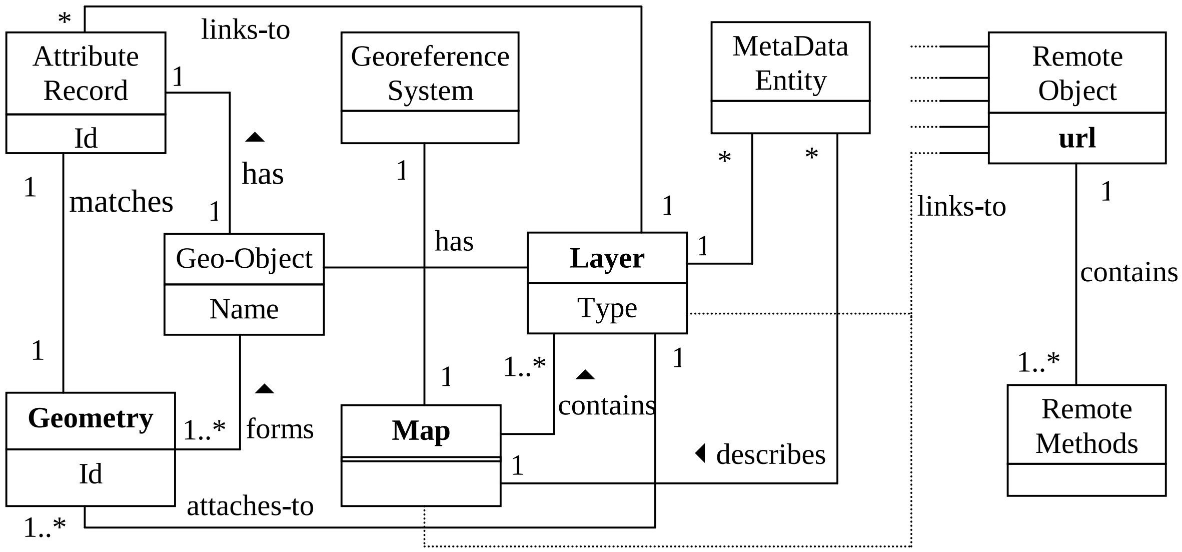


Figure 18: Abstract Geospatial Data Model for the Prototype Distributed GIS

(See Appendix A for data Model Notation)

In this model, basic geodata types are defined, such as geospatial geometry objects (Point, Polyline, Polygon, etc.), geospatial reference system objects (Datum, Ellipsoid, Projection, etc.), Metadata entities (Accuracy, etc.), and so on. A Geo-Object type is defined to integrate the geometry and the corresponding attribute records into single geospatial objects, such as a river which contains many sub-rivers or a city that may contain a lot of sub-objects. The geospatial reference system is applied to Map only since layers in one map usually have a unified reference system. Both Map and Layer objects may have Metadata entities to describe the data quality and other meta information of the map or layers.

Unlike a desktop GIS data model, remote objects are included in the data model in order to support remote communications. Remote objects that link to the internal objects (such as Map or Layer) provide remote methods to clients. The dotted lines that link to the remote object in Figure 18 are only examples of the relationships between remote objects and internal objects. In fact, any internal object may connect to the remote objects if the remote methods are required.

3.2.2 Data Structure Objects

The above abstract model is controlled by the Map-Layer-Object structure, in which the Map is the top-level type that connects to the user interface objects. A map may contain multiple Layers, which may be in different layer types. A layer may contain many Geo-objects or geometry elements that connected to the attribute records. The Layer should be an abstract class of all concrete layers and defines the common features of them. Each concrete layer has its interface to the data access objects. Figure 19 shows this structure.

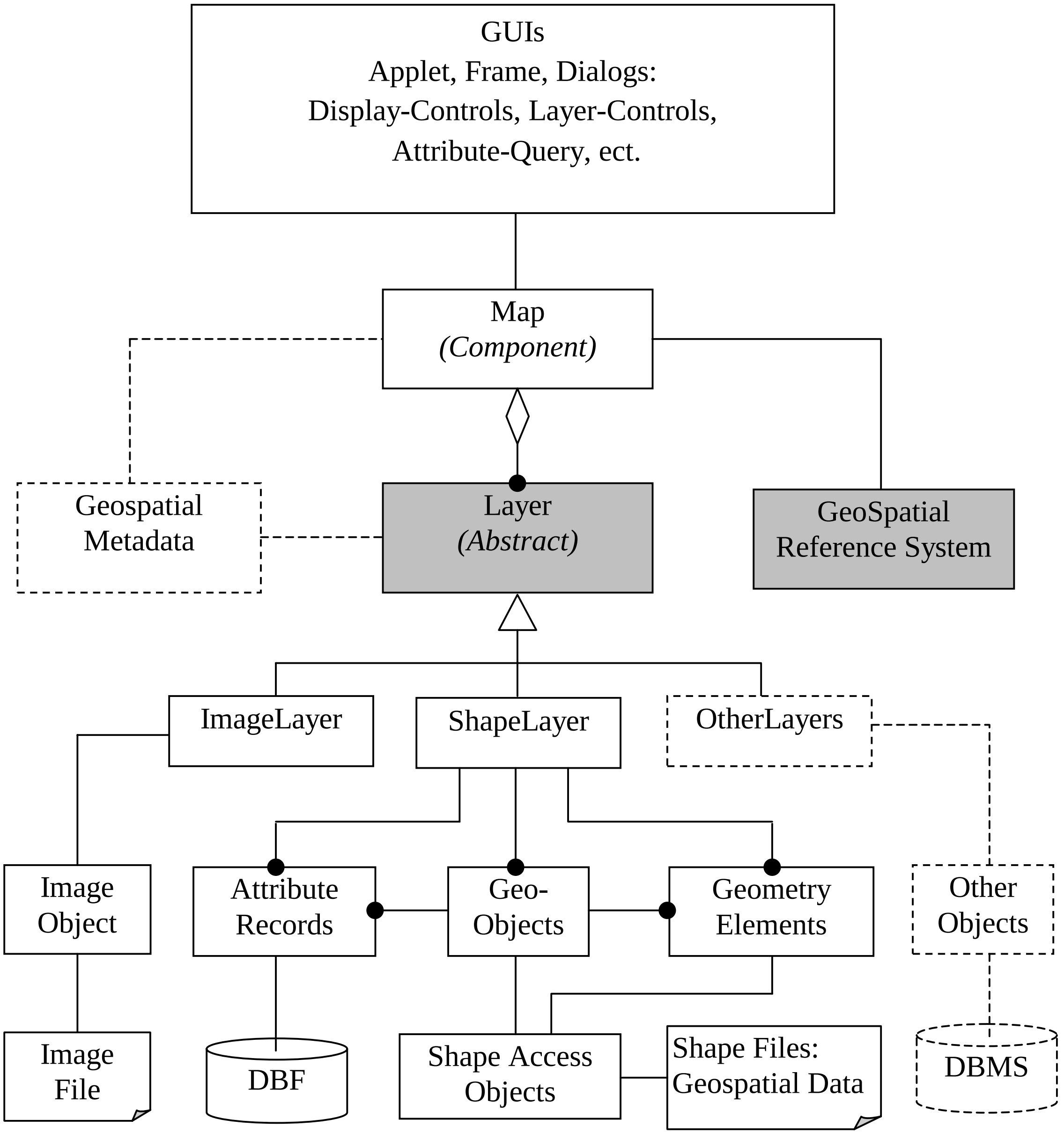


Figure 19: Structure of the Geospatial Data Model

(See Appendix A for data Model Notation)

As shown in Figure 19: Each concrete layer has its own data access objects which connect spatial data files or database with geometry elements and attribute records of the layer, or geo-objects that integrate geometry elements and attribute records of the layer. Take the ShapeLayer for example: ShapeLayer is a concrete layer type for manipulating ESRI Shape data. The shape data are composed of two parts: The spatial data stored in plain files and their corresponding attributes are stored separately in dBase database. So the data access objects of ShapeLayer contain two parts: Attribute Records for the attributes from DBF database and Shape Access Objects for the spatial data from shape spatial data files. The spatial data form the geometry objects and the attribute data form the Attribute Records Object. These two kinds of objects can be related via the unique ID and attach to the ShapeLayer directly or integrate into more complicated geo-objects as needed and attach the geo-objects to the ShapeLayer. One ShapeLayer may contains many geometry objects and many attribute records, or many geo-objects, and one geo-object may contain many geometry objects and many attribute records.

In Figure 19, the shaded boxes are the abstract data types. They will be implemented as abstract classes in the implementation. The boxes and the association lines in dashed lines are for the extension of the geospatial data model. They will not be implemented in the prototype system for this thesis. Refer to Appendix A for the data model notations used in this thesis.

3.2.3 Geometry Objects

As the most basic part of a GIS system, geographical geometry had experienced several decades' development and came to mature. Different GIS systems currently apply similar geometry model in their geodata model, despite there are minor differences between them. The typical geometry model uses Point, Polyline and Polygon as the primitive data types of two-dimension geometry, and defines 2D spatial relationships among them as operations (methods). The OpenGIS specifications also followed this typical model, but defined many interfaces to standardize the operations. There are two implementation specifications in OpenGIS specifications: OpenGIS Simple Features Specification for OLE/COM (OpenGIS Consortium 1999b) and OpenGIS Simple Features Specification for CORBA (OpenGIS Consortium 1998). However, there is no current implementation specification available for Java.

The geometry objects for this prototype system are defined at the beginning of the research, and at that time the OpenGIS Simple Features Implementation Specifications were not available. However, the basic structure of the geometry model is similar to the specifications as a result, despite the differences in names, function parameters and locations. It is much better to completely adopt an OpenGIS specification. However, due to the time limitation and the fact that there is no specification for Java currently, it is kept as a future work.

Figure 20 gives the geometry structure defined in OpenGIS specifications. In this structure, all geospatial elements are derived from an abstract root type: Geometry. The type Curve, Surface, GeometryCollection, MultiSurface, and MlitCurve are also abstract type only for the purpose of structure extension. The concrete types are Point, LineString (multi-point line), LinearRing (closed multi-point line), Line (two-point line), Polygon, MultiPoint, MultiLineString, and MultiPolygon. The type SpatialReferenceSystem is attached to the geometry so that each geometry objects may have different spatial reference systems.

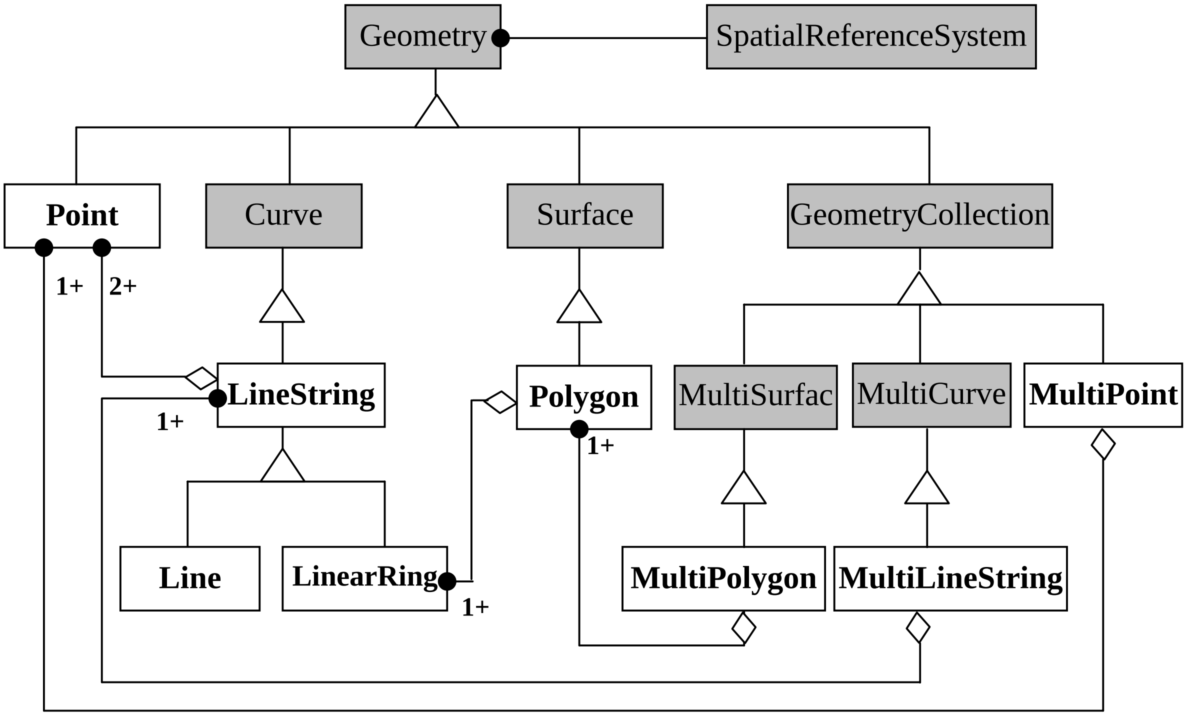


Figure 20: Geometry Hierarchy in OpenGIS Specification (Source: OpenGIS 1999b)

(See Appendix A for data Model Notation)

Figure 21 is the geometry structure designed for the prototype system of the distributed geoprocessing service model. In this structure, abstract types only for further system extension are not defined, such as Curve, Surface in OpenGIS simple feature specification. However, the concrete types such as Polygon, Polyline and Point are defined in the same manner with the specification. Spatial reference system is not applied to each individual geometry object, but applied to the Map that contain them. Most types defined in the structure have their correspondences in the OpenGIS simple feature specification, except two of them: GeoPolyParts and GeoRectangle. The GeoRectangle object is designed to model the bounding box of geometry objects and that of Map and Layers. The GeoPolyparts object is used to model the sub-element of a GeoPolygon or GeoPolyline. The role it plays is similar to the LinearRing in Polygon in the OpenGIS implementation specification.

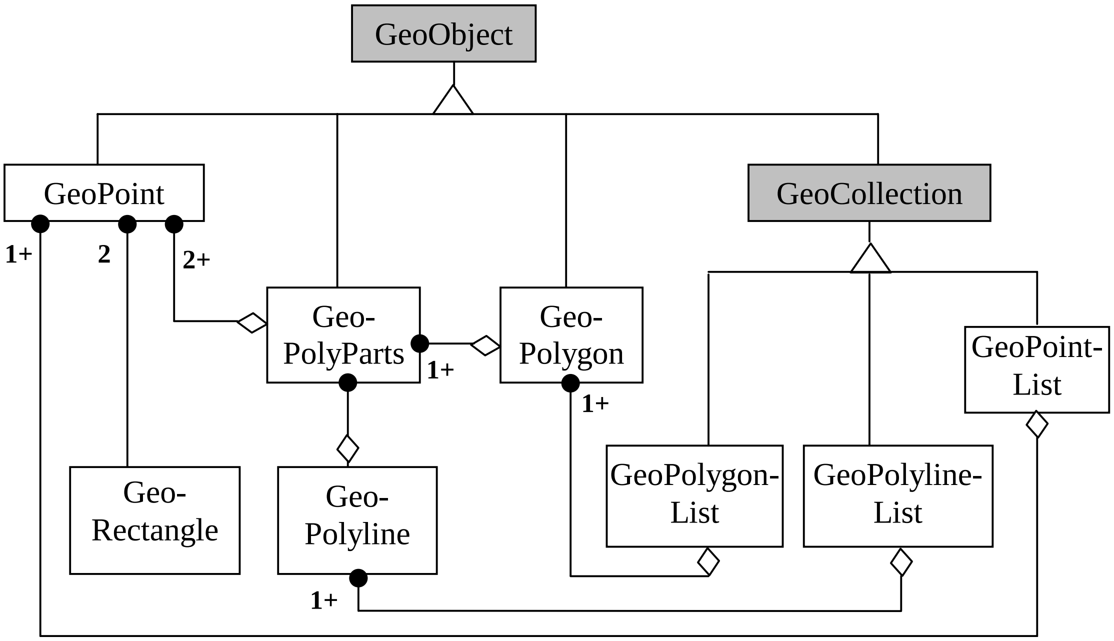


Figure 21: Geometry Objects in the Prototype System

(See Appendix A for data Model Notation)

3.2.4 Geospatial Reference System Objects

An important feature of GIS is that the data in GIS should be spatially referenced in a spatial coordinate system. A GIS object is the abstraction of a real world feature or phenomenon, which contains the location information and the corresponding attributes. The location information, technically the spatial data, is defined in coordinate pairs in a spatial coordinate system, in which the object is associated with a location on the Earth. Using the concept of reference systems, spatial object primitives such as geographically registered geometry are tied to the real world. Spatial objects are then decorated in a controlled manner with attributes.

