5-21-2004

COTS GIS Integration and its Soap-Based Web Services

Ying Wu
University of New Orleans

COTS GIS INTEGRATION AND ITS SOAP-BASED WEB SERVICES

A Thesis

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Master of Science
in
The Department of Computer Science

by

Ying Wu

Master of Engineering, Beijing University of Aeronautics and Astronautics, 1999

Spring 2004

Dedicated to:

My parents, JingQun Wu and LiWen Ren

My Husband, Liang Xu

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Professor Shengru Tu, for his guidance, support and encouragement. His kindness and generosity made me feel very comfortable working with him.

I also want to thank the members of my committee, Prof. Mahdi Abdelguerfi, Prof. Ming-Hsing Chiu, for their priceless instruction, help and guidance on my graduate study.

I am also grateful to Dr. Shengru Tu and Dr. Mahdi Abdelguerfi for their kind help and financial support of my research. Without their support and help, I would never have been able to complete this project.

I deeply appreciate my husband, Liang Xu, for his greatest support of my life and study. Greatest thanks for my parents, who have encouraged and guided me to independence, thanks for their support and patience while I pursued this master's degree.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

In the modern geographic information systems, COTS software has been playing a major role. However, deploying heterogeneous GIS software has the tendency to form fragmented data sets and to cause inconsistency. To accomplish data consolidation, we must achieve interoperability between different GIS tools.

In my thesis project, I developed Vector and Raster Data Adapters to implement the spatial data consolidation. I deployed ArcIMS to publish the spatial data and metadata onto Internet. Furthermore, the SOAP-Based GIS Web services are implemented to achieve the enterprise information system integration.

The contribution of ours in this project is we have streamlined the COTS GIS server, the J2EE coordinator server, the web service provider components, and the COTS web publishing tools into a hybrid web service architecture, in which the enterprise information system integration, the web publishing, and the business-to business online services are uniformed.

CHAPTER 1

INTRODUCTION

Building systems from commercial off-the-shelf (COTS) products has been a dominant practice in the information technology (IT) industry, as highlighted by the COTS-based systems initiative at the Carnegie Mellon Software Engineering Institute [1] and the International Conferences on COTS-Based Software Systems [2]. The advantages of using COTS are numerous including market-tested reliability, market-approved features, and an opportunity for expanding software capabilities and improving system performance by the commercial marketplace. However, the claimed performance and functionality of the COTS based systems are too often not realized in practice. Indeed, there have been more failures than successes in using COTS software products [3]. The mounting experience from both success stories and failings lessons shows that the effective use of COTS software products in major software systems demands new skills, knowledge and different techniques and processes [27]. The uses of GIS COTS software products usually achieved better results. This is because of at least three reasons. First, the requirements of GIS were typically well defined in specific GIS domains. Second, the traditional uses of GIS tools were for a group of domain experts, or a department or a project of controllable size. Third, the GIS COTS software products are usually built based on scientific theories with rigorous mathematics. However, as more and more

geographic information systems become web enabled and enterprise-wide, the use of GIS COTS products encounters the same challenges that the general COTS based systems are faced with in system integration.

The leading GIS vendors such as ESRI and Intergraph are providing suites of products that span from the traditional GIS databases and applications to web publishing and global data sharing. While these efforts are undeniably fruitful and beneficial to the users, integration of multiple systems constructed with COTS products of different vendors has become keen for large institution users.

In this thesis, I report the design and experiments on integration of COTS software for a large organization, where a number of GIS COTS software installations including ESRI's ArcInfo and Intergraph's GeoMedia are in use and each of them has established substantial data sets and user bases. The final goal of this enterprise-wide project is to provide users of various types with harmonious views representing various businesses. Technically, our strategy is first to consolidate the data sets into a commonly shared database, then to deploy map publish software for the corporate web site and to establish SOAP-based web services for internal third-party software users and external business partners.

The foundation of our data consolidation efforts has been the spatial database standard defined by the Open GIS Consortium (OGC). Since most GIS software vendors have been supporting OGC's standard, this effort has been straightforward for the vector data sets. However, we have realized a significant gap in data consolidation: the standards of raster images are all file formats rather than direct database storage. Currently, the few advanced images database technologies are still proprietary. This gap

forced us to develop an adapter for the high-performance image map database of a leading vendor, ESRI's ArcSDE server. This adapter has made ArcSDE's image capacity accessible through the open standard protocol SOAP.

While web publishing and web services are welcome by various users, they must be supported by a sufficiently capable server that enforces business rules and handles hundreds of concurrent requests. For web map publishing, ESRI's ArcIMS servers has illustrated its very good capacity. However, for business-to-business and tool-to-tool web services, we had to develop an unbiased solution. In my experiments, I have chosen the industry's best practice – the Java 2 Platform Enterprise Edition (J2EE) server. This server abridges and coordinates between the COTS software and the database and the service delivery components. The theme of this coordinating server is to improve performance. Many traditional design principles for distributed software are applicable to the design of web services, but also are more crucial to apply to the GIS web services.

In the remaining parts of this thesis, Chapter 2 provides the background about GIS, the COTS GIS products used in this project, and the background information of this project. Chapter 3 describes our design and architecture of the whole system. In chapter 4, I provide the details of the implementation. Chapter 5 concludes this research.

CHAPTER 2

BACKGROUND

In this chapter, I describe the background of this project, Geographic Information System (GIS) and the diversified GIS related businesses of the New Orleans District of the U.S. Army Corps of Engineers ("the Army Corps" for short).

2.1 Geographic Information Systems

GIS is a tool used by individuals and organizations, schools, governments, and businesses seeking innovative ways to solve their problems. Mapmaking and geographic analysis are not new, but GIS performs these tasks better and faster than the old manual methods do. The U.S. Geological Survey agency defines a GIS as a computer system capable of assembling, storing, manipulating, and displaying geographically referenced information, the data identified according to their locations [5]. A map in a GIS works for a viewer by relating information from different sources, which are stored in separate layers in its dataset. By storing all kinds of correlation data into GISs' database according to layers and locations, we can easily query, update and insert information from many different sources in many different forms according to the different applications.

Geographic information systems work with two fundamentally different models, the vector model and the raster model [5]. In the vector model, information about points, lines, arcs and polygons is encoded and stored as a collection of two dimensional (2D) or

three dimensional (3D) coordinates. The location of a point feature such as a tree can be described by a single coordinate. Linear features such as pipes and streets can be stored as a collection of point coordinates. Polygonal features, such as buildings and blocks, can be stored as a closed loop of lines. Vectors model is very useful to describe the logic nodes with less detail information such as: a library in a university campus area. But it is not easy to describe continuously varied information. The raster mode is good at this way.

GIS has its special information structure named geographic references**.** There are two types of geographic reference, the explicit geographic reference and the implicit reference. The explicit geographic reference has the coordinate information. The implicit reference has no such information; it only contains some information such as Zip code and address. Both of geographic references allow you to locate features on the earth's surface for analysis, such as a business or forest stand, and events, such as an earthquake.

2.2 COTS software products for GIS

According to the perspective of the Software Engineering Institute (SEI), a COTS product is: sold, leased or licensed to the general public; offered by a vendor trying to profit form it; supported and evolved by the vendor, who retains the intellectual property rights; available in multiple, identical copies; and used without source code modification [6].

Using "Commercial Off-the-shelf" (COTS) software components to build systems has been a means of developing software to reduce risk and cost and increase functionality and capability of the system. Building a system based on COTS components involves buying a set of pre-existing, proven components, building extensions to satisfy

local requirements, and gluing the components together. The advantage claimed is that the COTS components are honed in the competitive marketplace resulting in increased capability, reliability, and functionality for the end user over what would be available from custom built components. COTS software components from different vendors are expected to be integrated, applicable in a wide range of environments, and support extensions and tailoring to local requirements.

Currently, leading vendors of GIS products such as ESRI and Intergraph as well as smaller vendors such as Z/I Imaging often provide application programming interfaces (APIs) or for users to connect their software products with others'. Many middleware vendors are filling their gaps by providing adapters between popular software suites. Their next step should be providing higher-level components such as J2EE-CA adapters. This has been a trend in other general software industry, but has not been seen in the GIS software communities [7].

In this project, we employed several leading GIS COTS products ArcGIS, GeoMedia and Microstation which are mainly software tools for generating spatial data for many GIS projects at USACE – New Orleans (US Army Corps of Engineers).

2.2.1 ArcGIS—GIS Software family from ESRI

The ArcGIS system is an integrated geographic information system consisting of three key parts [8]:

- ArcGIS Desktop Software, an integrated suite of advanced GIS applications.
- ArcSDE gateway, an interface for managing geodatabases in a database management system (DBMS).

- ArcIMS software, internet-based GIS for distributing data and services.

ArcGIS provides a framework for implementing GIS which can be deployed on a single desktop, or be distributed on a heterogeneous computer network of workstations and servers. Users can deploy various parts of this system to implement a GIS of any size – from a single-user system to large enterprise, and even societal GIS systems. It has the architecture as shown in Figure 1.

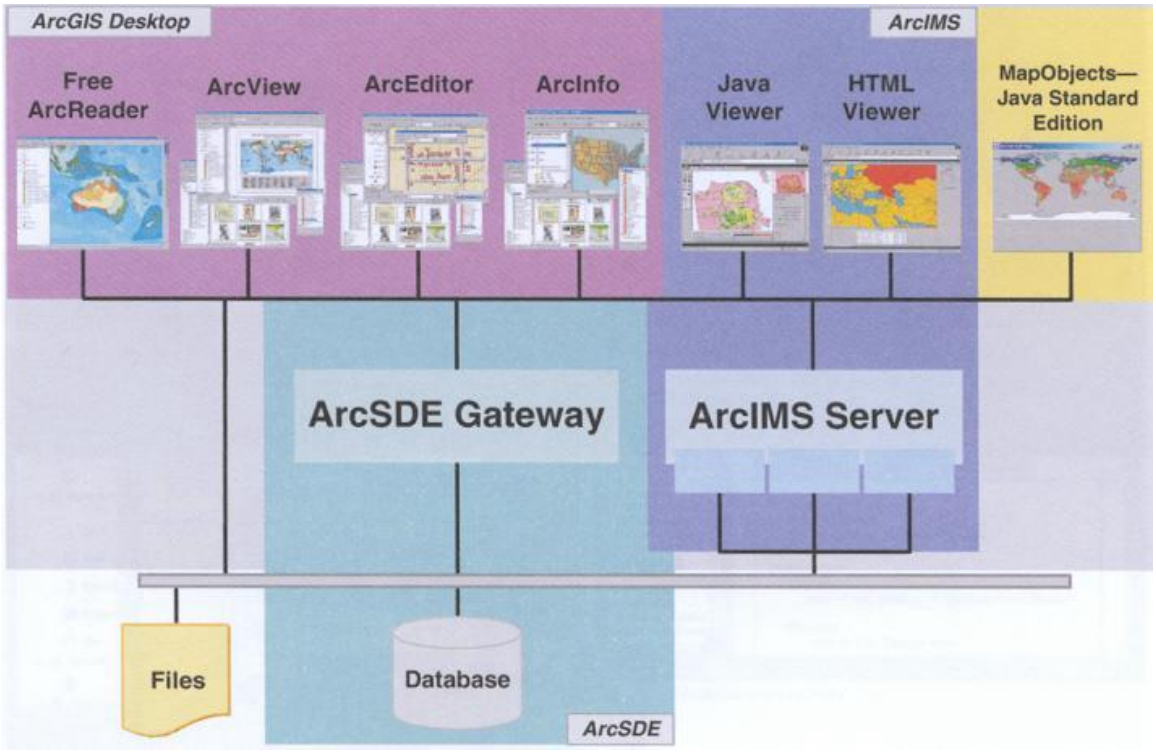


Figure 1: The architecture of ArcGIS

2.2.2 GeoMedia—desktop GIS Software from Intergraph

GeoMedia is one of software products of Intergraph Corporation. It provides the smart tools to capture and edit spatial data. You are able to use an industry-standard relational database for maintenance so you can keep up to date; analyze data

relationships; turn information into precise, finished maps for distribution and presentation; and put GIS data into the hands of users throughout the enterprise[9].

2.2.3 Microstation— CAD program from Bentley

Microstation is a full-featured 2-D and 3-D CAD program for MS-DOS, Microsoft Windows, Macintosh, and Unix workstations from Bentley Systems, Inc. created in 1984, MicroStation is a high-end package used worldwide in environments where many designers work on large, complex projects. MicroStation Modeler is a superset of MicroStation that provides solid modeling, and MasterPiece is MicroStation's rendering and animation program.

The latest version of Microstation is a single, comprehensive platform for design and engineering projects and is the foundation of the V8 Generation of software from Bentley. MicroStation capabilities make it better suited for 3D modeling, data versatility, workgroup productivity and application development [10].

2.3 The New Orleans District of the U.S Army Corps of Engineers

The U.S. Army Corps of Engineers (USACE) is the world's largest public engineering agency. The New Orleans District of the USACE plans, designs, constructs, operates and maintains federally sponsored navigation, flood control, hurricane protection and water resources development projects in 30,000 square miles of south central and coastal Louisiana. The district's jurisdiction includes more than 2,800 miles of navigable waterways, 950 miles of levees and floodwalls, 12 navigation locks, six major flood control structures, one freshwater diversion structure, and other projects to protect and enhance the coastal and inland wetlands of Louisiana. The district regulates activities

in navigation channels and wetlands, and provides real estate and engineering consultant support for other government agencies. The District helped make the ports of South Louisiana number one in the nation in total tonnage and number one in grain exports, and maintains 2,800 miles of navigable waterways including 400 miles of deep-draft channels (45 feet deep from the Gulf of Mexico to Baton Rouge). The New Orleans District of USACE operates 12 navigation locks. The District makes it possible to live and work along the lower Mississippi River by building 950 miles of levees and floodwalls and 6 major flood control structures to protect against river and hurricane flooding. The district's Old River Control Structure, northwest of Baton Rouge, prevents the Mississippi from changing its course (to the Atchafalaya River Basin). It was such a diversity that has forced the Army Corps to deploy heterogeneous GIS software suites, and motivated us to seek a flexible architecture to integrate existing fragmented GIS components.

2.4 The data resources and software applications at the Army Corps

With so many diversified businesses, the Information Service at the New Orleans District is supporting GIS applications, hydraulics numerical computation, spatial data warehouse design, and spatial data integration. The data and software are used to provide engineering management decision tools for short and long-term water resource, navigation, and environmental projects. Currently, the internal data resources are of various formats such as SWMM [11], UNET [12], LIDAR [13], geo-referenced images, USGS DEM [14], HEC [15], and ASCII.

At the Army Corps, the GIS software tools are mainly COTS products made by ESRI [16], Intergraph [17] and Bentley such as ArcInfo (ESRI’s flagship GIS toolkit for desktops and workstations, including ArcView and ArcMap), GeoMedia (including

GeoMedia Professional, Intergraph's flagship desktop GIS tool) and Microstation (MicroStation Version 7 and Microstation v8, Bentley's graphically oriented applications for Architecure/Engineering/Construction professionals). Each software suite has a substantial number of users running the daily businesses. In addition, some small software pieces are also used for special purposes. They were written in different languages such as Java, JSP, C/C++, and Tcl-Tk driven VTK toolkit [18, 19]. Currently, they access geographic information by files rather than databases.

As the data and applications have been constantly growing and diversifying, management of the data and applications become more and more challenging.

CHAPTER 3

SYSTEM DESIGN

3.1 The Project on System Integration

Most of the work reported in this thesis is based on a research project that supports the system integration for the New Orleans District of the U.S. Army Corps of Engineers (USACE). The USACE district plans, designs, constructs, operates and maintains federally sponsored navigation, flood control, hurricane protection and water resources development projects in south central and coastal Louisiana. Engineers and analysts have been using a myriad disparate commercial software packages to manage its GIS and CAD projects including products of companies such as Intergraph, ESRI, and Bentley. A team of IT workers have been working on the initiative called Enterprise GIS (E-GIS) aims at integration of all these software programs through a centralized means of data access.

The final goal of our project is to establish an Enterprise Geographic Information System (EGIS) that provides various users with consistent, responsive, secured and global views across different businesses.

This project has gone through two phases listed below:

- Phase One: spatial data consolidation

- Phase Two: COTS GIS integrations
- Phase Three: web map services, SOAP-based web services, and back-end server enhancement

Currently we are in Phase Three in which both proprietary and open-standard cutting-edge technologies are involved. It turns out that this gradual chicken-little approach has formed a systematic process toward the E-GIS's goal.