Defining spatial reference system objects involves many geographical concepts that describe a data set. These concepts include coordinate system, datum, unit, projection, etc. Fortunately, the spatial reference system model for the prototype system was later developed after the development of geometry model described above, and by that time the OpenGIS simple feature implementation specification for OLE/COM had come out. It is good idea to adopt the spatial reference system model defined in the specification. Figure 22 illustrates the spatial reference objects and their relationships.

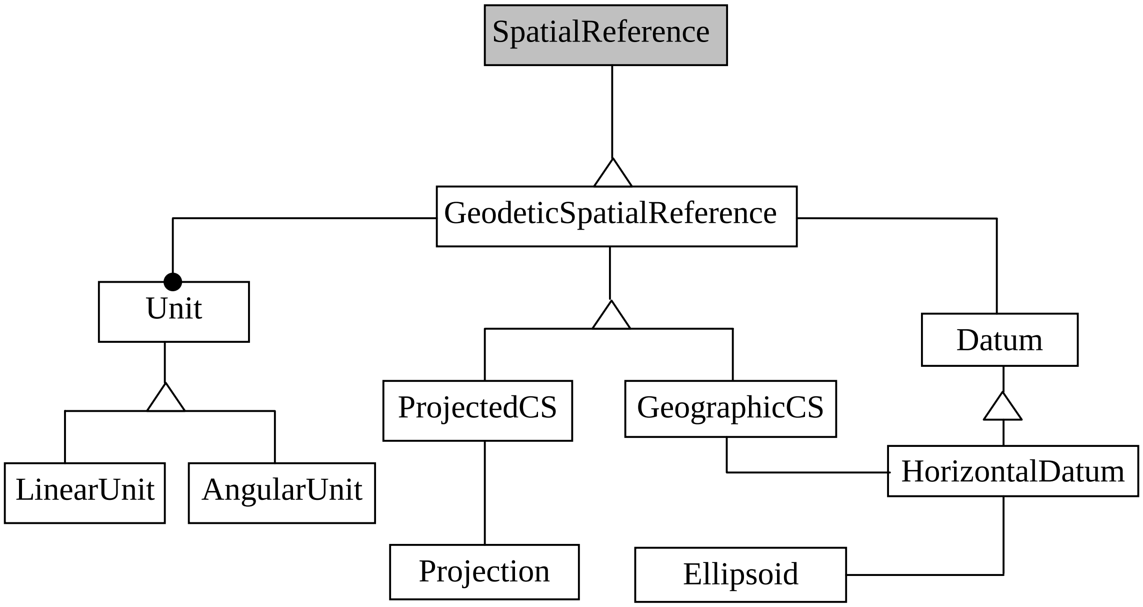


Figure 22: Spatial Reference System Object Model (Source: OpenGIS 1999b)

(See Appendix A for data Model Notation)

In this model, only two-dimension horizontal reference systems are defined. The SpatialReference is an abstract type of any spatial reference system at the top level. From GeodeticSpatialRefence type, it is the reference systems we used in GIS systems. All geodetic reference systems have Unit and datum properties, and it can be classified into two types: the projected coordinate system (ProjectedCS) and the geographical coordinate system (GeographicCS). The ProjectedCS object models the coordinate systems that have been projected to the plane, such as UTM coordinate system. The GeographicCS object models the global coordinate systems such as latitude and longitude. According to the data sets that come from different sources, different coordinate systems can be applied to different data sets and the transformation among different coordinate systems can be performed using the objects defined in the reference system object model.

3.3 Distributed Geoprocessing Component Registration Model

In the distributed geoprocessing model architecture illustrated in section 3.1.3, Figure 17, the distributed component registration model is the key part for supporting distributed object computing. In the following sections, the distributed component registration model will be discussed in more detail.

3.3.1 Distributed Geoprocessing Scenarios

To clearly describe the model, three typical scenarios are used to illustrate how the distributed geoprocessing components serve for clients.

I. Virtual Geoprocessing Library

A geospatial information provider has a geodata catalog service over the Internet that provides geospatial information to users with diverse geodata distributed all over the world. The service worked successfully for years until recently a lot of clients complained that they did not have geoprocessing tools to process the geodata to meet their specific requirements. They cannot afford the full license of heavy-weighted large GIS systems. The inability to offer geoprocessing tools affects the use of the geodata, which in turn affects the catalog service. In this situation, the geospatial information provider decides to enhance the system by building a virtual geoprocessing library, which got strong support from his technical committee.

A few months later, the virtual geoprocessing library project finished. Similar to the geodata catalog service in principle, virtual geoprocessing library manages the meta-information of geoprocessing components that distributed in the Internet. The meta-information is registered into the library by the component provider through the web pages. The library is virtual because it does not have to contain the components physically, but instead a catalog of these components.

A convenient user interface displaying geodata and geoprocessing components catalog is provided to the users through the Internet. Program common classes and interfaces are developed and available to any third party component developers. At first stage, the library builder developed some frequently used components to serve the users, and received very good responses. Later many third party developers joined the library to develop the distributed geoprocessing components, with more and more geospatial users joining the library to "rent" both geodata and geoprocessing tools. The library then becomes another successful story of that geospatial information provider.

II. Non-regular Geospatial Information User

A small consulting company recently was awarded a contract that involves the use of some geodata sets. However, the data sets are in different coordinate systems, so they must have these data sets transformed to the required coordinate system before the data sets become useful for their project. Unfortunately, the company does not have the tools to do this simple processing. There are two options for this company: purchase a full GIS system but in which only one command will be useful to them, or contract out to a GIS professional company to do this job for them. Both options are not economical which is not acceptable to this small company.

Occasionally, a technical person found that there is a Virtual Geoprocessing Library that "lends" geoprocessing tools through the Internet. He accessed the library web site, a list of available geoprocessing tools were listed in categories for his choice. He searched in the Spatial Reference category, read the meta-information about the tools, and finally found that a reference transformation component right meets their requirements. After choosing this tool, a GIS interface with a reference system transformation button popped up and asked them to load the his data set. After loaded the data, he pushed the transform button. The data set was transformed and saved to the local disk. The project manager was pleased because they found a way to get the job done economically (See Figure 23).

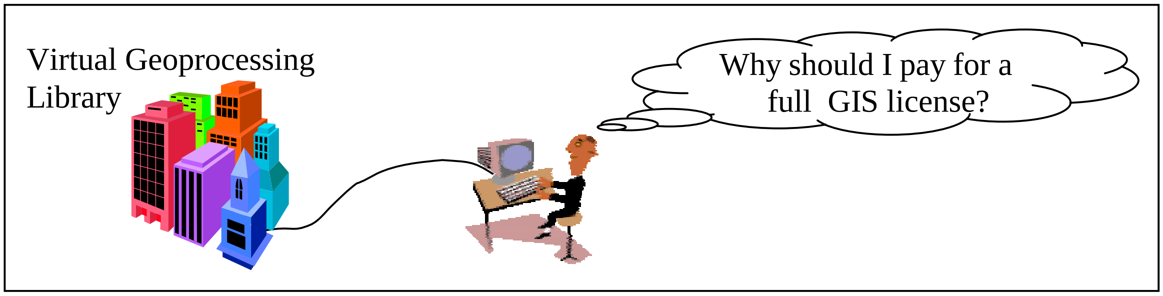

Virtual Geoprocessing Library
Why should I pay for a full GIS license?

Figure 23: Scenario: A Distributed Geoprocessing Service User

III. Geoprocessing Component Developer

Mr. XYZ has been a successful GIS consultant for many years. In these years, he developed many geoprocessing tools. Knowing that there is a virtual geoprocessing library that allows third party developer to "rend" geoprocessing components to users, Mr. XYZ decided to take the chance.

He first downloaded the program interfaces and common classes from the library, using them developed a wrapper for his existing geoprocessing components. Then he run these tools on his own machine that had an Internet connection, and registered these components into the library through the web page by providing the information about the connection and the components. After that he noticed that a lot of people used his software and he kept receiving money from his renting. Mr. XYZ is very happy for helping people and making money from his existing software. He decides to develop more convenient geoprocessing tools to serve people in the Internet.

3.3.2 Distributed Geoprocessing Registration Model

The distributed geoprocessing component registration model is the core of the whole service architecture illustrated in Figure 17, section 3.1.3 that makes "Geoprocessing anywhere" possible. A geoprocessing component registration server is introduced to manage the meta-information of each component. Together with an arbitrary client, a geoprocessing server, and the web server, a pyramid relationship between client and servers is formed, shown in Figure 24.

In this model, taking the Virtual Geoprocessing Library as an example, the web server and the geoprocessing registration server are located in the "library" and bounded together. External geoprocessing components can be connected to the library by registering the meta-information of the components into the library categories – the component meta-information database – before the components are accessed. Any client can look up the component meta-information database, read component descriptions, choose, connect and use specific components.

The client and the geoprocessing server can be many in the component registration model. Each individual client and a geoprocessing server can form a pyramid with the web server and the registration server, as illustrated in Figure 24.

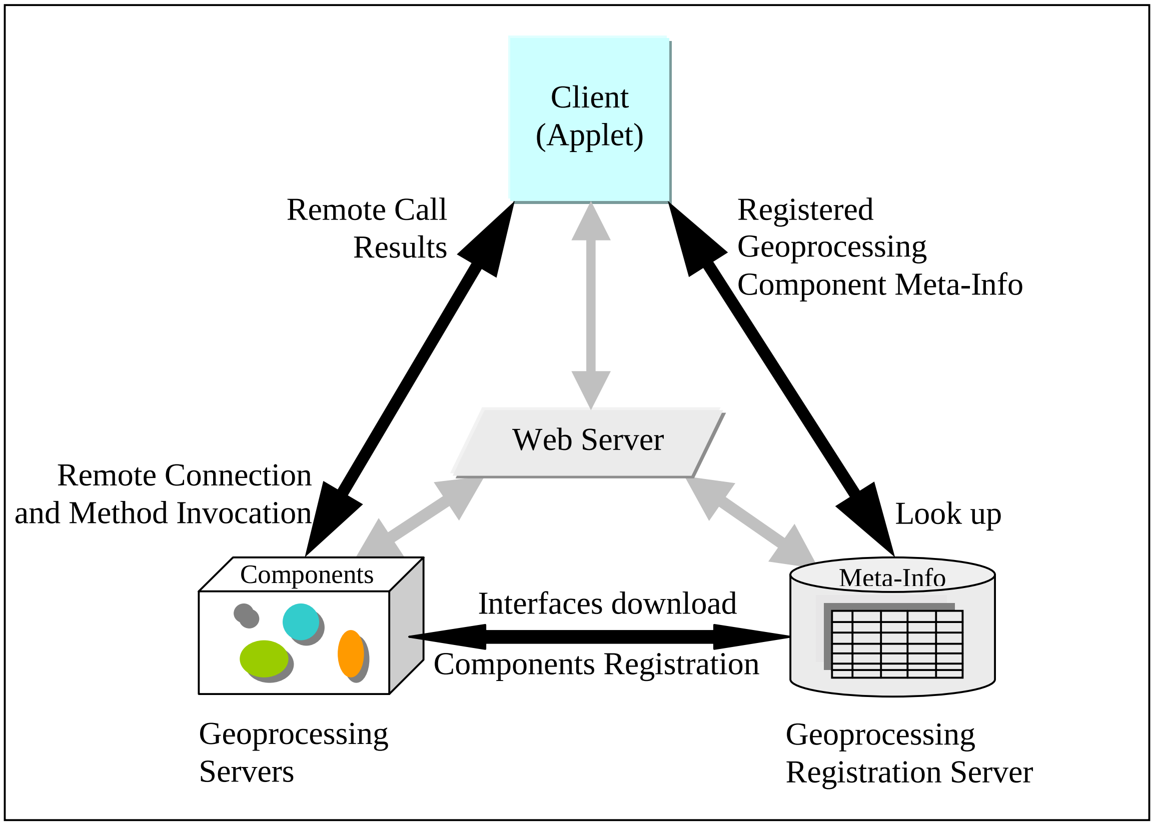


Figure 24: Relationships of Client and Servers in Geoprocessing Registration Model

In the Java solution of the model, the client program is an extendable applet, which may bind components to the user interfaces of the applet. Different object communication methods may be used between the applet, geoprocessing components, and registration server, as shown in Figure 25.

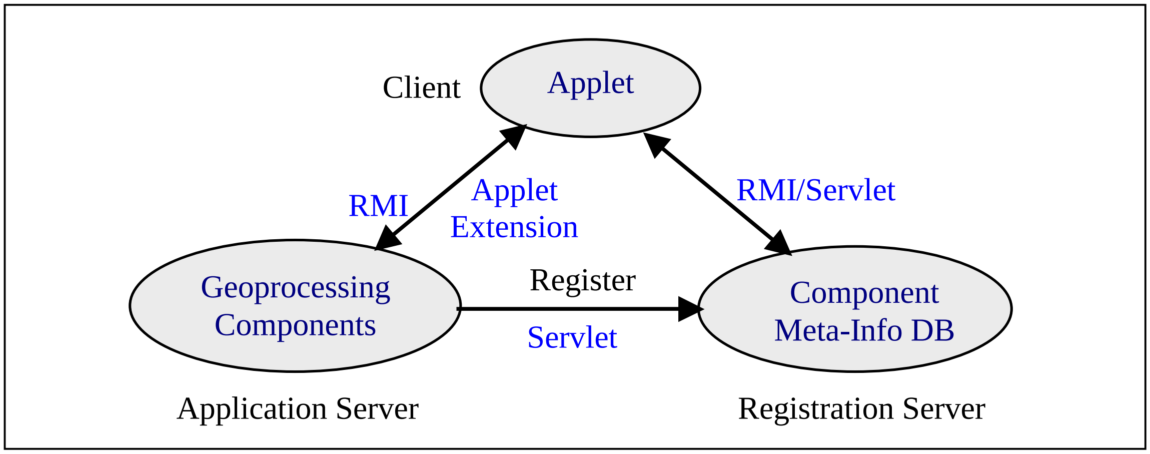


Figure 25: Java Object Communication Solutions in Geoprocessing Model

The web server is not demonstrated in Figure 25 because it is not critical in the component registration model. However, many processes will involve the web server, such as class downloading to the client, HTML registration form for the application server, etc. Components can be registered into component meta-information database by Servlet methods, which can accept and process the inputs from HTML forms and place the information into the database. The client applet may get the meta-information records from the registration server by Java Servlet. The meta-information should contain component connection information so the component can be connected to the applet via RMI to extend its functionality. Component interface objects should be uploaded to the web server in the process of component registration so that they can be downloaded as a kind of applet extension to the client and used in the remote methods looking-up process.

Having the above solutions for the component registration model, the next step is to decide the contents of the component meta-information.

3.3.3 Geoprocessing Component Meta-information

Geoprocessing component meta-information is a new concept that deserves a lot of research efforts. Similar to geodata catalog services, in which geospatial metadata plays a core role, geoprocessing component meta-information will play a key role in geoprocessing services. Geospatial metadata contains much information about the geodata, including identification information, data quality information, spatial data organization information, spatial reference information, entity and attribute information, distribution information, metadata reference information, contact information, etc. In other words, it provides users with information to share the geodata. There are several geospatial metadata standards that are in use, such as the Content Standard for Digital Geospatial Metadata (FGDC 1998). One could say that it was geospatial metadata that made the sharing of geodata possible. Similarly, the geoprocessing component meta-information will be inevitable in a geoprocessing service model. The following study on this topic is very preliminary and the author hope that this preliminary study can provide a starting point for the further research and bring the distributed geoprocessing service to a higher lever.

Meta-information of a geoprocessing component should contain information about:

1. how to identify the software component,
2. how to find and connect to the software component,

3. what the component can do,
4. how the component works in principle,
5. how to use the component,
6. what is the quality of the component, and
7. whether other components are referenced by the component, etc.

The first item is inevitable from software engineering perspective, and the others are necessary to help people recognize and use the component.

According to the above principles, we can define the following component meta-information items, which not all of them are used in the prototype system.

Note: The following definition is very preliminary and maybe partial.

Symbol meanings: Items included in "( )" are optional in some cases.
Elements included in "[ ]" are data types for the items.
"m{item }n" means that "item" iterates from "m" to "n".
Type "Complex" means it consists of more items.

```
Componen_meta-information  = Identification_Info [Complex] +
                             Software_connection_Info [Complex] +
                             Fuctionality_Info [Complex] +
```

Principle_Info [Complex] +

Instruction_Info [Complex] +

Quality_Info [Complex] +

0{Component_Reference_Info [Complex] }n +

(Contact_ Info [Complex])

In which:

Identification_Info = Component_Name [String] +

Component_Version [String]+

Author_Info [Complex]

Author_Info = Author_Name [String] + (Password [String])

Software_connection_Info = Component_URL [String]+

Port_Number [Interger]+

Component_Type* [String] +

Interface_Class_Name** [String] +

(Database_Info [Complex])

Database_Info = Database_URL [String] +

Database_UserId [String] +

Database_PassWord [String]

Component_Type : Type of the component in software engineering point of view, such as Applet_Extension, RMI_Component, etc.*

*Interface_Class_Name*** *: Name of component wrapper class or interface at top level to link to the Applet.*

Fuctionality_Info = Component_Functionality_Category* [String] +
Component_Functionality_Description [String]

*Component_Functionality_Category** *: Component classification from application point of view. Such as Data Format Conversion, Geospatial Reference Conformation, Spatial Analysis, Modelling, etc.*

Principle_Info = Algorithms_Description [String] +
Component_Geographic_Location [String] +
Communication_Approach* [String]

*Communication_Approach** *: Socket, Servlet, RMI, etc.*

Instruction_Info = Requirement_Description [String] +
Input_Description [String] +
Output_Description [String] +
Operation_Description [String]

Quality_Info = Performance_Description [String] +
Processing_Accuracy_Description [String] +

Component_Stability [String]

Component_Reference_Info = Identification_Info [Complex] +
Quality_Info [Complex]

Contact_Info = Contact_Person_Name [String] +
Address [String] +
Post_Code [String] +
Phone [String] +
Email_Address [String].

CHAPTER 4 MODEL PROTOTYPE IMPLEMENTATION

4.1 GeoEye™ – Implementation of Foundation Classes and Client Interfaces

A prototype service system, which was implemented to test and demonstrate the model designed in Chapter Three, was started from the foundation classes and the applet – an client-side user interface that can run in an Internet browser. A client side program, which was trademarked by the University of Calgary in the name of GeoEye™, was developed at this stage (GeoEye™ has been licensed by the University Technologies International Inc.). Since the application in Java technologies is still a new phenomenon in GIS software development, the whole system was entirely developed from ground up. The implementation of GeoEye™ is described in the following sections.

4.1.1 Foundation Classes

Foundation classes define system common objects described in section 3.1.2 (part d, Figure 16) and section 3.2 (Figure 19). The system common objects include basic GIS geometry, data structure objects and basic data access objects, which define an internal data model that provides clients and servers with a basic understanding of geodata in the system. In the process of client/server communication, common objects can be used as parameters and results from remote geoprocessing.

Foundation classes implemented include gismodel package that contains classes such as MapBean (a Java Bean Component that models the concept of Map), Layer, etc., gismodel.base package that contains geometry definitions, gidmodel.layers.shape package that defines classes to access ESRI Shape File, etc.

Manipulation Methods and Events that define the behaviors of the common objects are defined in these foundation classes, such as map Display Controls (Paint, Zoom, Pan, Overview, Etc.), Layer Controls (Add, Remove, Rename, ChangeProperties, Etc.), Object Operations (Select, Identify, AttributeTable, Etc.), and Spatial Relationship Functions (PointInPolygon, PointOnLine, LineInPolygon, etc.).

In the implementation of GeoEye™, ESRI Shape file was chosen to be the first geodata format to support. Like most other geodata formats, it use the geometry-centric geo-relational data model to organize the data, in which spatial objects – the geometry – are stored in file based system while their associate attribute data are maintained in a relational database. Data integrity had been the major concern of the data model implementation due to the separated management and organization of spatial data elements and attribute data elements in geodata sets.

A clear trend in GIS software development is to develop an object-based data model, in which a geographic object is defined by geometry features, attributes as well as its behaviors. Figure 26 shows an integration of geometry objects and their corresponding attribute records, which had been implemented in the prototype system (Yuan and Tao, 2000). In the integration, the attributes of a spatial object that are stored in an external relational database are bounded to a GeoRecord object that aggregates into a collection object, GeoRecordSet, and is attached to the Layer object directly.

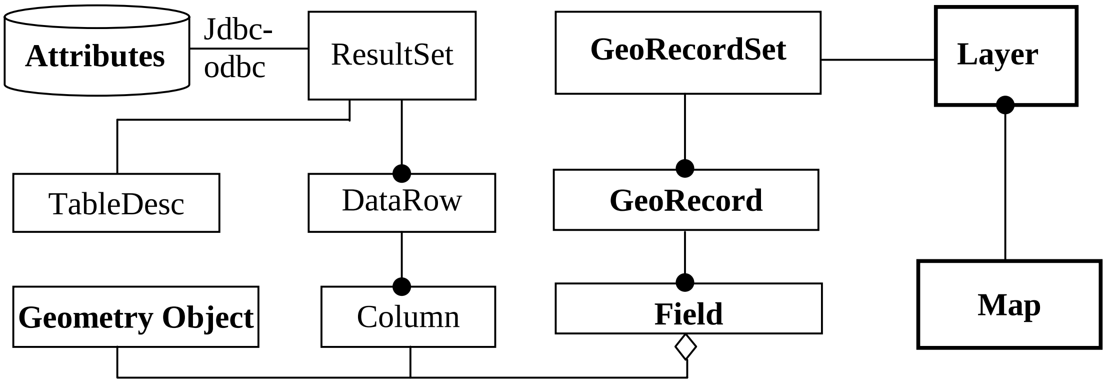


Figure 26: The Integration of Geospatial Data and Their Attributes

(See Appendix A for data Model Notation)

The attributes in a database are accessed by JDBC/ODBC provided by Java language, the query result forms a ResultSet in which attribute records are extracted from database. A DataRow object stores one record of attributes that is matched to a geometry object by the unique Id. A Field object is used to model both attribute Column of a DataRow and the geometry object. That is, a Field object may be a geometry object, or one of its attributes. The GeoRecord integrates the geometry Field and many attribute Fields into one object that is ready for manipulations.

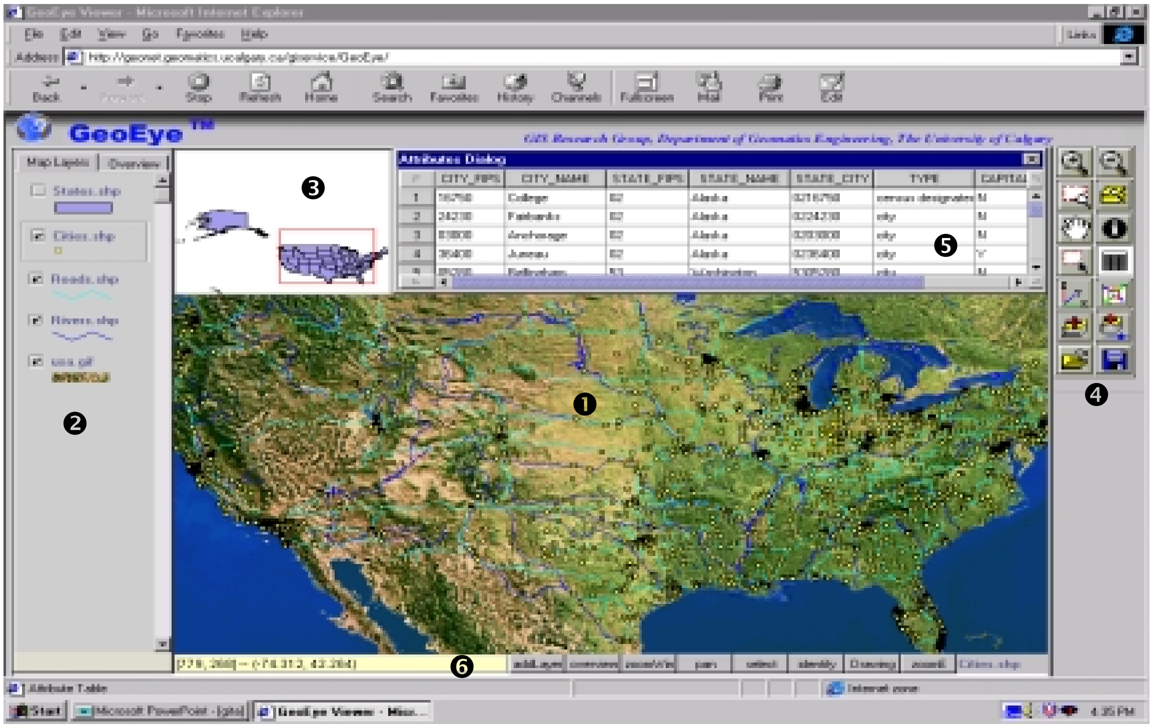
4.1.2 The Applet and Client Interfaces

The client program is sent to users via the GeoEye™ applet, which includes client user interfaces and the foundation classes. Functions implemented in it are listed in Table 3:

Table 3: GeoEye™ Functions

Geodata Access	– Local and Remote (Web Server) Shape File access – Access both spatial and non-spatial data
Image layer support	– Support GIF and JPEG images – Georeference image as map background
Map display control	– Repaint, Zoom (In/Out/Window/Extent), and Pan.
Map overview window	– Overview window On/Off, Resize, and Relocate – Layer On/Off in overview window – Display and Drag current map display position, etc.
Layer control	– Layer On/Off, Add/Remove Layers – Layer Reorder, Rename, Change Layer Color, etc.
Attributes manipulation	– Identify, Select by Point/Box, Attribute Table, etc.
Drawings	– Draw and Save Point, Polyline, and Polygon object.
Project Management	– Save and Reload project file.

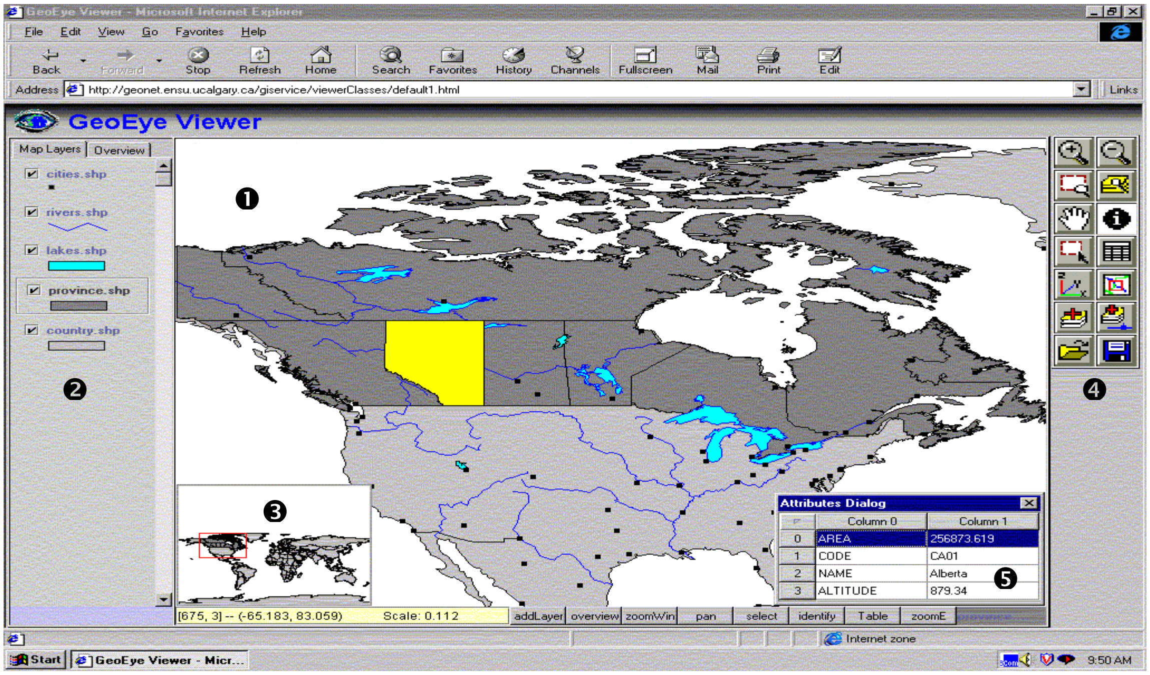
Figure 27 and 28 are the screen shots of GeoEye™ Applet.


GeoEye Viewer - Microsoft Internet Explorer
File Edit View Go Favorites Help
Address http://geonet.geomatics.ucalgary.ca/giservice/GeoEye/
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print Edit
GeoEye TM
GIS Research Group, Department of Geomatics Engineering, The University of Calgary
Map Layers Overview
States.shp
Cities.shp
Roads.shp
Rivers.shp
usa.gif
❶
❷
❸
❹
❺
❻
[779, 260] -- (-78.312, 43.264)
addLayer overview zoomWin pan select identify Drawing zoomE Cities.shp
Attribute Table
Internet zone
Start Microsoft PowerPoint - [gis] GeoEye Viewer - Micr...
4:35 PM

Attributes Dialog

#	CITY_FIPS	CITY_NAME	STATE_FIPS	STATE_NAME	STATE_CITY	TYPE	CAPITAL
1	16750	College	02	Alaska	0216750	census designated	N
2	24230	Fairbanks	02	Alaska	0224230	city	N
3	03000	Anchorage	02	Alaska	0203000	city	N
4	36400	Juneau	02	Alaska	0236400	city	Y
5	05280	Bellingham	53	Washington	5305280	city	N

Figure 27: GeoEye™ Applet User Interfaces with Image and Shape Layer


GeoEye Viewer - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print Edit
Address http://geonet.ensu.ucalgary.ca/giservice/viewerClasses/default1.html
Links
GeoEye Viewer
Map Layers Overview
cities.shp
rivers.shp
lakes.shp
province.shp
country.shp
❶
❷
❸
❹
❺
[675, 3] -- (-65.183, 83.059) Scale: 0.112
addLayer overview zoomWin pan select identify Table zoomE
Internet zone
Start GeoEye Viewer - Micr...
9:50 AM

Attributes Dialog

#	Column 0	Column 1
0	AREA	256873.619
1	CODE	CA01
2	NAME	Alberta
3	ALTITUDE	879.34

Figure 28: GeoEye™ Applet User Interfaces with Object Identifying

In Figure 27 and 28:

➊: Map display area *➋: Layer control area* *➌: Map overview window*

➍: Map Operation Buttons *➎: Attribute display window for Identify*

➏: Status bar that displays mouse coordinates, scale, and the current layer.