3.2 Spatial Data Consolidation

Since the data sets are fragmented at the Army Corps, pulling out maps and data sets from different databases or files is not automated and time consuming. As a result, the data sets of some areas are no longer consistent. That is, the updates for the same area spread over in multiple versions; no single version contains all the up-to-date data.

Learning from the lessons of having fragmented data sets, consolidating spatial data was our first attempt toward the E-GIS. It helps reduce the overhead of managing fragmented data sets, and guarantee data consistency and integrity. While consolidating vector data sets were straightforward, doing so for raster images lacks supports from industrial standards. I had to develop an adapter for ArcSDE server, the COTS product of best practice, that manages raster images. In Phase One, we accomplished the architecture in Figure 2.

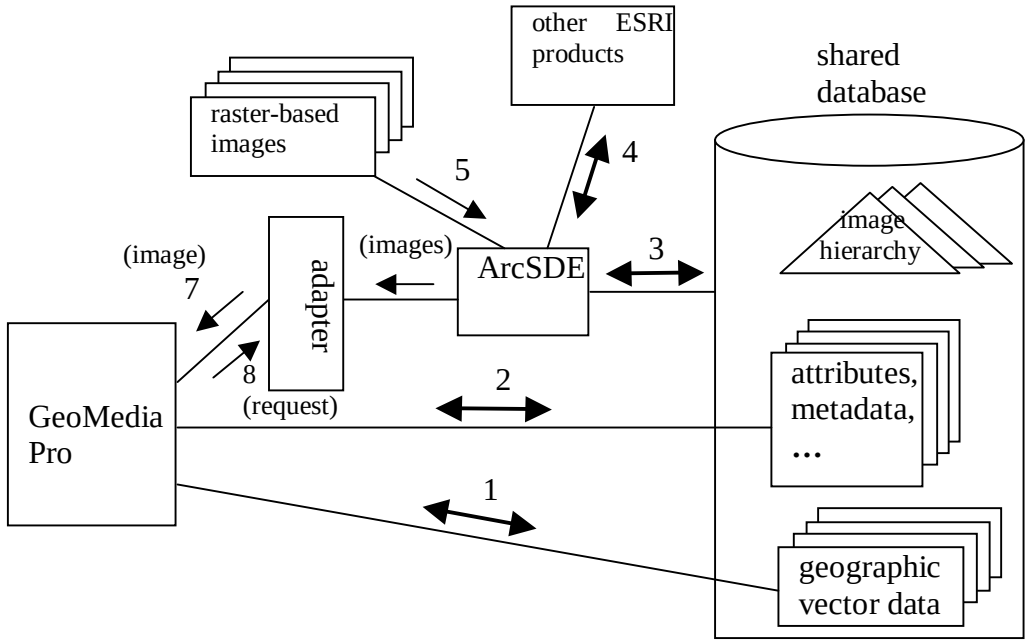


Figure 2: Spatial Data Consolidation

3.2.1 Consolidating Vector Data Sets

To store the vector data sets originally managed separately by either GeoMedia or ArcInfo into the same Oracle Spatial database can dramatically simplify the enterprise-wide data management. Owing to the tremendous efforts on interoperability by the Open GIS Consortium (OGC), most of the GIS vendors have made their products compliant -- capable of accessing and manipulating data stored in the OGC spatial database standard. Oracle Spatial Database is a leading COTS product that fully implemented OGC spatial database standard. In our project, the Oracle spatial database has taken a pivot role in managing vector data sets. Since the ESRI's ArcView and Intergraph's GeoMedia are both used at USACE, We have to be able to store the vector data sets originally managed by either GeoMedia or ArcInfo into the same Oracle Spatial database. This was achieved in Phase One. Having these two COTS products accessing the shared database directly,

the features of both COST projects such as catalog management can be leveraged to the full extent. This can dramatically simplify the enterprise-wide data management.

Our initial experiments were very convincing. As its initial characteristics, GeoMedia stores vector data sets in its data warehouse, a database. On the other hand, ESRI's products access databases through their spatial database engine, ArcSDE. By connecting ArcSDE to GeoMedia's warehouse database (an Oracle database), ArcView could immediately display every map made by GeoMedia through ArcSDE without any manual operation. Furthermore, ArcView's catalog viewer could display every item that GeoMedia's catalog could display. In another experiment, we imported the vector map files (called Shapefile) produced by ArcInfo into the Oracle database by using ArcSDE's importing capability. Then, with the "insert feature class metadata" command (a button click), GeoMedia can access every piece of geographic information and feature data in those maps, except for the high-level map descriptions.

3.2.2 Consolidating Raster Images

Traditionally, desktop GIS application tools all access images based on files. However, as the geographic information systems grow to enterprise-wide and Internet enabled, file-based image accessing has a number of disadvantages in supporting geographical query processing. Especially for large images, transmitting the files often unnecessarily slows down the response, when the user only needs to see a (small) part of the image.

In the past, we defined a simple hierarchical structure and stored large images into a database by partitioning the images into small blocks and recursively producing an overview hierarchy consisting of equal-size overview images that cover multiple blocks'

areas. Accessing images is optimized by fetching the right size of the image (or overview image) with the right resolution depending on the nature of the request. For example, loading an image with multiple megabytes would cause substantial delay. With the hierarchies in the database, the access delay will be kept under a reasonable threshold because a request to view any part of the same image always results in a few needed blocks to be delivered. This approach successfully supported a GIS web view service [22]. The advantage of doing so is the elimination of any long waiting time for loading large image files.

ESRI's ArcSDE stores images in the database by decomposing images into 128 by 128 pixels. The entire process of constructing the overview hierarchies is completely automated. Viewing any part of the images does not suffer from any tangible delay. By storing the images in the same database that hosts the vector data sets, seamless integration between vector and raster data sets is achieved easily. This is particularly important, because vector maps are often projected over images. ArcSDE's database approach to storage of images represents the best practice in the industry. However, it is not an industrial standard. Image format standards are all based on files.

GeoMedia handles images differently. The raster-based images are brought into GeoMedia's data warehouse but left out from the database. While projecting vector maps over raster-based images is well supported, no performance optimization is provided.

By consolidating images in the same database, the image data integrity problem can be naturally solved as we did with the vector data sets. Not only does this approach open an avenue for accessing speed improvement, it also allows us to assemble images across image file boundaries. These advantages are illustrated by ArcSDE. ESRI's

software tools from desktop applications to web publishing exhibit data consolidation and optimization. We chose to consolidate the raster data into ArcSDE Sever for web publishing and web services. I have developed an adapter that allows other software components including GeoMedia to retrieve images of any location, of any scope with a proper resolution level in a minimum time. The details of implementation is given in Section 4.3.1.

3.3 Back-End Server Enhancement

It has been clear that COTS software integration is more than sharing a database. Different COTS components need to interact with one another, often through some adapter. When the interactions happen in a large-scale system, service requests, responses and the service delivery components such as the adapters must be well managed. We have chosen the J2EE server technology as the mechanism for the back-end server management.

The Java™ 2 Platform Enterprise Edition (J2EE) defines an industrial standard of a software framework for developing multiple-tier enterprise applications. The J2EE standard intends to embrace resources with a component-based application model. The J2EE framework provides implementation of many commonly required system-level services including the database/data resource connections pooling service, the naming service, the transactions management, the persistence management, and the security management. Each of these services requires significant development efforts. The application servers that implement the J2EE standard are called J2EE application servers. J2EE application servers lift a great amount of burden from the developers and

integrators. As a result, they can focus on implementing business rules. The application business rules are encapsulated in the Enterprise Javabeans (EJBs) [20].

One of the key capabilities of J2EE application servers is connection pooling. Establishing a connection between systems is expensive in terms of time. The connection pool creates and then maintains a desired number of connections. The expense to establish connections is avoided by recycling old connections for new use. The popular Java database adapter for relational databases, JDBC, is native to J2EE application servers. Every J2EE application server provides JDBC connections pooling service. However, for data resources (such as a GIS) that provide high-level information (such as area maps), the J2EE application server does not know how to manage the connections. To solve this problem, the J2EE standard includes the J2EE Connector Architecture specification (J2EE-CA) as a protocol for non-relational data resources to be managed by J2EE application servers [21]. The J2EE-CA adapters are implemented in methods dependent on the data resource software such as a GIS. By encapsulating the application software dependency in the adapters, all the other components in the system are independent of the data resource software; this is a common practice. The unique advantage of the J2EE-CA is that any data resource software can fully leverage the J2EE application servers' system-level services by making the adapter compliant to the J2EE-CA. standard.