4.2 GeoServnet – Implementation of Servers

GeoServnet, as its name implied, will be the server implementation of the proposed distributed geoprocessing model. Servers in GeoServnet include a GIS web server, a component registration server and distributed geoprocessing servers. In this section, the implementation of web server program and the component registration server will be depicted. The implementation of distributed geoprocessing servers will be discussed in next section.

4.2.1 Web Server Program

The GIS web server is in the center of the model. It contains basic applet classes, remote component interfaces and applet extensions of geoprocessing servers. It physically exists in the same machine with the component registration server. Geodata resources in the web server should be accessible to the GeoEye™ clients. A geodata access service example – the access of ESRI Shape data – was developed in the web server. Java

Servlet are used in the object communication between the GeoEye™ Applet and the Servlet. Objects are passed through the Internet.

Figure 29 illustrates the object communication processes when the client requests remote data from the web server. RequestObjects is used to pass the request commands to the web server, and different results can return to the client according to the different requests. The first two requests happen when users looking for and access the data in the remote data access interface (see Figure 30) and the third request happens when the user operates the attributes of a layer, such as identify or display attribute table.

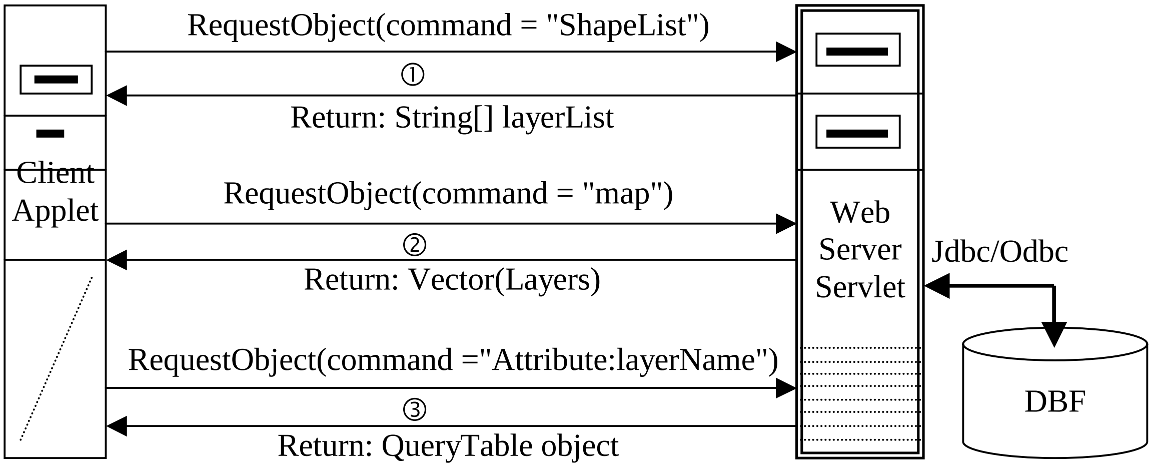

RequestObject(command = "ShapeList")
①
Return: String[] layerList
Client Applet
RequestObject(command = "map")
②
Return: Vector(Layers)
RequestObject(command ="Attribute:layerName")
③
Return: QueryTable object
Web Server Servlet
Jdbc/Odbc
DBF

Figure 29: Servlet Object Communication between Client and Web Server

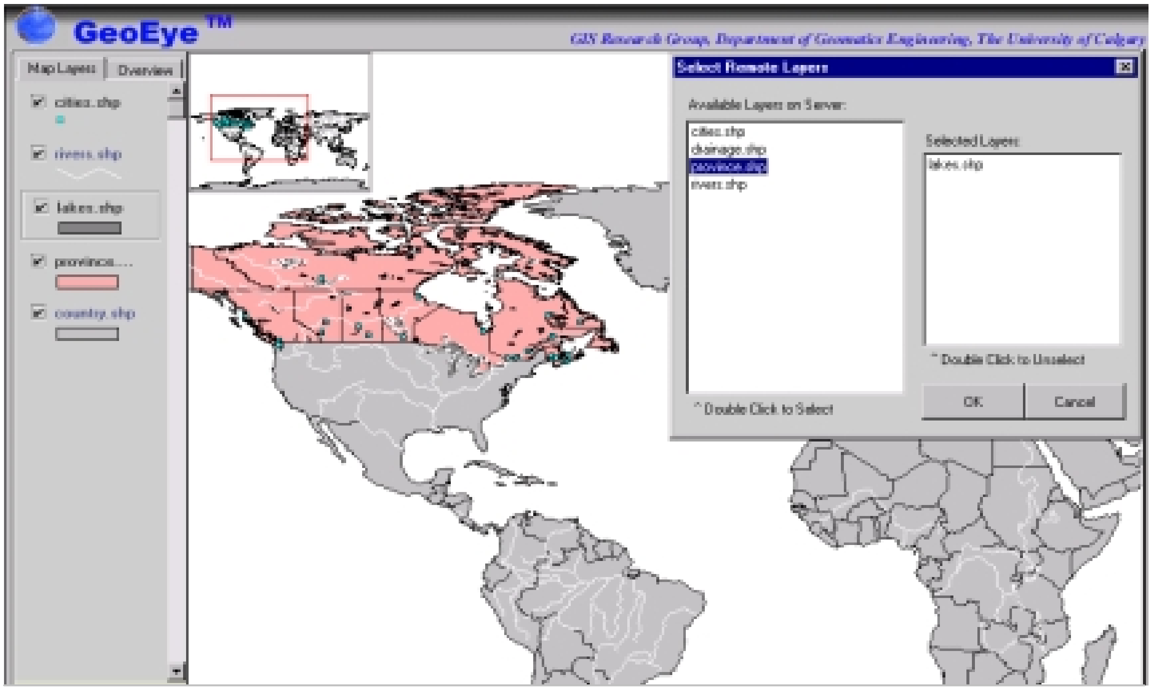


Figure 30: Remote Shape Data Access Interface

The Servlet program running on the web server is responsible for the shape data cataloging, shape data access and Jdbc-Odbc database connection. It responses to user's requests by communicating with applet directly.

Other responsibilities of web server in the model include providing web pages to component developers for downloading Foundation Classes and uploading component interface classes, which will not be addressed in this thesis.

4.2.2 Component Registration Server

Geoprocessing component registration is implemented through a FORM embedded in a standard HTML file that links with a Java Servlet running in the web server. The FORM collects and sends component information to the web server. The servlet in the server puts the information into the component meta-information database. Figure 31 illustrates this principle. Figure 32 illustrates the component registration FORM.

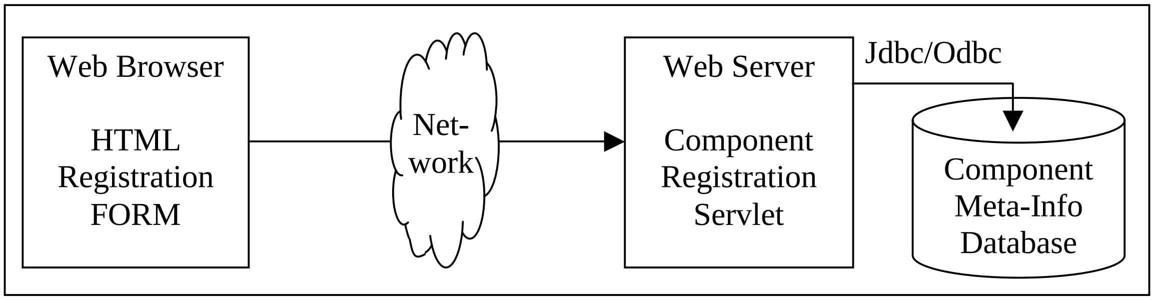
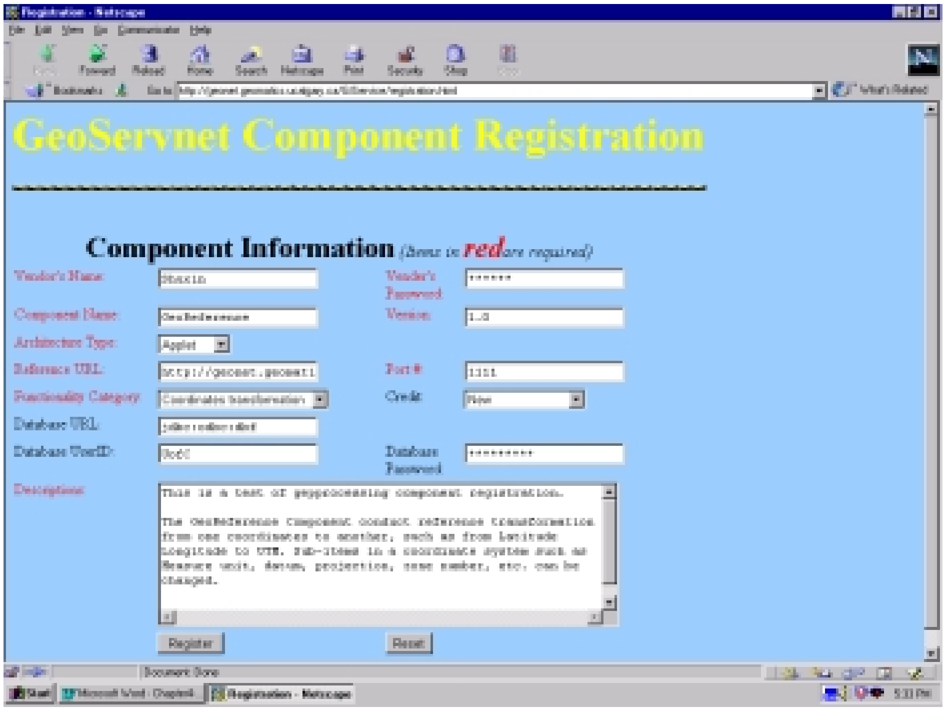


Figure 31: Geoprocessing Component Registration Process


Registration - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Go to: What's Related
Register
Reset
Document: Done
Start
Microsoft Word - Chapter6
Registration - Netscape

GeoServnet Component Registration

Component Information *(Items in* ***red*** *are required)*

Vendor's Name:	Shaxin	Vendor's Password:	******
Component Name:	GeoReference	Version:	1.0
Architecture Type:	Applet		
Reference URL:	http://geonet.geomati	Port #:	1111
Functionality Category:	Coordinates transformation	Credit:	Free
Database URL:	jdbc:odbc:dbf		
Database UserID:	Bob?	Database Password:	*********

Description:

This is a test of geoprocessing component registration.

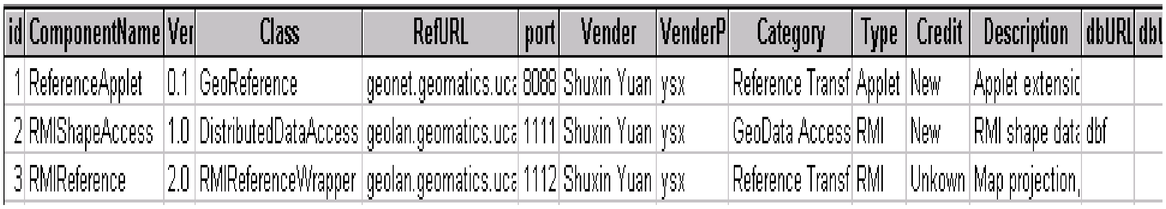
The GeoReference Component conduct reference transformation from one coordinates to another, such as from Latitude Longitude to UTM. Sub-items in a coordinate system such as Measure unit, datum, projection, zone number, etc. can be changed.

Figure 32: Component Registration Form in a Web Browser

The component meta-information in the above registration FORM can be extended to contain more information to help users use the component. The current FORM is only an example and focuses on software engineering aspect.

When a component is registered successfully, its meta-information is saved to a meta-information database. Microsoft Access database is used for this purpose in the implementation. Table 4 is a part of the main table in the meta-information database.

Table 4: Example of Component Meta-Information in the Database



id	ComponentName	Ver	Class	RefURL	port	Vender	VenderP	Category	Type	Credit	Description	dbURL	dbl
1	ReferenceApplet	0.1	GeoReference	geonet.geomatics.uca	8088	Shuxin Yuan	ysx	Reference Transf	Applet	New	Applet extensic		
2	RMIShapeAccess	1.0	DistributedDataAccess	geolan.geomatics.uca	1111	Shuxin Yuan	ysx	GeoData Access	RMI	New	RMI shape data	dbf	
3	RMIReference	2.0	RMIReferenceWrapper	geolan.geomatics.uca	1112	Shuxin Yuan	ysx	Reference Transf	RMI	Unkown	Map projection,		

4.2.3 Component Wrapper

With all the meta information about the component, the next step to make the component accessible to clients is to connect the component with the client user interface – the GeoEye™ Applet. Since components may be developed independently without informing the client program, the GeoEye™ Applet may not have the detail information of the components, such as class names, function names, parameters, return types, etc. However, the concept of component wrapper with specific software interfaces to the outside may be used to solve this problem. These software interfaces, which can make an agreement between two parties, the component and the client program, should be the only way that makes the communication possible between the component and client program. Figure 33 demonstrates this idea.

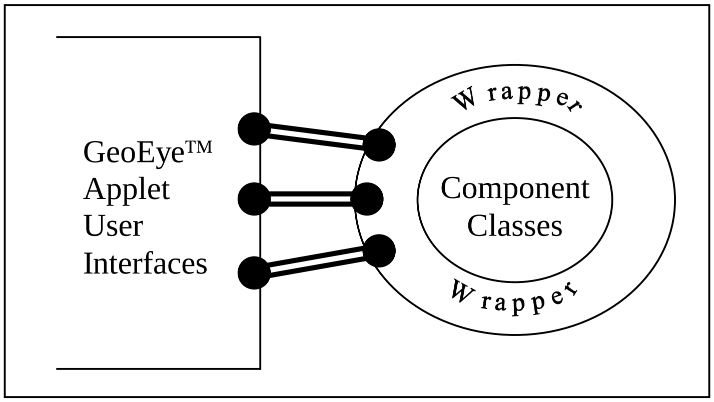


Figure 33: Component Wrapper

The component wrapper actually is a kind of class interface between the client and the server. The implementation of the component wrapper should comply with certain predefined rules. In the current model implementation, the component wrapper interfaces predefined an abstract class AbstractComponent that only has an abstract function *void play(void)* in it. A concrete component wrapper implementation just need to implement a class that derives from AbstractComponent (AbstractComponent class should not be changed) and implement the specific function *void play(void)* to start the component. For example, in the implementation of Geo-reference component wrapper, the following codes can be used to connect to the component:

```
public class GeoReference extends AbstractComponent {  // component wrapper class
    public GeoReference() { }  // A public null class construction
    public void play(){  //each component wrapper class should implement this method
        ReferenceControl refDlg = new ReferenceControl ( getFrame(), getMap(),
        "Spatial Reference System");  // connection to component user interface
         if(!refDlg.isShowing()) refDlg.show();
        else refDlg.hide();
    }  // end of play() method
} // end of class definition
```

This specific wrapper class connects the wrapper with the existing component user interface class – the *ReferenceControl* class, which in turn makes the component visible to the clients since the client applet knows that each component have an interface function named *play(void)*. Also the wrapper class name must be registered into the component meta-information database so that it can be available in the client program.

In the client applet, the wrapper is used to access the component. As an example, the following codes in the applet can be used to activate the component:

```
 ... ...  // get the wrapper name from the meta-information database
Class theClass = Calss.forName(wrapperClassName); // get the wrapper class
Object theObject = theClass.newInstance();              // get the wrapper object
if(theObject instanceof AbstractComponent) {
        // use the AbstractComponent instead of concrete component
        AbstractComponent ext = (AbstractComponent)theObject;
        ext.setFrame(this);    // this is predefined in AbstractComponent
        ext.setMap(map);   // this is predefined in AbstractComponent
        ext.play();         // The function that activates the component
}
... ...  // other processes
```

Two types of geoprocessing components can be implemented in the current model –

Applet extensions and RMI remote components. For the applet extensions, all of component classes including the component wrapper classes should be uploaded to the web server during the component registration process and downloaded to clients as a part of the applet at run time, when a client requested the specific component. For the RMI remote components, an applet extension also need to be uploaded to the web server, however, the extension may only include the component wrapper classes and remote invocation interfaces, such as remote object stubs. The component itself is still located in the geoprocessing server. The client can connect to the remote component via the remote interfaces in the wrapper and communicate with it by using Remote Method Invocation (RMI). Figure 34 illustrates the difference between these two types of components:

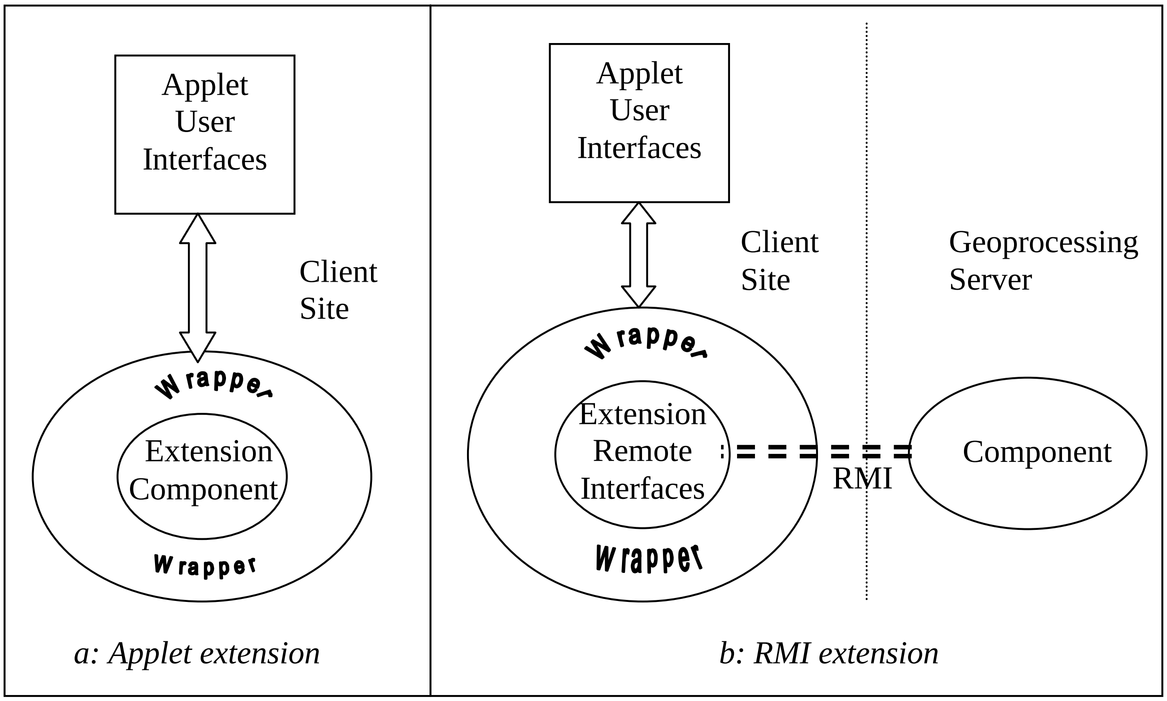
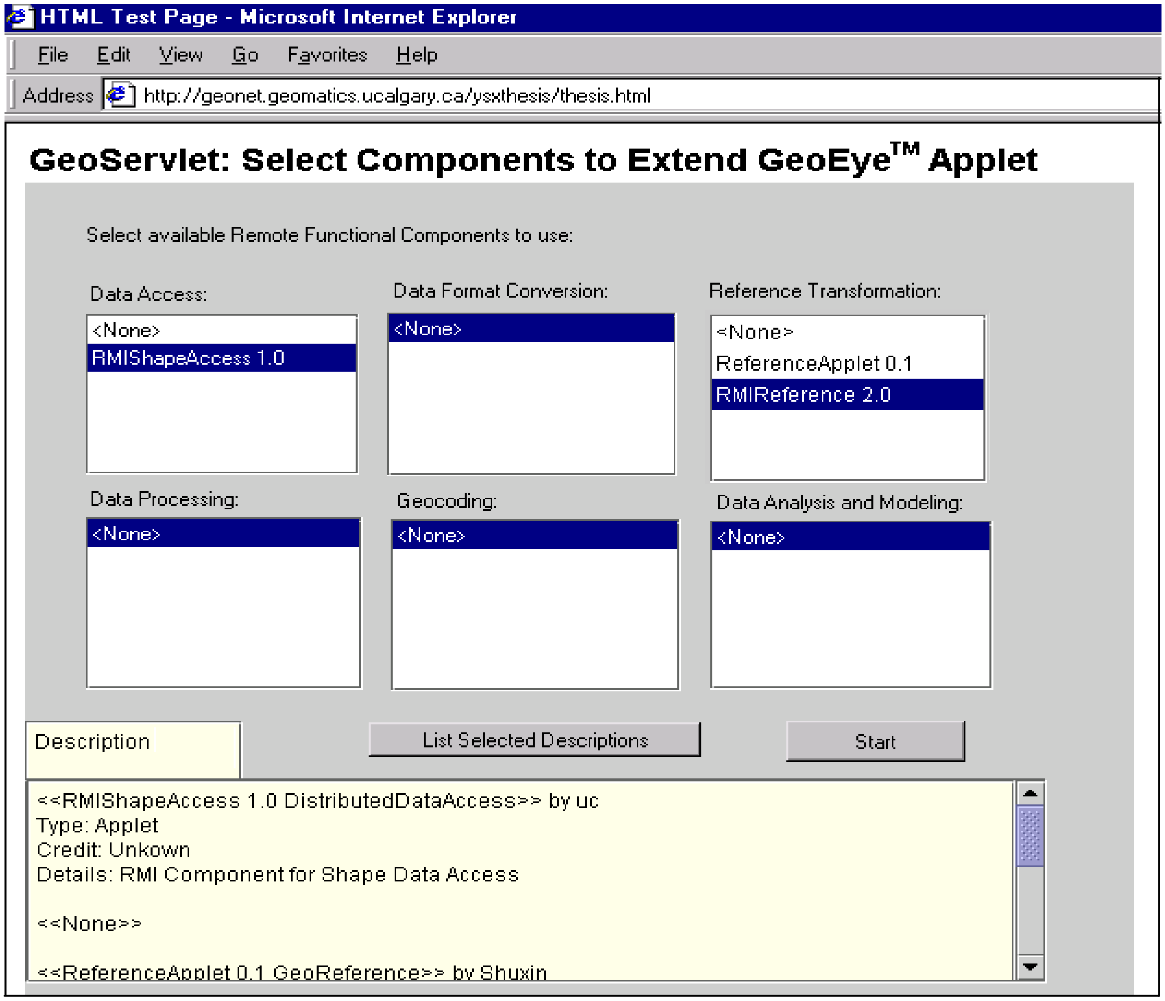


Figure 34: The Difference of Two Types of Components

4.2.4 Component User Interfaces

An important responsibility of the client program is to get the component meta-information from the database in the component registration server, and provide user with convenient operation interface to select components that right meet their needs. Figure 35 is an example implementation of the interface for user's choice.


HTML Test Page - Microsoft Internet Explorer
File
Edit
View
Go
Favorites
Help
Address
Description
List Selected Descriptions
Start

http://geonet.geomatics.ucalgary.ca/ysxthesis/thesis.html

GeoServlet: Select Components to Extend GeoEye^{TM} Applet

Select available Remote Functional Components to use:

Data Access:
- <None>
- RMIShapeAccess 1.0

Data Format Conversion:
- <None>

Reference Transformation:
- <None>
- ReferenceApplet 0.1
- RMIReference 2.0

Data Processing:
- <None>

Geocoding:
- <None>

Data Analysis and Modeling:
- <None>

```
<<RMIShapeAccess 1.0 DistributedDataAccess>> by uc
Type: Applet
Credit: Unkown
Details: RMI Component for Shape Data Access

<<None>>

<<ReferenceApplet 0.1 GeoReference>> by Shuxin
```

Figure 35: Component Selection User Interface

In the interface, available components are classified into several groups according to their functionality. Component meta-information is transformed into component descriptions to help the users make their decisions.

After users have made their choice, the components are connected to the GeoEye™ Applet interface by adding additional buttons for the extension, see Figure 36. When the user push the extension buttons, the extension classes (wrapper only or the component itself) will be downloaded to the client and running on the client machine as an extended part of the original applet. As shown in Figure 37, a component interface for remote data access displayed.

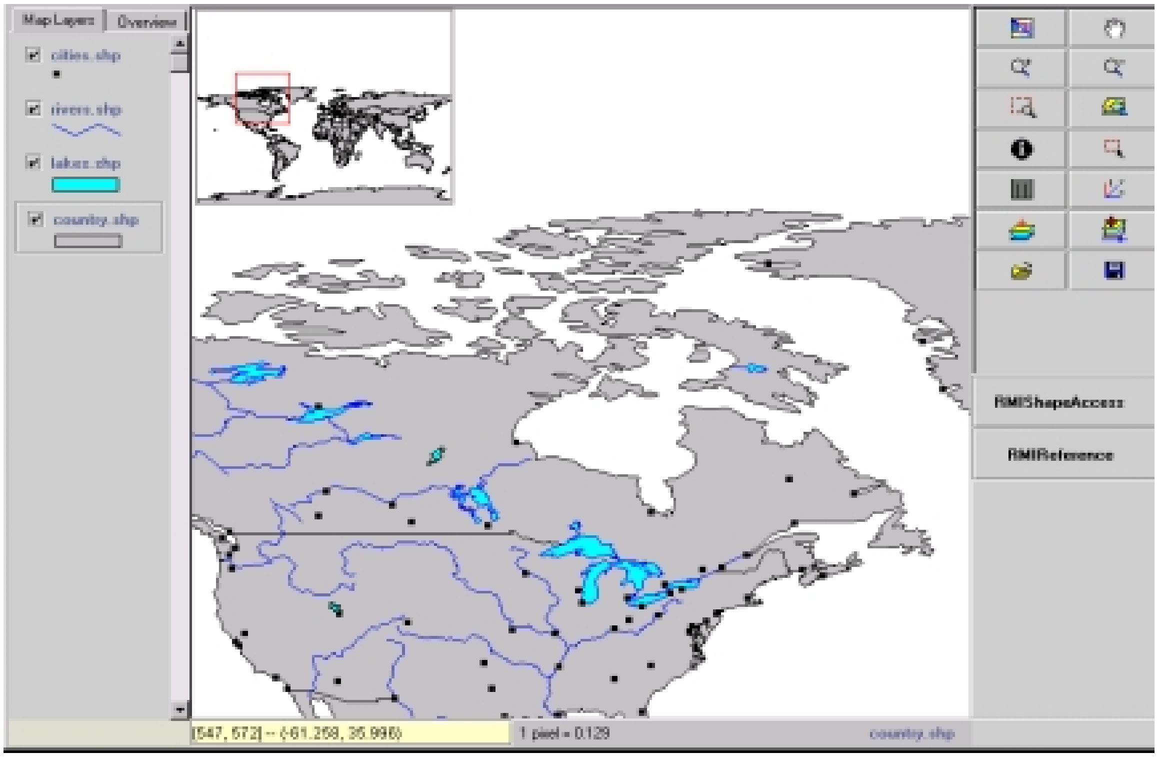


Figure 36: Extended Applet Interface

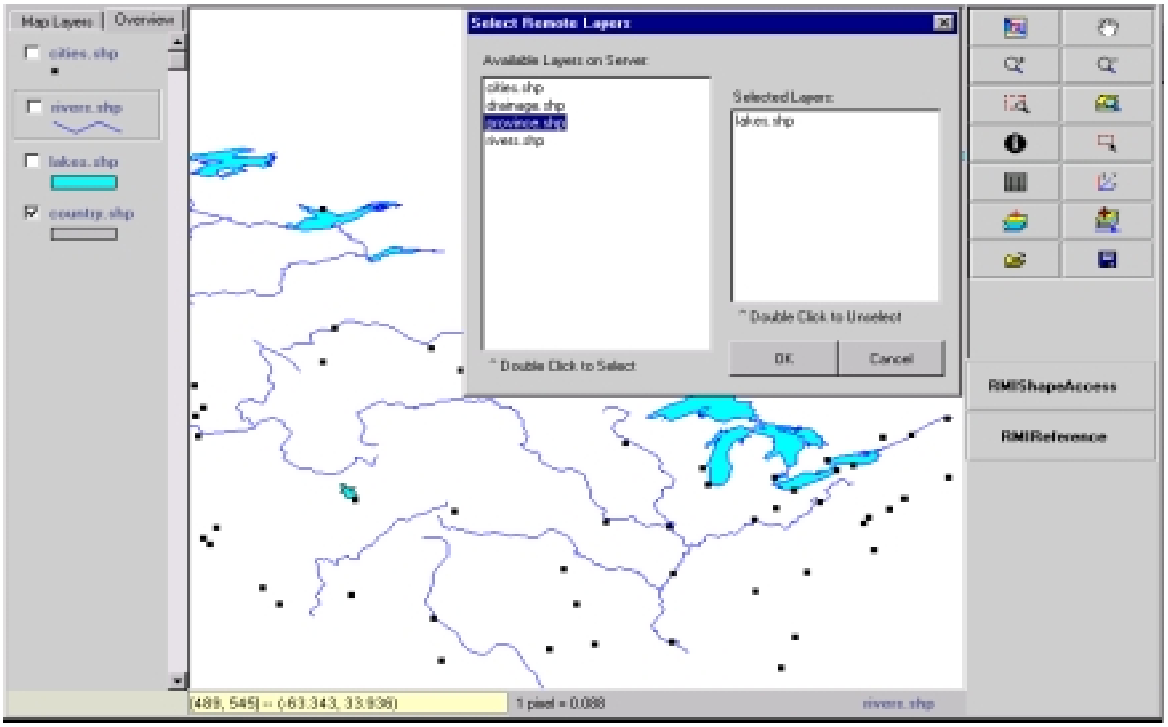


Figure 37: Remote Geodata Access Component Interface

4.3 Examples –The Implementation of Distributed Geoprocessing Components

The geoprocessing components to be implemented under this model can be very diverse, however, only some simple examples are implemented to demonstrate and test the model.

4.3.1 Distributed Geoprocessing Component Examples

The purpose of the development of geoprocessing component examples is to demonstrate the application of the model and test the model performance. Simple examples used in the implementation are the remote ESRI Shape File access and Geo-Reference System Transformation components. Different types of applet extensions have been tested in different servers. As shown in Figure 38, simple approaches such as Java Servlet and applet extension are applied in the web server. In the geoprocessing server, the same example components are used in different component wrappers and communication method, the Java RMI, which demonstrate the distributed computing ability of the model.