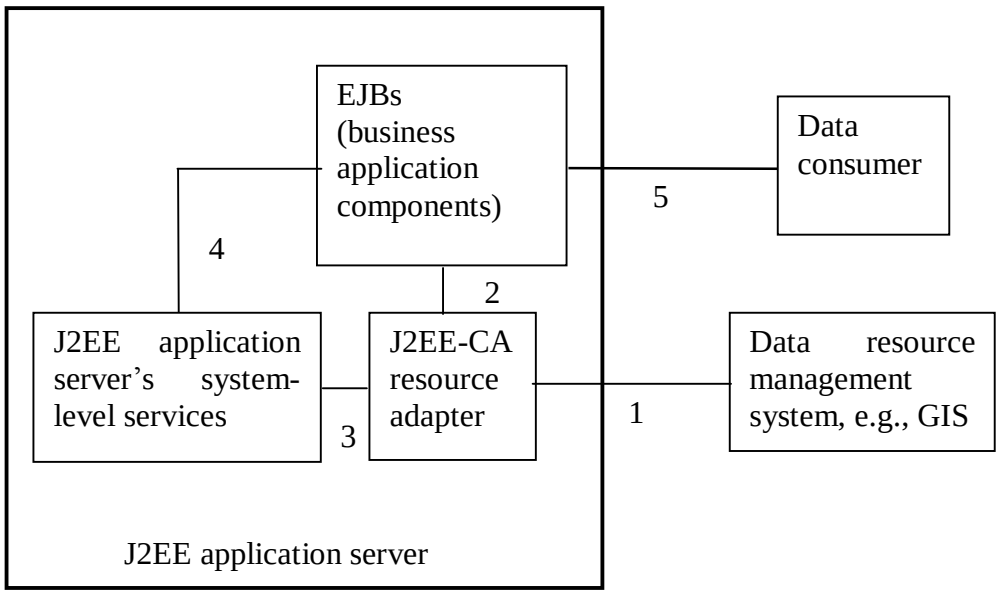


Figure 3: Interactions between the J2EE Connector Architecture components

A J2EE-CA adapter must communicate with three parties, the data resource, the client applications, and the J2EE server shown in Figure 3 [21]. The adapter must be able to access the data resource using the methods provided by the data resource software such as a GIS product. Typical COTS GIS database software provides some application programming interfaces (API). These APIs give the methods for the adapter to access the data resource (Link 1 in Figure 3). The business applications must be able to access the data resource through the adapter. The J2EE-CA recommends a set of optional common client interfaces (CCI). In our implementation, we let the business applications request data by XML messaging (Link 2 in Figure 3). These two aspects are common to all kinds of adapters. The additional requirement for the J2EE-CA adapters is the system contracts (Link 3 in Figure 3). The system contracts consist of three sets of Java interfaces for connection management, transaction management, and security management. These interfaces are not called by the client applications but by the J2EE application server. By

implementing these interfaces, the adapter submits the control to the J2EE application server. Every J2EE-CA adapter must be deployed in the J2EE application server. Typically, adapters are accessed by the business EJBs. Link 4 in Figure 3 indicates that all the EJBs are managed by the system-level services, too. External data consumers such as web servlets can access the adapter through an EJB or directly (Link 5 in Figure 3).

After we succeeded in the (image) adapter, we are now ready to layout the architecture for developing the enterprise geographic information system, in which many other GIS applications will be able to share the database. A true enterprise geographic information system must also be able to integrate itself into the enterprise business information systems. This trend has been recognized by Intergraph who has responded to this requirement in their latest GeoMedia release [24]. As described in Chapter 2, the GIS applications at the Army Corps is so much diversified that the need for a GIS services manager is justified. Since the Army Corps anticipates a platform-neutral, open environment, we have chosen a J2EE application server to host the GIS services manager shown in Figure 4. This is a high-level architecture of the Enterprise GIS. Since we want to leverage the existing COTS products' capacities to the full extent, the GIS services manager does not intercept the connections between the COST components and the database. On the other hand, the manager facilitates data accesses for the other applications such as the SWMM analyzer and LIDAR data rendering applications mentioned in Chapter 2. Compared to Figures 2, there are many additional components in Figures 4.

First, the "other applications" are considered in this architecture. These applications may request any kind of GIS data and non-GIS data. Consequently, two

types of adapters are introduced. They are the GIS data adapter that fetches data from the ArcSDE server and the standard JDBC adapter facilitates accessing non-GIS data. Finally, a J2EE application server along with a companion web server are utilized to manage a set of Enterprise JavaBeans (EJB) that encapsulate the business rules, as well as three types of adapters including the GIS data adapter, the JDBC adapter, and the image adapter (providing the same function as the adapter in Figure 2). Since the "other applications" vary widely, we let them access the GIS information in a very flexible protocol, HTTP. To shield the details of the data resources, "other applications" access the Enterprise GIS through servlets, which in turn communicate with "session enterprise beans" [20] (represented by item "EJB" in Figure 4). The explanations for the connections are listed in Table 1.

Specifically, the connection pooling service is particular handy for our system. Considering the development of the Enterprise GIS, the more enrichment of the GIS applications and web publishing, the more Army Corps' users and outside entities will use the system. As a result, the backend system may be overwhelmed. Experience shows that the most fragile spots are not in the database but at the connections between the data resources and the applications [21]. For example, even though the adapter we made in Figure 2 works fine with a few concurrent requests, it can be overwhelmed by many simultaneous requests long before ArcSDE is drawn. As another example, when many current requests coming from the "other applications", multiple JDBC connections will also be needed.

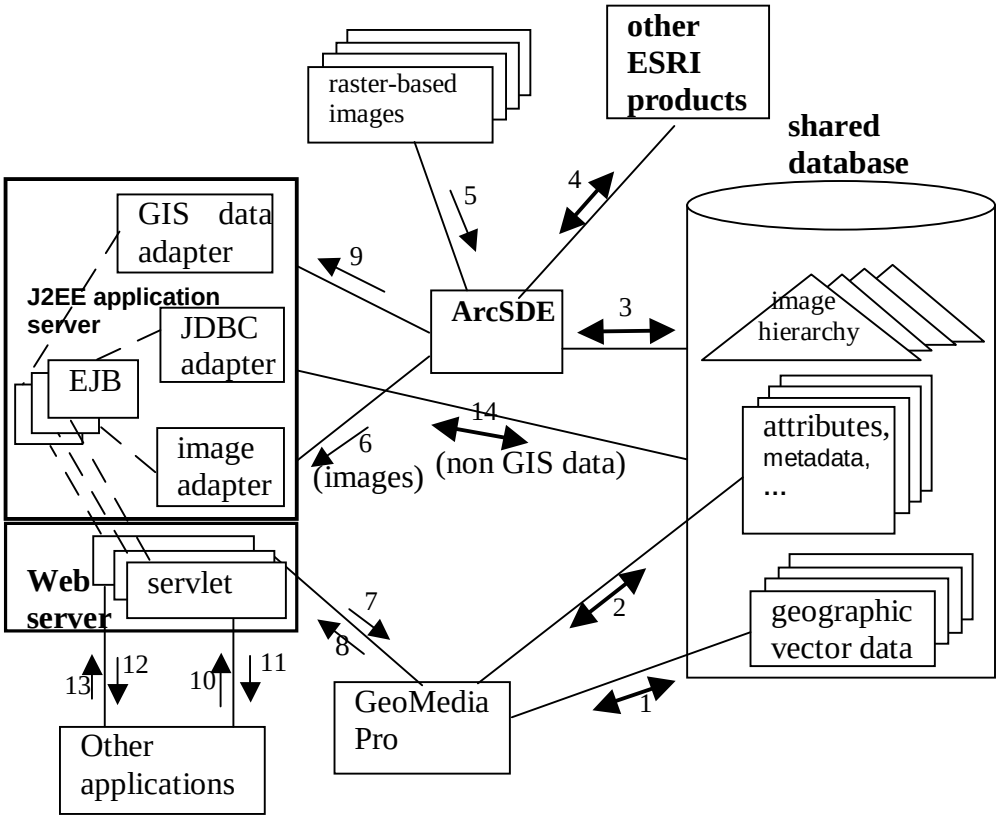


Figure 4: A Sketch of the Enterprise GIS

The solution is to have a pool of connections. As described above, J2EE application servers have built-in connections pooling capability, and the adapter must fulfill the J2EE-CA system service contracts. We have built a J2EE-CA adapter based on the adapter described in Section 3.2.2. Among the J2EE-CA system contracts, we have implemented the interfaces for the connection management. Even though the other interfaces for the transaction management and security are left blank. The adapter could be deployed into the J2EE application server and perform. Although all the vector data and attributes in the GIS are accessible using the JDBC adapter, this kind of low-level operation cannot sustain possible changes in the COTS products upgrading. For instance, the vendor can slightly change the internal database table structures in their future release

in order to improve or enhance something. Then our integration would be broken. Thus, the "other applications" access GIS data through the GIS data adapter which in turn connects to the ArcSDE server using the ArcSDE API. In order for these two non-JDBC adapters to submit the control of themselves to the J2EE application server, they have to comply with the J2EE Connector Architecture and they must reside in the J2EE application server. As a result, GeoMedia uses the image adapter through a servlet, servelet-Image, in HTTP.