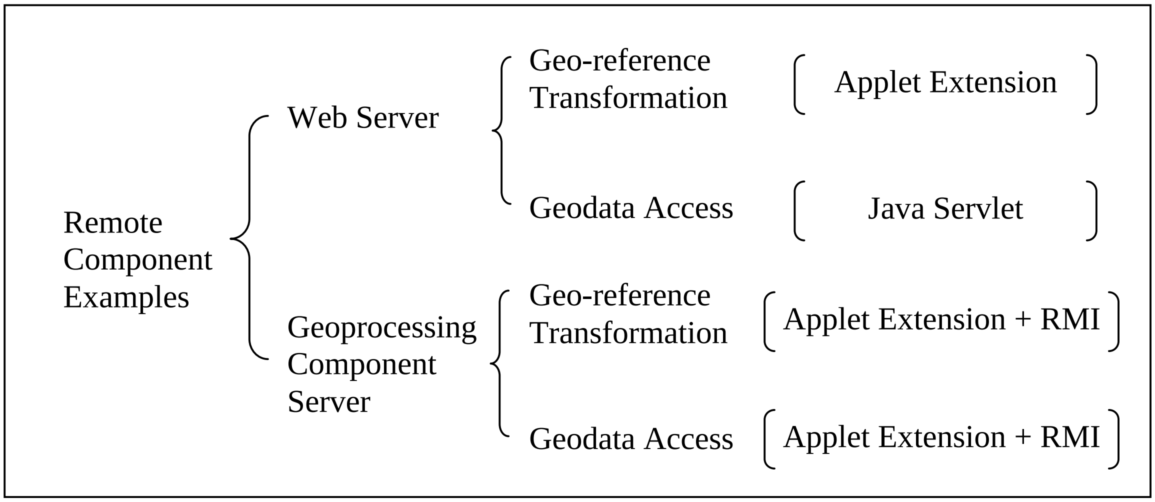

Remote Component Examples
Web Server
Geoprocessing Component Server
Geo-reference Transformation
Geodata Access
Geo-reference Transformation
Geodata Access
Applet Extension
Java Servlet
Applet Extension + RMI
Applet Extension + RMI

Figure 38: Application Examples Implemented in the Prototype System

The shape data access component is a very simple example to demonstrate the capability of "data anywhere" of the model. Each geoprocessing server, if it has geodata that wants to be provided to the clients, should have its own geodata providing components. These components could be as simple as a certain type of geodata access component as in the sample implementation, or as complex as a geodata catalog service component that can provide diverse distributed geodata to the clients.

Generally two types of applet extensions can be adopted to develop the components. For the components that have simple functionality and short code length, such as data format conversion, geo-reference system transformation, etc. direct applet extension should be the best choice, since they can avoid large remote geodata transportation. However, if the geodata exist on a remote server and geodata transportation cannot be avoided therefore, a RMI component may be the best choice. Especially, on the web server, Java servlet is a convenient choice for providing these services.

In the model design, data access objects and the geo-reference system are designed relatively independent from the internal data model. During the implementation, they are developed in components which can be separated from other common objects such as geometry objects, Layer, Map, etc. This is the reason why they are used as component examples.

The user interface and client/server communication sequence of shape data access component in a geoprocessing server is the same as the one used in shape data access servlet in the web server, as shown in Figure 29, 30, section 4.2.1. The difference is that Java RMI is used in communications instead of applet-servlet communications.

Geo-reference system transformation component has been implemented and tested in both applet extension and Java RMI component. The user interface and component classes are the same in these two types of extensions. The efficiency of the applet extension is much better since it is running locally in the client machine and there is no need to transfer the large geodata. However, geodata transportation is inevitable if RMI extension method is used. The followings list the sequence of the client/server communication during the RMI process:

(1) the user fills in the geo-reference system information in the given interface and press the Transform button (see Figure 39);

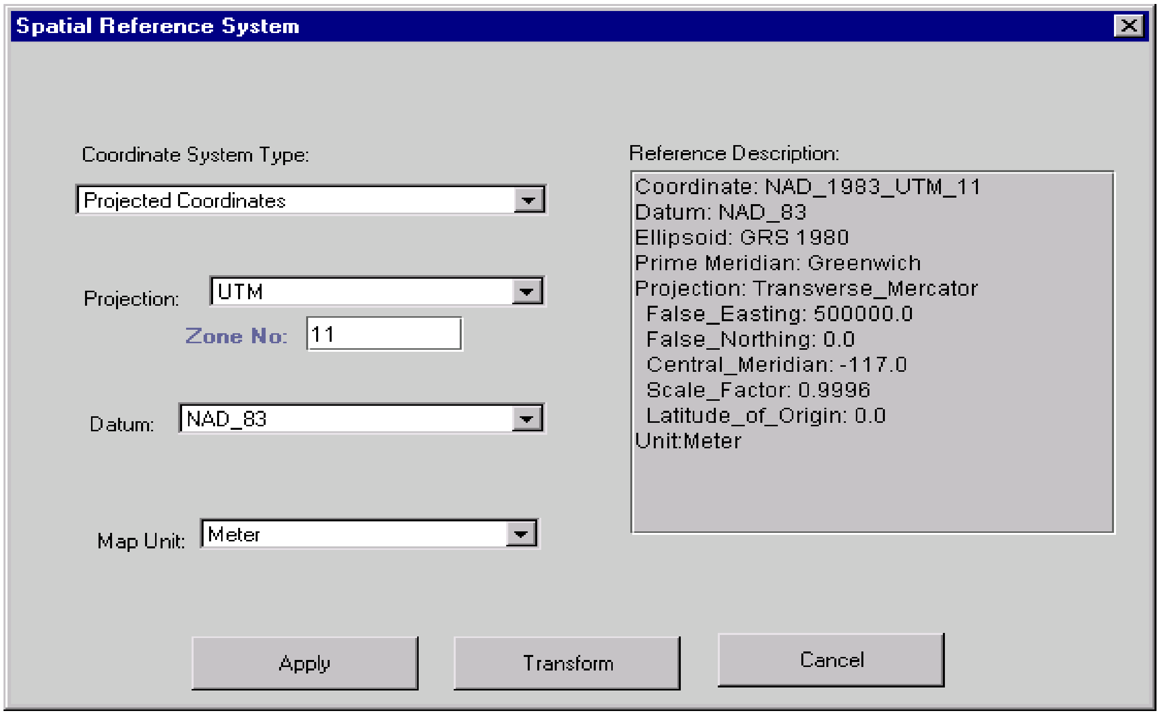
(2) the client program sends the current map and the new geo-reference object to the geoprocessing server (the existing geo-reference information is stored in the Map);

(3) the server receives the map and the new geo-reference object;

(4) the server performs the transformation and return the result map to the client; and

(5) the client replaces the old map with the new one and displays it.

The current geo-reference component supports horizontal geospatial coordinate transformation, including transformation between geographical coordinated system (Latitude-longitude) and projected coordinate system such as UTM, change of datum (NAD27, NAD83), unit (linear and angular) and projections.



Coordinate System Type: Projected Coordinates

Projection: UTM

Zone No: 11

Datum: NAD_83

Map Unit: Meter

Reference Description:

```
Coordinate: NAD_1983_UTM_11
Datum: NAD_83
Ellipsoid: GRS 1980
Prime Meridian: Greenwich
Projection: Transverse_Mercator
  False_Easting: 500000.0
  False_Northing: 0.0
  Central_Meridian: -117.0
  Scale_Factor: 0.9996
  Latitude_of_Origin: 0.0
Unit:Meter
```

Figure 39: Geo-reference System Transformation User Interface

Formulas used in UTM projection in the geo-reference component is the most practical form of equations, in which a set of series approximations converge rapidly to the correct centimeter or less at full scale in a zone extending 3° to 4° of longitude from the central meridian. Beyond this, the forward series as given is accurate to about a centimeter at 7° longitude, but the inverse series does not have sufficient terms for this accuracy (Snyder 1987). The formulas are given in Appendix B.

The above component examples demonstrated that the proposed distributed geoprocessing service model is "data anywhere and geoprocessing anywhere model". The geodata accessibility and the functionality of the geoprocessing component depend on how the components are developed. The extendibility of the model by distributed geoprocessing components is the key feature of the service model.

4.3.2 Potential Use of the Distributed Geoprocessing Service Model

Although only simple component examples are developed so far for this model implementation, there are many potential uses of the model in both GIS industry and research. Figure 40 illustrates the classification of these potential applications.

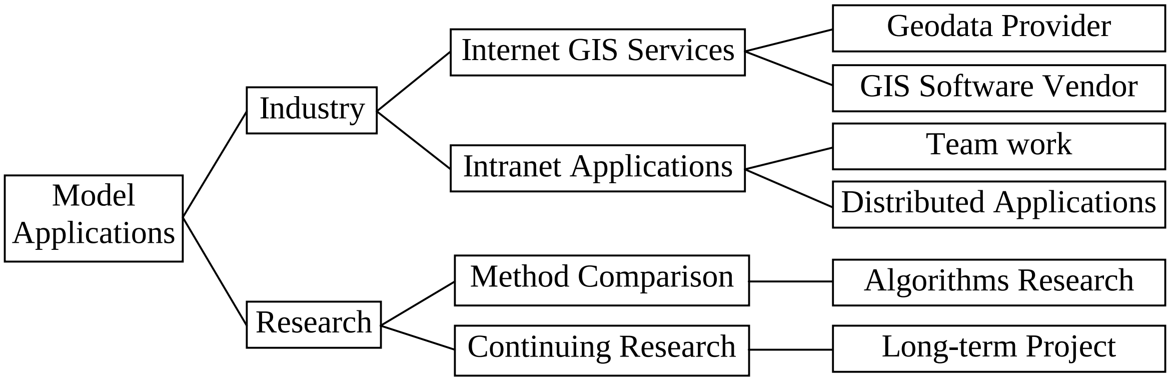


Figure 40: Potential Uses of the Distributed Geoprocessing Model

In Industry

For geodata providers, they can apply the model to extend their old geodata catalog service by adding some geoprocessing components to let users operate the data. The old geodata catalog service may become a sub-system or a component of the model. If a provider has street maps and the corresponding address database of a city, in addition to the old services that only allows users to search and view the maps, they can add geoprocessing services such as geocoding, best route analysis to help users use their data to make decisions. Figure 41 demonstrates the integration of geodata catalog service and other geoprocessing services.

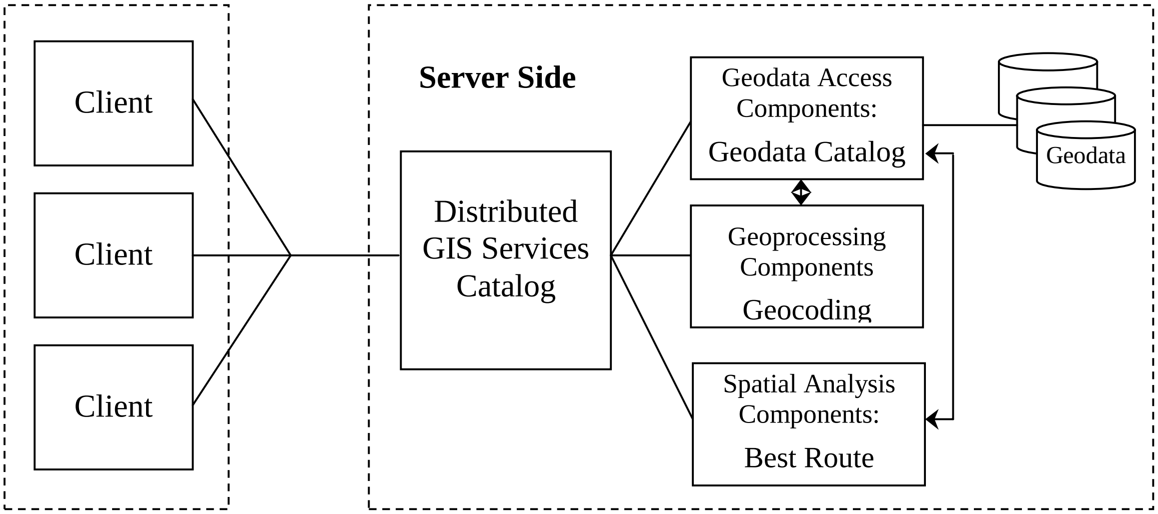


Figure 41: Integration of Geodata Services and Geoprocessing Services

The integration of distributed geodata access and distributed geoprocessing is very important in GIS service providing systems. Further research in this area will be of great value to GIS industry.

For GIS software vendors, no matter GIS software companies or individual developers, under the proposed geoprocessing service model, the development of GIS software will be component oriented instead of large system oriented. The components may be "rent" to users instead of being sold. The components will be maintained at the vendor's site, so updating and upgrading of the component become easier.

The components that can be developed under the proposed model may cover a very wide area. Many geospatial domain services can be developed using this model. In OpenGIS Service Architecture (OpenGIS 1999a), 15 categories of GIS services are defined, as shown in Figure 42:

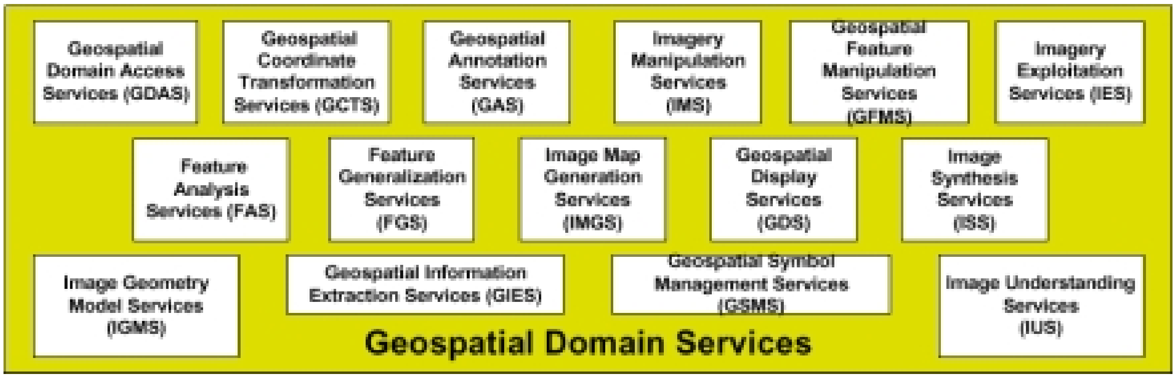

Geospatial Domain Access Services (GDAS)
Geospatial Coordinate Transformation Services (GCTS)
Geospatial Annotation Services (GAS)
Imagery Manipulation Services (IMS)
Geospatial Feature Manipulation Services (GFMS)
Imagery Exploitation Services (IES)
Feature Analysis Services (FAS)
Feature Generalization Services (FGS)
Image Map Generation Services (IMGS)
Geospatial Display Services (GDS)
Image Synthesis Services (ISS)
Image Geometry Model Services (IGMS)
Geospatial Information Extraction Services (GIES)
Geospatial Symbol Management Services (GSMS)
Image Understanding Services (IUS)
Geospatial Domain Services

Figure 42: Geospatial Domain Services (Source: OpenGIS 1999a)

Some of the above services are divided into more specific services, such as Geospatial Domain Access Services (GDAS), Geospatial Feature Manipulation Services (GFMS), Feature Analysis Services (FAS), and so forth. Figure 43 gives the extensions of these categories. These services can be further divided into smaller services according to needs.