Currently, we are in the process of prototyping this services management system using the J2EE application server technology.

No.	*explanation*	*protocol*
1	GeoMedia accessing geographic data	Oracle Net8
2	GeoMedia accessing tables holding attributes and metadata	Oracle Net8
3	ArcSDE's database connection, it may access every GIS information in the database.	Oracle Net8
4	ESRI products communicate with ArcSDE	TCP
5	Loading raster-based images into ArcSDE	(local)
6	Image adapter fetches images from ArcSDE	TCP/IP
7	GeoMedia receives images from image adapter	FTP
8	GeoMedia sends requests to image adapter	HTTP
9	GIS data adapter fetches GIS information from ArcSDE	TCP
10	Other applications receive non-GIS information from servlet-non-GIS	HTTP
11	Other applications sends requests to servlet-non-GIS	HTTP
12	Other applications receive GIS information from servlet-GIS	HTTP
13	Other applications send requests to servlet-GIS	HTTP
14	J2EE application server's JDBC connection to the shared database for non-GIS data	TCP

Table 1: Communications between Components of the Enterprise GIS

3.4 Internet Map Services

The tremendous growth in Internet use has resulted in an increased demand for the delivery of geographic data, maps, and applications over the Internet. Many leading GIS vendors developed their web solution and add it into their GIS product family, such as Intergraph's GeoMedia Web Map and Esri's ArcIMS, to meet this growing demand.

3.4.1 Create and administer Web sites using ArcIMS

To support spatial data web publish for USACE, we deplyed ArcIMS (Internet Map Service) from ESRI product family. ArcIMS is a GIS solution which allows us to centrally build and deliver maps, data, and tools over the Internet. ArcIMS takes advantages of the Internet technology that makes it possible to share information and data with many users, either locally or around the world. ArcIMS is also a flexible and scalable tool to publish maps and develop applications based on the complexity of the needs.

ArcIMS runs in a distributed environment and consists of both client and server components as shown in Figure 5. The ArcIMS HTML Viewer and ArcIMS Java Viewers are clientside components. On the server side, the ArcIMS server can provide three different types of services. They're Feature Service, Image Service and Metadata Service.

The ArcIMS server can accept map request from Internet client. The IMS server forwards the request to ArcSDE server that will in turn access the geo-spatial database and retrieve the spatial data of interest. The spatial dataset is transmitted back to the IMS server in a reverse direction and the IMS server will serve the client with a map.

ArcIMS provides four applications to help us create and administer web sites. There are ArcIMS Manager, ArcIMS Author, ArcIMS Administrator, and ArcIMS Designer. ArcIMS Manager is a Web-based application that supports the three main tasks in ArcIMS—map authoring, Web site design, and site administration. These tasks can also be completed using the other three applications.

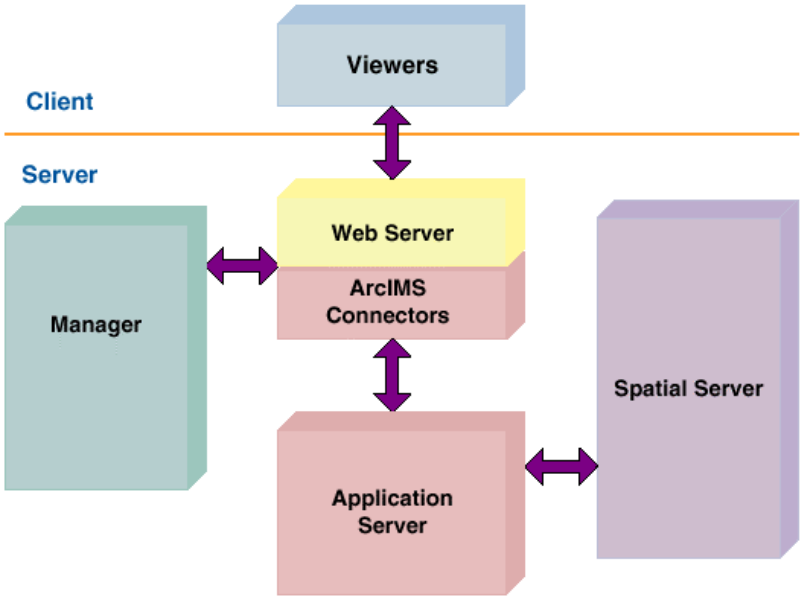


Figure 5 : ArcIMS Architecture

3.4.2 Web-based spatial data search on arbitrarily organized file system

At USACE, most data are stored in the different file systems and different computers. It is not easy for administrator to manage them. They need some approaches for administrating the data in the file system. We developed a Web-based map search tool for them. In this application, we take advantage of the HTML viewer of ArcIMS which is a lightweight viewer employs JavaScript and Dynamic HTML. Java Servlet technology was used to query metadata stored in database.

ArcIMS features largely in this approach. The ArcIMS Spatial Server is the backbone of ArcIMS. It processes requests for maps and related information. When a request is received, the ArcIMS Spatial Server performance one or more of these functions shown in Figure 6:

- Image—creates image files from maps
- Feature—streams map features
- Query—searches for features matching search criteria
- Geocode—performs address-matching operations
- Extract—creates shapefiles from selected map features
- Metadata—publishes metadata
- Route—caculates routes between a set of two or more stops

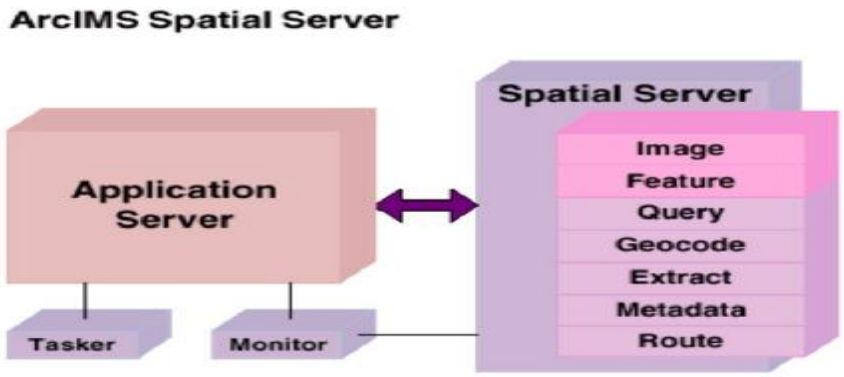


Figure 6: ArcIMS Spatial Server

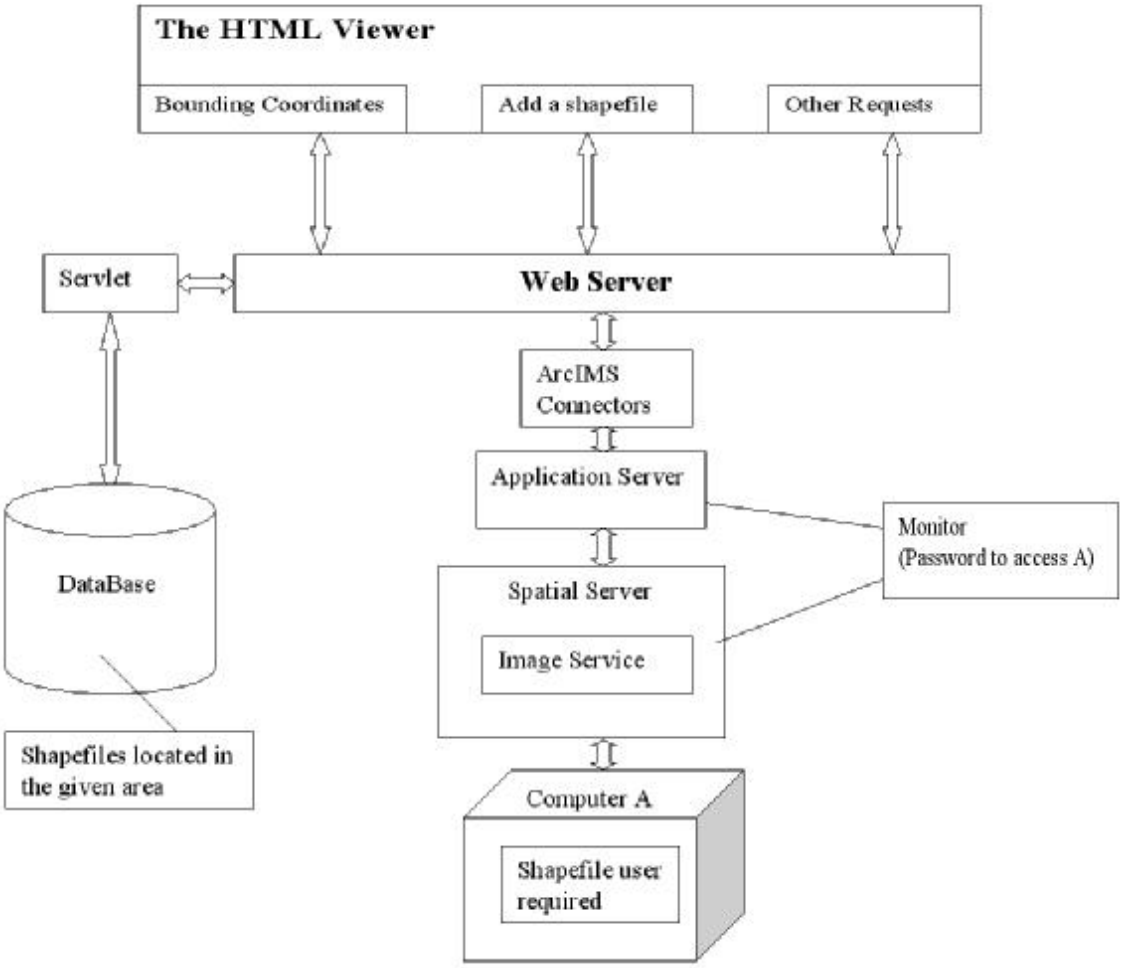


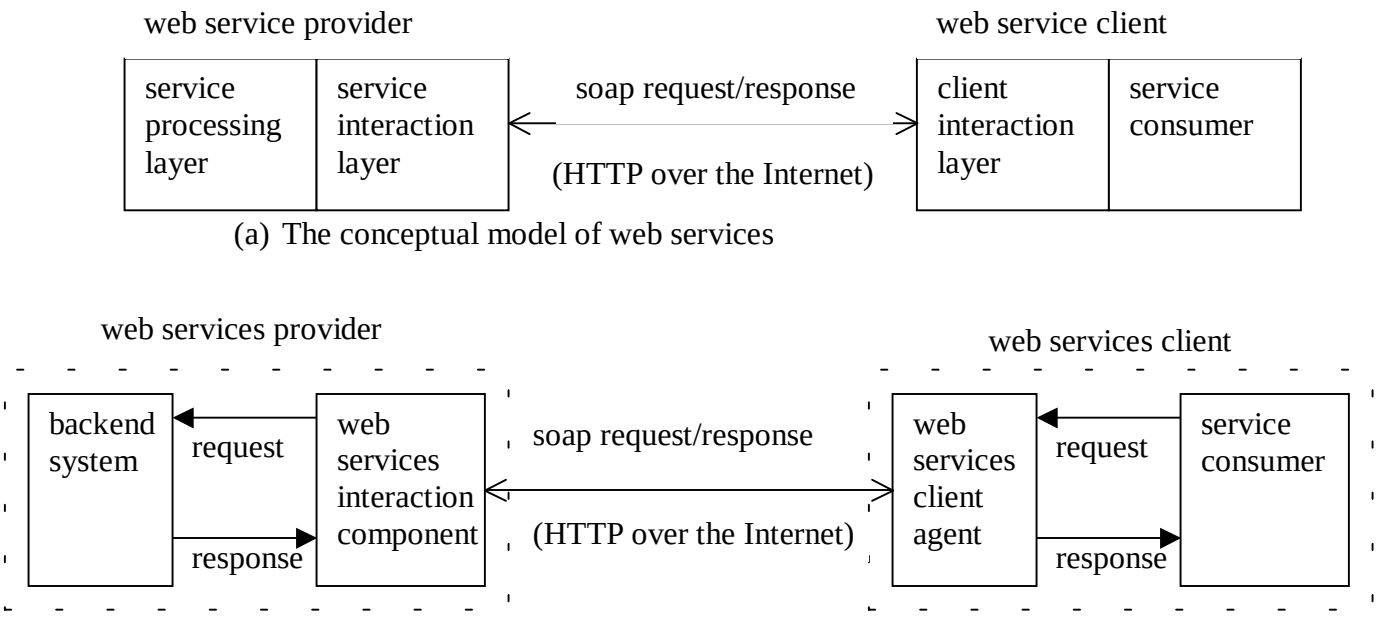
Figure 7: The architecture of web-based map search tool

The architecture of our application shows in Figure 7. The bounding coordinates are sent to a servlet which will retrieve the data stored in database and get a list of maps stored in the file system and located in the area we specified. The client can choose maps from the list in HTML viewer. The viewer send the client's request and ask ArcIMS add the selected map onto the background as a dynamic layer. ArcIMS processes the request and gets the information from the computer where the chosen spatial files are stored, assembles them into the Image Service. Then retunes a new "snapshot" of map to the client.

3.5 SOAP-Based Web Services Framework

In the following research, we extended our previous design by introducing Web Service technology so that MicroStation or other GIS application in a distributed environment can also access spatial data through ArcSDE. In effect, web Services represent a widely accepted point-to-point communication protocol that connects a pair of computational parties shown in Figure 8 (a). For nontrivial systems, behind each web service is a backend system that fulfills the task. In front of each web service client is typically an application that consumes the service. This application can be as simple as a web browser or as complex as another GIS or CAD application suite. Figure 8 (b) depicts a conceptual structure of such a web services system. The “service consumer” and the “web services client agent” components are in the same local area network (LAN) or even in the same computer; they may communicate in TCP. In our experiments, we used Java RMI. Similarly, the backend system and the web services interaction component typically locate in the same LAN. However, between the web services provider and the web services client, we can only assume the SOAP/HTTP protocol because different departments and external clients are to be involved in the system.

Conceptually, the web services provide us with another mechanism to construct distributed computing systems. Many design principles for distributed software are applicable to the design of web services, but also are more crucial to apply to the GIS web services.



(a) The conceptual model of web services

(b) The web services model that includes the backend and front-end

Figure 8: The web service model

Compare to general purposed web services, GIS web service requires more careful design due to at least three factors. First, services provided by GIS system typically requires heavy CPU utilization for complex computation. Second, GIS services are often heavy-duty software tools that traditionally stand-alone. Third, their moving towards the web service era is lagged. For enterprise GIS, successful communications between components are essential but not adequate. Performance should always be a pivot consideration in the design for GIS web services system.

This wrapper web service is going to be developed by Java language and using SOAP message to stream the output image data. The Java web service is going to access a DLL library using the JNI (Java Native Interface). The DLL includes the methods that can access the spatial dataset in the database through SDE server. There are only three methods in the DLL, one is for establishing a connection to SDE server, the other is for

disconnect. The main method is the getRasterFile(), this method will be designed to return a array of bytes to the caller. The web service will use this byte array to construct the SOAP message.

3.6 A Hybrid Web Service Architecture

Over years, the software architecture of the E-GIS has evolved into a hybrid web service architecture shown in Figure 9. In this architecture, the essential principle is to consolidate all spatial data sets into a shared spatial database (an Oralce database in particular). Enhanced by the ArcSDE Server, the raster images can also be stored in and managed by the database. Our adapter allows us to connect the ArcSDE server to web Service clients, in the standard SOAP protocol. By accommodating both the COTS Internet Map Service software ArcIMS and SOAP-based web services in the web server, we have not only full leveraged the features of ArcIMS, but also supplemented the COTS with more flexible web service provider components. They can be used by both external business partners and the other software tools that are not capable of accessing OGC standard spatial data or ArcSDE images. When these web services are popular, the management of the back-end server software must be scalable. Therefore, we have introduced the J2EE server into the architecture in order to enhance the back-end service performance. We believe this hybrid architecture can endorse performance, scalability and flexibility, while minimizing software development efforts by using COTS products.