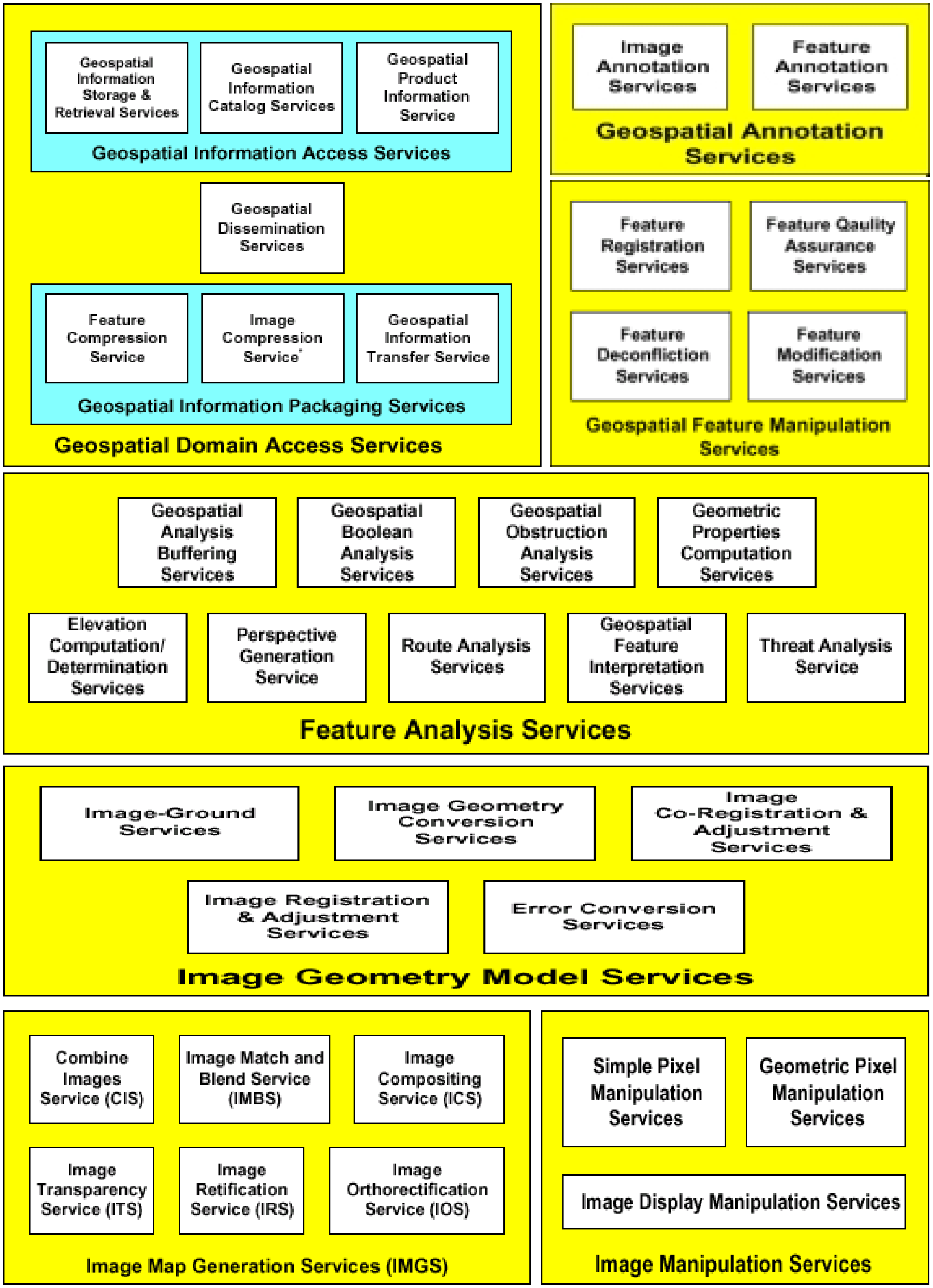

Geospatial Information Storage & Retrieval Services
Geospatial Information Catalog Services
Geospatial Product Information Service
Geospatial Information Access Services
Geospatial Dissemination Services
Feature Compression Service
Image Compression Service*
Geospatial Information Transfer Service
Geospatial Information Packaging Services
Geospatial Domain Access Services
Image Annotation Services
Feature Annotation Services
Geospatial Annotation Services
Feature Registration Services
Feature Qaulity Assurance Services
Feature Deconfliction Services
Feature Modification Services
Geospatial Feature Manipulation Services
Geospatial Analysis Buffering Services
Geospatial Boolean Analysis Services
Geospatial Obstruction Analysis Services
Geometric Properties Computation Services
Elevation Computation/ Determination Services
Perspective Generation Service
Route Analysis Services
Geospatial Feature Interpretation Services
Threat Analysis Service
Feature Analysis Services
Image-Ground Services
Image Geometry Conversion Services
Image Co-Registration & Adjustment Services
Image Registration & Adjustment Services
Error Conversion Services
Image Geometry Model Services
Combine Images Service (CIS)
Image Match and Blend Service (IMBS)
Image Compositing Service (ICS)
Image Transparency Service (ITS)
Image Retification Service (IRS)
Image Orthorectification Service (IOS)
Image Map Generation Services (IMGS)
Simple Pixel Manipulation Services
Geometric Pixel Manipulation Services
Image Display Manipulation Services
Image Manipulation Services

Figure 43: Geoprocessing Services in Their Categories (Source: OpenGIS 1999a)

In addition to the Internet services providing, The proposed model also may apply to an Intranet environment. For software development, the model will establish a basic working environment for the development team in which each team member can work relatively independent in specific component development. For distributed applications in an Intranet, distributed enterprise resources can be accessed efficiently in the same way as they are in the Internet in principle. The user interfaces and security management can be even more flexible. For the user interfaces, not only Applet, but also Application for the client become acceptable.

In Research

The proposed distributed model may also be used in research, in which different methods need to be compared. In GIS discipline, a lot of geoprocessing algorithms need to be developed and improved. This model can provide a basic test platform for different methods or algorithms. Different methods or algorithms can be developed in different components and added to the model so that the performances and efficiencies of the components can be compared. Since the model is a distributed model, distributed geoprocessing algorithms can be tested in this model.

At universities, research projects are usually conducted by students. Therefore, for long-term projects the continuity of the research remains a problem when students graduate and leave. Under the proposed model, after the establishment of the basic framework, student's work could be relatively independent. All the students have to do is extend the model by adding new components into the model. Students can concentrate on different components of a project, so that the cooperation among the students become looser and easier. The negative impact of student's graduation to the project will be minimized and new students can easily join the project.

CHAPTER 5 CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The topic presented in this research is one of the most important issues in the migration of GISystems towards GIServices. As geodata become available over the Internet, people enjoy the convenience of new geodata services. On the other hand, when people want to use these on-line geodata to do some geoprocessing or analysis, they feel disappointed because of the limitation in geoprocessing tools. The GIService providers realize the shortcomings of the tools but nothing can be done, since it is the problem of current GIService model. The existing GIService model, no matter the thin-client CGI model, the plug-in plus intelligent document model, or the thick-client Java applet / ActiveX model, may be good in terms of geodata and some fundamental data query and display functionality providing. However, they are not good enough in terms of providing distributed geoprocessing tools to the users. A new GIService model that support distributed geoprocessing tools in the Internet is therefore developed.

The proposed model is a "geodata anywhere, geoprocessing anywhere" model. Solutions provided are concentrated on "geoprocessing anywhere" problem, since we think "geodata anywhere" can be solved by geodata access services in which a lot of researches have been done and the technology is relatively mature, such as geodata catalog services.

The geodata access services can be a component of the proposed model to achieve "geodata anywhere".

The distributed component registration sub-model plays a key part in the GIService model. The idea of component registration is new and it turns out to be practical in support of distributed component interoperability. Some new concepts are therefore introduced to the model study, such as component meta-information and component wrapper that make the component interoperable. With these new ideas or concepts, third-party software components could be added into the service model to extend the geoprocessing capability. These components could be distributed at any sites in the network and accessible to the client at run time. So in terms of both geoprocessing functionality and related geoprocessing servers, the proposed GIService model is a scalable model.

The proposed model is a Java technology based model in which a lot of Java features are included in the model design and implementation. Also the model could be implemented with other technologies such as DCOM and CORBA by carrying out slight modification. Either DCOM or CORBA has the correspondences or interfaces to Java technologies. Java technologies have a lot of advantages in Internet programming comparing to other technologies. It is the platform independent feature, simple implementation requirements and acceptable network communication efficiency that made Java our first choice.

In this research, a sample implementation of the model has been conducted to demonstrate and test the design of the model. A client user interface program – the GeoEye™, a server side implementation that support distributed component – the GeoServnet, and some component examples have been developed from ground up. A relational object data model, Open GIS geo-reference object model, and component registration model are therefor established. The implementation of these models itself is a good practice in Open GIS development.

The geoprocessing components and functionality that are provided in the proposed model could be very diverse. It could be used in two ways:

(1) Combine geodata services with geoprocessing services: provide geodata services with distributed geoprocessing tools available.

(2) Provide geoprocessing services: "rent" software components to users for the processing of local and distributed data.

The following gives some possible components that are useful to users:

- GIS data access services: Discover, Catalog, Store, and Access different geodata from different locations.
- Data transformation services: Projection transformation, Coordinate transformation, Data format transformation, etc.
- Geospatial information extraction services: Extract specific features from images or maps.

- Feature generalization services: Generalize features from large-scale maps to small-scale maps.
- Feature manipulation services: Map editing, Feature cleaning, Feature digitizing, Topology building, Geocoding, etc.
- Geospatial analysis services: Network analysis, buffering, etc.
- Geospatial modeling services: TIN modeling, 3D visualization, etc.
- Geospatial annotation and symbolization services: Map making, etc.
- Image processing services, etc.

Studies and applications of distributed geoprocessing service model could finally promote the evolution of GISystems towards GIServices and distributed or ubiquitous geodata and geoprocessing will come to reality. Firstly, this evolution will change the way GIS software vendors develop and deploy GIS software. Instead of developing and selling large powerful GIS systems, GIS components will become more popular and rent to users through the Internet. Secondly, this evolution will change the way GIS information providers provide GIS information. Geodata and geoprocessing tools will be integrated and available ubiquitously in the network. Thirdly, this evolution will change the way GIS users use GIS software. Users do not have to pay the full license fee for the GIS system in which most of functions they never or seldom use. They can pay right according to their needs.

The following concludes the author's contributions in the thesis research:

- Developed a distributed geoprocessing model, which may produce a positive impact to the existing GIS service model.
- Introduced some new concepts for distributed geoprocessing researches, such as component registration, component meta-information, and component wrapper, which made component interoperable in the distributed model. It is expected that these new concepts will produce good affects to the relevant researches.
- Implemented a prototype GIService system in pure Java technologies, which involved the design and development of data model, geospatial reference model, component registration model, etc. The implementation of these models itself is a good practice in Open GIS development.
- The developed programs are from ground up and turned out to be very solid. Some of the implementation of industry or research value, such as GeoEye™, which has been using in the industry, and GeoServnet, which provides a solid foundation for the continuing studies in Distributed GIS.

5.2 Recommendations

Distributed geoprocessing service model is a new research topic that is expected to have a brilliant prospect. The research conducted is a preliminary start to enable distributed geoprocessing services. A lot of issues under this topic are of further research value to improve the proposed model and bring the model to practice uses.

(1) Open GIS data model

Open GIS Consortium published a series of abstract and implementation specifications to promote the interoperability of GIS. They are very good references in the model design and implementation. For geodata model, they defined OpenGIS Simple Features Specification for OLE/COM and CORBA implementation in which basic 2D GIS geometric objects, relationships of these objects, operational interfaces of these objects are defined. Unfortunately there are no Java implementation specification available so far. The current data model used in the proposed service model is slightly different from the above Open GIS implementation specifications, and many spatial operations have not defined in it due to the tight time limitation. Definitely if use an Open GIS data model, it will be more standard and easier to extend. It is recommended to rebuild the data model completely according to one of the above specifications.

(2) Component meta-information

The component registration model is the core of the proposed service model to support distributed geoprocessing. The component meta-information is the key concept of this core. In this research, some important items have been defined. However, this is only a preliminary start. The concept of component meta-information is very similar to that of metadata that describes geodata. In metadata, people have systematically defined many metadata items, from both practical and theoretical perspectives, and metadata standards become available as the result. From metadata concept to standards, it experienced several decades. Similarly for component meta-information, there should be a long way to go to become mature.

(3) Security and metering management

The proposed geoprocessing service model did not discuss the issue of network security management. However, security management should be an important part of the model if one wants to use the model in practice, especially in the Internet environment. A security management sub-model needs to be developed therefor, although it is more business than GIS research. Fortunately, Java technologies have very good reputation in network security management and many successful secure systems are developed, such as on-line banking. There is no reason to doubt Java security capability.

Another sub-model to bring the geoprocessing services to the market is the metering management. Without metering to support component renting, the service model can only provide free services. However, software metering is not a new concept and many successful models can be references.

(4) Open computing interfaces (CORBA)

The current model is a pure Java model and only Java components can be added to the model. This limitation can be changed by develop open interfaces to CORBA, a platform and language independent open distributed computing architecture.

(5) Distributed geoprocessing algorithms

In a distributed environment, many geoprocessing algorithms need to be redesigned and need to be optimized to minimize the response time of remote object invocation. Parallel computing methods and compressed object transportation can be used to improve the efficiency of algorithms. Therefore, in the development of geoprocessing components, researches on distributed spatial handling algorithms need to be emphasized.

(6) Transaction control

The current model does not address the transaction control of geodata operation. However, in certain applications, when multiple clients are working on the same geodata set in a collaborative manner, to maintain the data consistency, transaction control will become significantly important (Tao and Yuan, 2000b). Therefore, when developing remote geodata collaboration components, transaction control techniques need to be studied and applied.

REFERENCES

Abel, D. J., 1998, Towards integrated geographical information processing, International Journal of Geographical Information Science, 1998, Vol. 12. No.4, pp353-371

Albertson, T., 1998, Best Practices in Distributed Object Application Development: RMI, CORBA and DCOM, Developer.Com Journal, February 1998

Asbury, S. GIS Industry Outlook 2000: The Birth of a New Millennium, GeoWorld, December 1999, Vol. 12, No. 12

Autodesk, 1997, Autodesk MapGuide: State-of-the-art network-centric GIS application architecture for publishing and accessing geodata, A White Paper Series of Autodesk Inc., Web Document, http://www.autodesk.com/solution/gis/whtpaper/index.htm, last visited in December 1999

Buehler, Kurt, 1998, OpenGIS Technology Development Overview, OpenGIS Consortium Presentations, web document, http://www.opengis.org/techno/presentations/overview/index.htm, last visited in December, 1999

Buehler, Kurt and McKee, Lance, 1998, The OpenGIS Guide, Introduction to Interoperable Geoprocessing and the OpenGIS Specification, third edition, Open GIS Consortium Technical Committee

Charles, B., et al, 1999, Adding an Interoperable Server Interface to a Spatial Database: Implementation Experiences with OpenMap, in Interoperating Geographic

Information Systems, Proceedings of Second International Conferernce: INTEROP'99, Zurich, Switzerland, 1999

Coppock, J. and Rhind, D., 1995, The History of GIS in Geographical Information Systems - Principles and Applications, edited by David Maguire, Micheal Goodchild and David Rind, New York, pp21-43

Cornell, G. and Horstmann, C. S., 1997, Chapter 15: Remote Objects, in book: Core Java, second edition, SunSoft Press, 1997

Cuthbert, A, 1999, OpenGIS: Tales from a Small Market Town, in Interoperating Geographic Information Systems, Proceedings of Second International Conferernce: INTEROP'99, Zurich, Switzerland, 1999

Engen, David, 1997, Using MapObjects With Java to Internet Enable GIS applications, 1997 ESRI International User Conference Proceeding, Paper 207

ESRI, 1990, Understanding GIS: The ARC/INFO Method, Redlands, CA: Environmental System Research Institute, pp1.2

ESRI, 1995, ARC/INFO: The World's GIS, ESRI White Paper Series, March 1995, Environment Systems Research Institute, Inc.