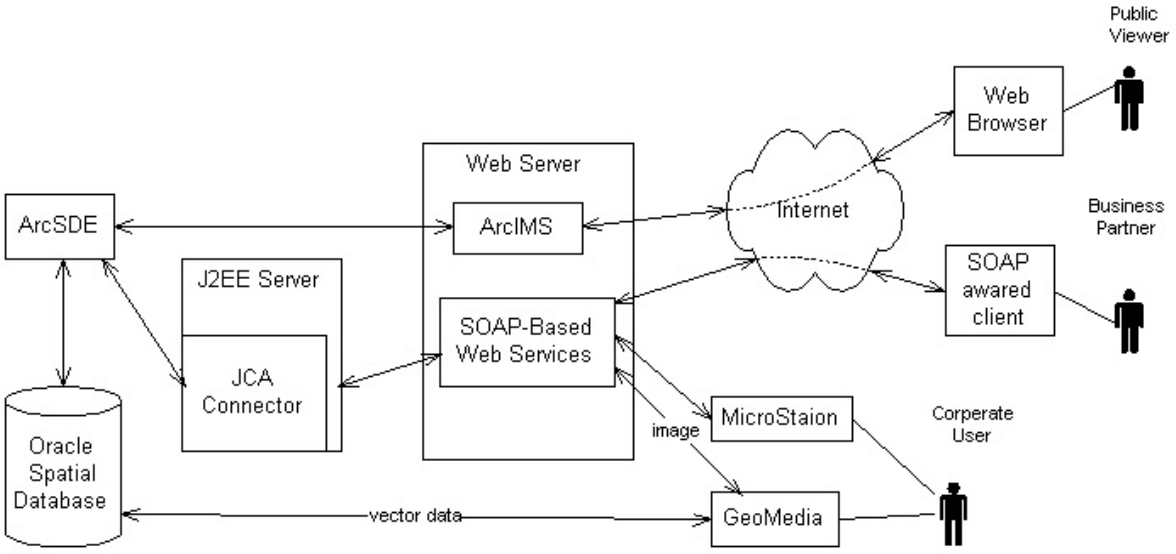


Figure 9: The integration of COS GIS System

CHAPTER 4

IMPLEMENTATION

Our approach to enterprise GIS system is to put spatial data into a centralized database system and make the dataset sharable across heterogeneous GIS desktop software. We have implemented the vector data integration using Oracle spatial technology. In our system, the raster data sets are also saved in the Oracle database. ESRI Company provides raster data import and export functions to its own desktop software. In order to support raster dataset sharing for non-ESRI client, we developed an adapter that conveys the raster dataset import and export functionality of ArcSDE server in a set of interfaces for external components. The adapter was developed using C language and encapsulated into dynamic link library package for deployment in windows system. The client software can invoke the functions through any programming environment that recognizes DLL packages. In our implementation, the DLL package was developed mainly for calls from VB and Java language, the most common used development languages for both ESRI and Intergraph.

4.1 Technologies Review

4.1.1 ArcSDE Server and development API

ESRI enterprise GIS solution is based on a centralized geo-spatial database. To facilitate spatial data sharing and management, the solution include SDE (Spatial Data Engine) server that acts as an application server. SDE server is responsible for receiving the spatial data request and serving the client request with spatial data accordingly. SDE server also maintains the data integrity, manage the transaction and tune the overall performance of the spatial data service. The ArcSDE server also provides a set of API for customized application development. ArcSDE contains two main application programmer interfaces for developers to build applications that work with and query information contained in multi-user geo-databases: ArcSDE Client API for C programmers and ArcSDE Client API for Java programmers. These APIs provide GIS functions for advanced application development. ArcSDE works as an application server, delivering spatial data to many kinds of applications or even serving spatial data across the Internet.

4.1.2 ArcSDE raster adapter and JNI

ArcSDE raster adapter is developed in the EGIS system to support non-ESRI GIS application to access and query raster-based spatial information served by the ArcSDE server. We choose to use the ArcSDE high-level C API for the development based on two facts. First, at the time of our development, ESRI hadn’t release a full Java version API for the ArcSDE server, in order to take advantage of the raster data manipulation, we must use the C API as the primary development language. Second, the C API is the

native language of the ArcSDE server application that can achieve the best performance leverage the full capacity of ArcSDE server.

The raster adapter developed by C API is compiled into dynamic linked library in windows system for deployment. The raster adapter DLL package consists of generic subroutine for interacting with the ArcSDE server as well as application specific functions. Since, we are going to integrate the raster adapter into our EGIS based on J2EE framework, we need to address the issue of connecting the J2EE framework with ArcSDE server.

The Java Native Interface technology is employed for the interaction between the J2EE framework and ArcSDE server. The JNI comes with the standard Java Development Kit (JDK) from Sun Microsystems. It permits Java programmers to integrate native code (currently C and C++) into their Java applications.

4.1.3 TIFF LIB and GEOTIFF LIB

Tag Image File Format (TIFF) is a widely used format for storing image data. TIFF LIB is a set of C functions (a library) that support the manipulation of TIFF image files. The library requires an ANSI C compilation environment for building and presumes an ANSI C environment for use. Our ArcSDE raster adapter needs TIFF LIB to help generate TIFF image file as the output data format which can be consumed by non-ESRI client.

TIFF file is the mostly supported image format for GIS purpose. In most cases, geographical information is added to the TIFF to form so called geo-TIFF file. This kind of file has geographical information that been saved in the file tag header which can be

used to identify the coordinate system and location of the raster file. This is why the GEOTIFF LIB is employed to geo-reference a TIFF file.

4.1.4 Web services

A web service is a software component that can be accessed over the Internet for use in other application. Their application field can be wide. They are used in e-business, Geographical Information System, and many other application areas. Specifically, web services are a stack of emerging standards that describe a service-oriented, component-based application architecture.

The main interest of web services is that they can be accessed programmatically by HTTP protocol. Unlike web sites and desktop applications, web services are not designed for direct human interaction, and they do not have a graphical user interface. Rather, web services operate at the code level; they are called by and exchange data with other software. Web services can be incorporated into software designed for human interaction.

For example, GIS web services offer a way for developers to include GIS content and capabilities in their application without having to host the data. The result is significant savings of development time, expense and computer resources.

By web services, we mean XML web services, the software services exposed on the web through SOAP. SOAP is an XML-based communication protocol and encoding format for inter-application communication. Originally conceived by Microsoft and Userland software, it has evolved through several generations and the current spec, SOAP 1.1, is fast growing in popularity and usage. The W3C's XML protocol working group is in the process of turning SOAP into a true open standard, and as of this writing has

released a working draft of SOAP 1.2, which cleans up some of the more confusing areas of 1.1 spec. SOAP is widely viewed as the backbone to a new generation of cross-platform cross-language distributed computing applications, the web services.

4.1.5 ArcIMS and ArcXML

To support spatial data web publish, ESRI product family also include an IMS (Internet Map Service) server. The IMS server can accept map request from Internet client. The IMS server forwards the request to ArcSDE server that will in turn access the geo-spatial database and retrieve the spatial data of interest. The spatial dataset is transmitted back to the IMS server in a reverse direction and the IMS server will serve the client with a map.

The ArcIMS server itself has a built-in map request language based on XML called ArcXML (The Arc Extensible Markup Language). ArcXML file format provides a structured method for communication between all ArcIMS components. It defines content for ArcIMS services and is used for requests and responses between clients, the business logic tier and servers.

4.2 Use of the Shared Spatial Database

To achieve the vector data sharing for GeoMedia, we employed Oracle Spatial technology shown as Figure 10. Oracle spatial is a well-defined, object-oriented industry standard on vector-based data storage and is supported by most GIS vendor. Both ESRI and Intergraph give support to Oracle Spatial technology. Arcinfo and Geomedia can both directly access the vector data that stored in Oracle spatial. Oracle spatial is the major integration point of the whole system.