ESRI, 1999, MapObjects 2 Object Diagram, Web Document last visited on March 18, 2000, http://www.esri.com/library/whitepapers/mo_lit.html

ESRI, 2000, ArcInfo 8: A New GIS for the New Millennium, an ESRI whitepaper, January 2000, http://www.esri.com/library/whitepapers/pdfs/ai8_newmill.pdf

Evans, J. D., 1999, Interoperable Web-based Services for Digital Orthophoto Imagery, Photogrammetric Engineering & Remote Sensing, May 1999, Vol. 65, No. 5, pp567-571

FGDC, 1998, Content Standard for Digital Geospatial Metadata, FGDC-STD-001-1998, http://fgdc.er.usgs.gov/standards/documents/standards/metadata/v2_0698.pdf

Gardels, K., Html, A Comprehensive Data Model for Distributed Heterogeneous Geographic Information, http://www.regis.berkeley.edu/gardels/geomodel_def.html, last visited on March 20, 2000

Gebharlt, J. C. and Henderson, L., 1999, WebCGM – Industrial strength vector graphics for the Web, CGM Open Consortium, Inc. web document: http://www.cgmopen.org, last visited on March 12, 2000.

Gopalan, Suresh Raj, HTML, A detailed Comparison of CORBA, DCOM and Java RMI, http://www.execpc.com/~gopalan/misc/compare.html

Intergraph, 1995, GIS ... The MGE Way: Take the Open Road with Intergraph, An Intergraph Technical Paper, Intergraph Corporation, Web Document last visited in December 1999, http://www.intergraph.com/mge/mgegis.htm

Intergraph, 1997, “Trends in GIS”, A White Paper on Geographic Information System (GIS) based on Jupiter Technology, Intergraph Corporation.

Intergraph, 1999, GeoMedia Web Applications - GIS For The Web, Intergraph Corporation White Paper Series, July 1999, Web Document last visited in December 1999, http://www.intergraph.com/software/geoengineering/press/gwe_white.asp

Intergraph, 2000, Internet/Intranet Online Publishing, GeoMedia Web Map White Paper, web document, http://www.intergraph.com/software/gwmregister/white_paper.asp#5.1.2, last visited on March 12, 2000.

Kafatos, E, et al, 1999, Earth Observing Data Systems in the Internet Era, Photogrammetric Engineering & Remote Sensing, May 1999, Vol. 65, No. 5, pp540-548

Kuhns, R. D., 1998, Strategies for designing and Building Reusable GIS Application Components, 1998 ESRI International User Conference Proceeding, Paper 557

Larman, Craig, 1998, Applying UML and Patterns - An introduction to Object-Oriented Analysis and Design, Prentice Hall PTR, pp273-291

Limp, W.F., 1999, Don't Hit Warp Speed with the Wrong Equipment!, GEOWorld, Vol.12, No.11, November 1999

MapInfo Corp., 1999, MapX v4.0 User's Guide, web document last visted on March 18, 2000, http://www.mapinfo.com/mapx/html/mapx_docs.html

Müller, R., 1999, From GISystems to GIServices: Spatial Computing on the Internet Marketplace, in Intermediaries for Information Services, Humbolt-Universität zu Berlin, October 1999, pp195-211

OpenGIS Consortium, Inc., 1998, OpenGIS Simple Features Specification for CORBA, Version 1.0, http://www.opengis.org

OpenGIS Consortium, Inc., 1999a, The OpenGIS Abstract Specification Topic 12: OpenGIS Service Architecture, Version 4, OpenGIS Project Document Number 99-112.doc

OpenGIS Consortium, Inc., 1999b, OpenGIS Simple Features Specification for OLE/COM, Version 1.1, OpenGIS Project Document Number 99-050

Schell, David, 2000, Introduction to OpenGIS Consortium, Inc., OpenGIS Consortium Documents, 2000.

Smith, T. R.,1996, UCSB-UCGIS Proposed Research Priority: Distributed GIS architecture and semantic Interoperability for a Networked Information Society, Web document, http://www.ncgia.ucsb.edu/research/ucgis/proposals/networked.html, last visited at December 20, 1999

Synder, J. P., 1987, Map Projections – A Working Manual, U.S. Geological Survey Professional Paper 1395, United States Government Printing Office, Washington, 1987, pp60-64

Tang, Q., 1998, Component Software and Internet GIS, 1998 CPGIS Conference proceeding, Association of Chinese Professionals in Geographic Information System (Abroad), 1998

Tao, C. V. and Yuan, S., 2000a, Development of Web-based GIService, Proceeding of GIS 2000 conference (CD-ROM), Toronto, March 14-16, 2000.

Tao, C. V. and Yuan, S., 2000b, Development of A GIS Service Model in Support of On-line Geoprocessing, Proceeding of GITA 2000 conference (CD-ROM), GITA (AM/FM) Association, March 26-29, 2000.

Thoen, Bill, 1999, How has the Internet Influenced GIS, GEOWorld, December 1999, Vol. 12, NO. 12

Wielderhold, Gio, 1999, Mediation to Deal with Heterogeneous Data Resources, in Interoperating Geographic Information Systems, Proceedings of Second International Conferernce: INTEROP'99, Zurich, Switzerland, 1999

Xia, F and Chao, C, 1995, The Internet GIS, in the Geographic Information Science, The Accociation of Chinese Professionals in Geographic Information System (Abroad), Vol. 1, No. 2, December 1995

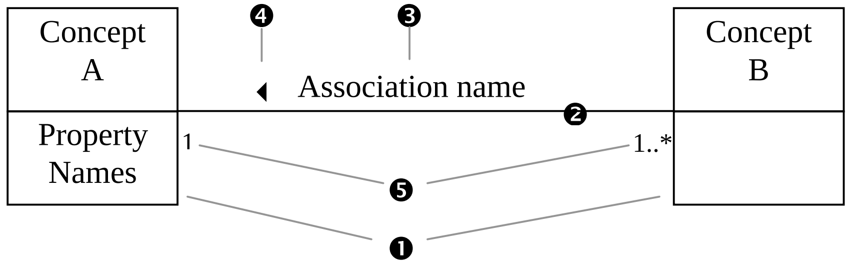
Yaakko, K., et al, 1999, Interactive Visualization of Geographical Objects on the Internet, International Journal of Geographical Information Science, 1999, Val. 13, N0. 4, pp429-438

Zhang, L., Li, B. and Lin, H., 1998, "A Model of GIS Virtual Machine", Geographic Information Sciences, Vol 4. No.1-2, December 1998, pp23-28

Zhang, M. and Zhang P., 1995, The design of the core GIS, Chinese journal of geography, Vol. 50, in Chinese

APPENDIX A DATA MODEL NOTATION

1. Conceptual Model Notation



In the above diagram:

❶: Concepts or types in a conceptual model. The upper part of the rectangle box gives the name of the concept and the lower part list the important properties of the concept. In many cases, the property name could be empty.

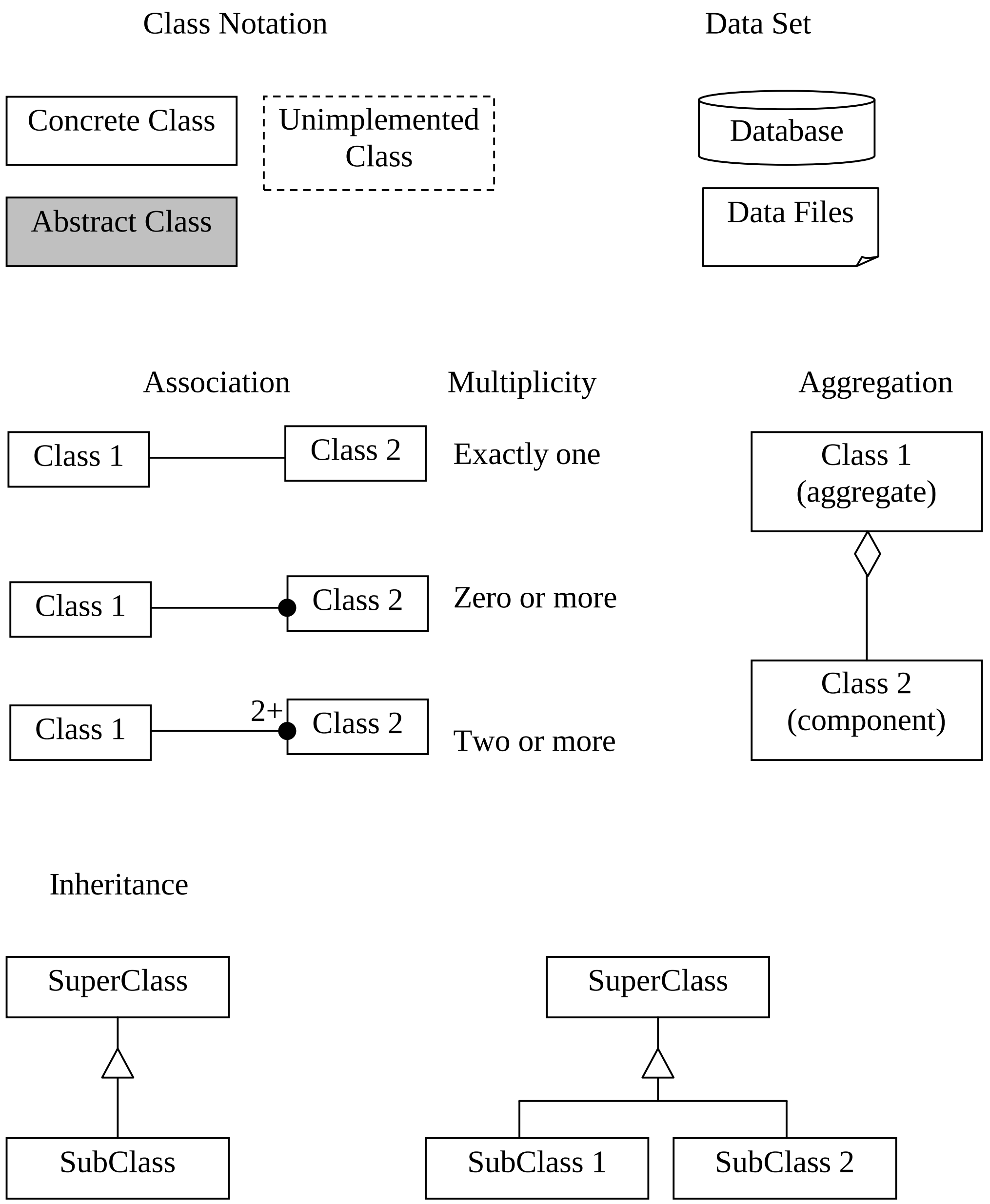
❷: The association between two concepts. In this thesis, dotted lines represent association between remote objects.

❸: Association name: The name that describe the association between two concepts.

❹: Direction of association: It gives the direction to read the association. The default direction of an association is from left to right or from top to bottom, in which the arrow can be ignored.

❺: Multiplicity of an association. For example: "1" means only one, "*" means many, "1..*" means 1 or more, and so forth.

2. Class Diagram Notation



APPENDIX B UTM PROJECTION FORMULAS

Forward formulas:

$$x = k_0 N \left[A + (1-T+C) A^3/6 + (5-18T + T^2 +72C-58e'^2) A^5/120\right] \qquad \text{(B-1)}$$

$$y = k_0 \{ M- M_0 +N \tan \phi [A^2 /2 + (5-T+9C+4C^2)$$

$$A^4/24 + (61-58T + T^2 +600C -330 e'^2) A^6/ 720] \} \qquad \text{(B-2)}$$

$$k = k_0 [1+ (1+C) A^2/2 + (5-4T+42C+13C^2 -28e'^2) A^4/24$$

$$+ (61 - 148T + 16T^2) A^6/720] \qquad \text{(B-3)}$$

where: k_0 = scale on central meridian, 0.9996 for UTM projection

$$e'^2 = e^2 / (1-e^2) \qquad \text{(B-4)}$$

$$N = a / (1-e^2 \sin^2 \phi)^{1/2} \qquad \text{(B-5)}$$

$$T = \tan^2 \phi \qquad \text{(B-6)}$$

$$C= e'^2 \cos^2 \phi \qquad \text{(B-7)}$$

$$A = (\lambda-\lambda_0)\cos\phi, \text{ with } \lambda \text{ and } \lambda_0 \text{ in radians} \qquad \text{(B-8)}$$

$$M = a[(1-e^2/4 - 3e^4/64-5e^6/256 - ...)\phi - (3e^2/8 - 3e^4/32 - 45e^6/1024+$$

$$...)\sin 2\phi + (15e^4/256+45e^6 1024 + ...)\sin 4\phi$$

$$- (35e^6/3072)\sin 6\phi + ...] \qquad \text{(B-9)}$$

with ϕ in radians.

M_0 = M calculated for ϕ_0, the latitude crossing the central meridian λ_0 at the origin of the x, y coordinates.

If $\phi = \pm \pi/2$, all equations should be omitted except (B-9), from which M and M_0 are calculated. Then x = 0, y = k_0(M - M_0), k = k_0.

To obtain UTM coordinates, the "false easting" 500,000.0m is added to the x.

Inverse formulas:

$$\phi = \phi_1 - (N_1 \tan \phi_1 / R_1)\,[D^2/2 - (5 + 3T_1 + 10C_1 - 4C_1^2 - 9e'^2)D^4/24 + (61 + 90T_1 + 298C_1 + 45T_1^2 - 252e'^2 - 3C_1^2)D^6/720\,] \qquad \text{(B-10)}$$

$$\lambda = \lambda_0 + [D - (1 + 2T_1 + C_1)\,D^3/6 + (5 - 2C_1 + 28T_1 - 3C_1^2 + 8e'^2 + 24T_1^2)\,D^5/120\,] / \cos \phi_1 \qquad \text{(B-11)}$$

where ϕ_1 is the "footpoint latitude" or the latitude at the central meridian which has the same y coordinate as that of the point (ϕ, λ).

$$\phi_1 = \mu + (3e_1/2 - 27e_1^3/32 + \ldots)\sin 2\mu + (21e_1^2/16 - 55e_1^4/32 + \ldots)\sin 4\mu + (151e_1^3/96 + \ldots)\sin 6\mu + (1097e_1^4/512 - \ldots)\sin 8\mu + \ldots \qquad \text{(B-12)}$$

where

$$e_1 = [1 - (1 - e^2)^{1/2}] / [1 + (1 - e^2)^{1/2}] \qquad \text{(B-13)}$$

$$\mu = M / [\,a\,(1 - e^2/4 - 3e^4/64 - 5e^6/256 - \ldots)\,] \qquad \text{(B-14)}$$

$$M = M_0 + y / k_0 \qquad \text{(B-15)}$$

with M_0 calculated from (B-9) for the given ϕ_0

$$e'^2 = e^2 / (1-e^2) \qquad \text{(B-4)}$$

$$C_1 = e'^2 \cos^2\phi_1 \qquad \text{(B-16)}$$

$$T_1 = \tan^2\phi_1 \qquad \text{(B-17)}$$

$$N_1 = a / (1-e^2 \sin^2\phi_1)^{1/2} \qquad \text{(B-18)}$$

$$R_1 = a\,(1-e^2) / (1-e^2 \sin^2\phi_1)^{3/2} \qquad \text{(B-19)}$$

$$D = x / (N_1\, k_0) \qquad \text{(B-20)}$$