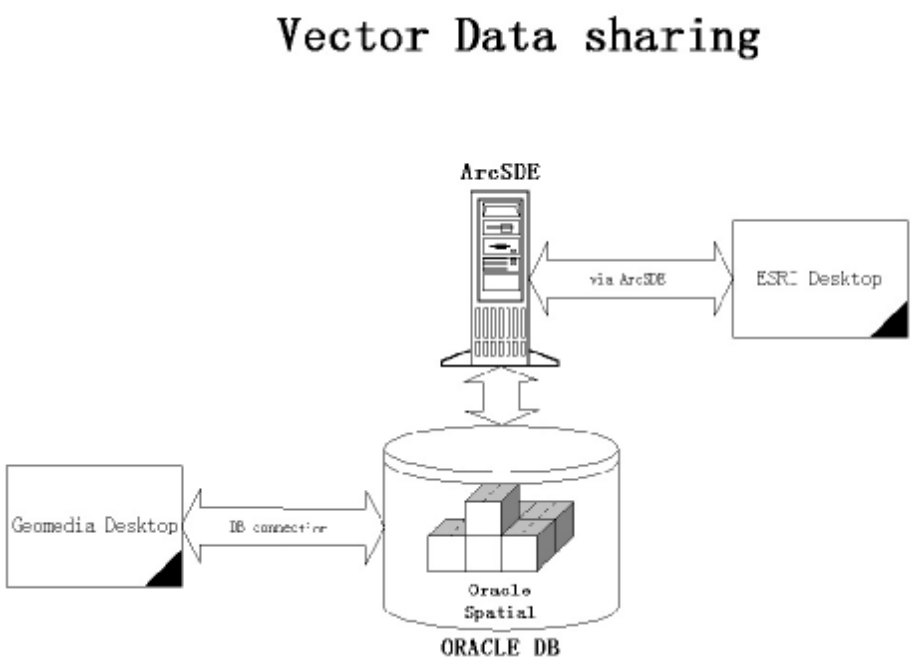


Figure 10: Vector Data sharing for GeoMedia

Database accessing performance has been one of our concerns. More specifically, our concern is about the impact of crossing accesses. That is, one GIS tool accesses the data imported by another one. Both Geomedia and ArcSDE create spatial index when they import vector data sets into the Oracle database. ArcSDE also provides several options for spatial index creations that can be specified by the users. Different spatial index methods may affect the spatial data query performance. In an initial evaluation, we used ArcMap to query vector data through the ArcSDE server from the Oracle database, and compared the response time of accessing the "native data sets" (imported by ArcSDE) and that of accessing the "foreign data sets" (imported by GeoMedia) as shown in Table 2. In testing, the query for every data set was to load the entire layer. No significant performance penalty to the foreign data sets was observed. This confirms the feasibility of our data consolidation plan. We also realized that the initial test was rudimentary. Indeed, performance is related to the nature of queries and the type of

indexes. Additionally, ArcSDE has a comprehensive performance tuning guide. Therefore, further studies on the impact of crossing accesses to query performance are needed.

Data set name	Geometry type	Features Count	Imported by ArcSDE (second)	Imported by Geomedia (second)
Barge	Multipoint	778	2	2
Powerlines	Polyline	9226	3	3
PrimaryRoad	Polyline	1633	3.5	4
Aquifer	Polygon	2388	11	13

Geometry storage type: Oracle Spatial Object Model
Spatial Index: true
Open connection delay: 3 seconds

Table 2: Accessing time comparison between native and foreign data sets

4.3 ArcSDE Adapters: Raster Adapter and Vector Adapter

4.3.1 ArcSDE Raster Adapter

We developed software component to support raster data retrieval from ArcSDE server. ESRI Company provides raster data import and export functions to its own desktop software. In order to support the raster data sharing for non-ESRI client, we have to develop a middleware. This middle-ware is to wrap the raster dataset import and export functionality and provide a set of interfaces for them. This middleware is called "ArcSDE Raster Adapter".

The adapter is developed using C language and encapsulate into DLL package for deployment. The client software could take advantage of the DLL by invoking the functions through any programming environment that could manipulate DLL package. Here the DLL package is developed mainly for call from VB language. It can also be invoked by Java language through the Java Native Interface.

As shown in Figure 11, the Geomedia desktop software sends request through the adapter. SDE adapter will first make connection to the oracle server through SDE server and export the raster data into a Tiff file according to the given bounding coordinates. There are two issues must be addressed for the raster adapter. One is that we need to add geographic information to the tiff file to generate geo-tiff file. The other is the resolution control problem. To enhance display performance, the ArcSDE server has built pyramid for raster dataset. Given bounding coordinate, we can return the required image data with different resolution to suit to the user's display resolution.

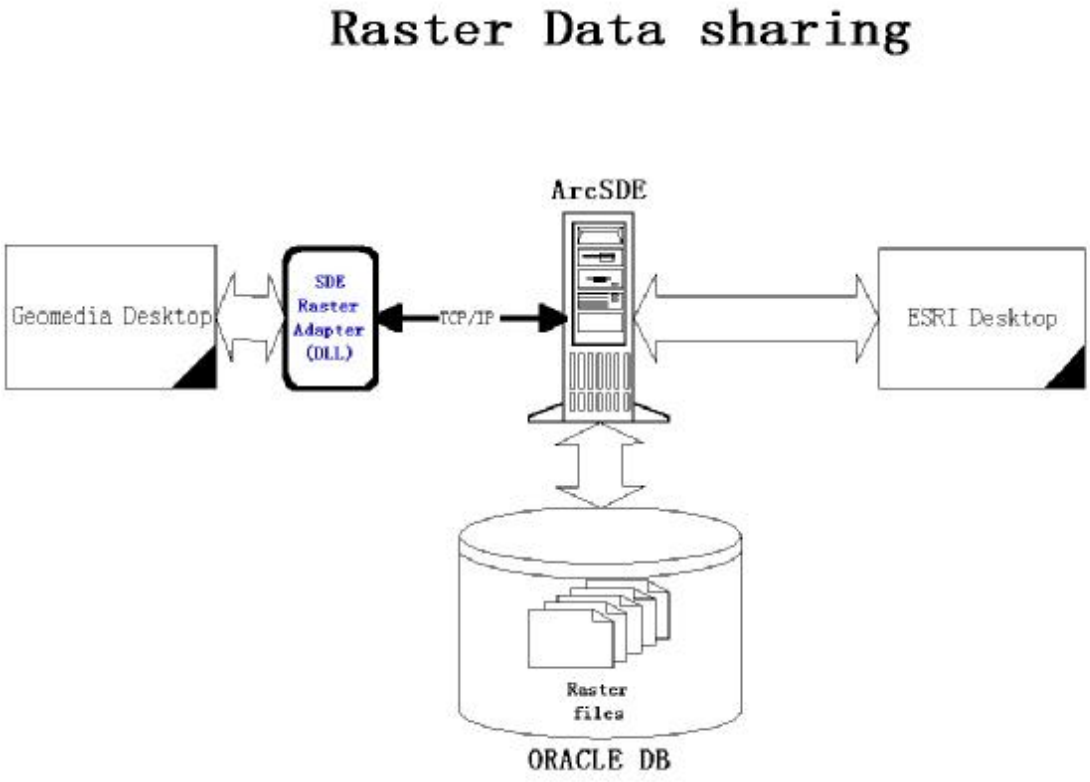


Figure 11: Raster data sharing for GeoMedia

4.3.1.1 Coordinate System

A coordinate system (CS) defines the location of a point on a planar or spherical surface. Without a concise definition of the coordinate system used to represent a point, it

is difficult to change to a new coordinate system. Changing coordinate systems is required when integrating data from different sources into a common coordinate system.

A geographic coordinate system is a three–dimensional reference system that locates points on the Earth's surface. The unit of measure is usually decimal degrees. A point has two coordinate values: latitude and longitude. Latitude and longitude measure angles. Even though geographic coordinates are angular units, ArcSDE stores and treats them as if they are planar. In this case, longitude values are considered the x-coordinate, while latitude values are the y-coordinate.

A projected coordinate system is a two-dimensional planar surface. However, the Earth's surface is three-dimensional. Transforming three-dimensional space onto a two-dimensional surface is called projection. Projection formulas are mathematical expressions that convert data from a geographical location (latitude and longitude) on a sphere or spheroid to a corresponding location (x and y) on a flat, two–dimensional surface.

In order to display the geographical information on a monitor of computer, the spatial object are represented as simple or complex geometries and displayed on the computer monitor. Thus, we also need to introduce a raster space coordinate system to help define how a geographical CS or Projection CS are mapping to the computer output devices.

The Figure 12 below shows the relationship of the three coordinate systems:

- Raster Space CS: Raster space Coordinate System which unit is pixel
- Projection CS: Projection Coordinate System which unit is meter or foot
- Geographic CS: World Coordinate System which unit is degree

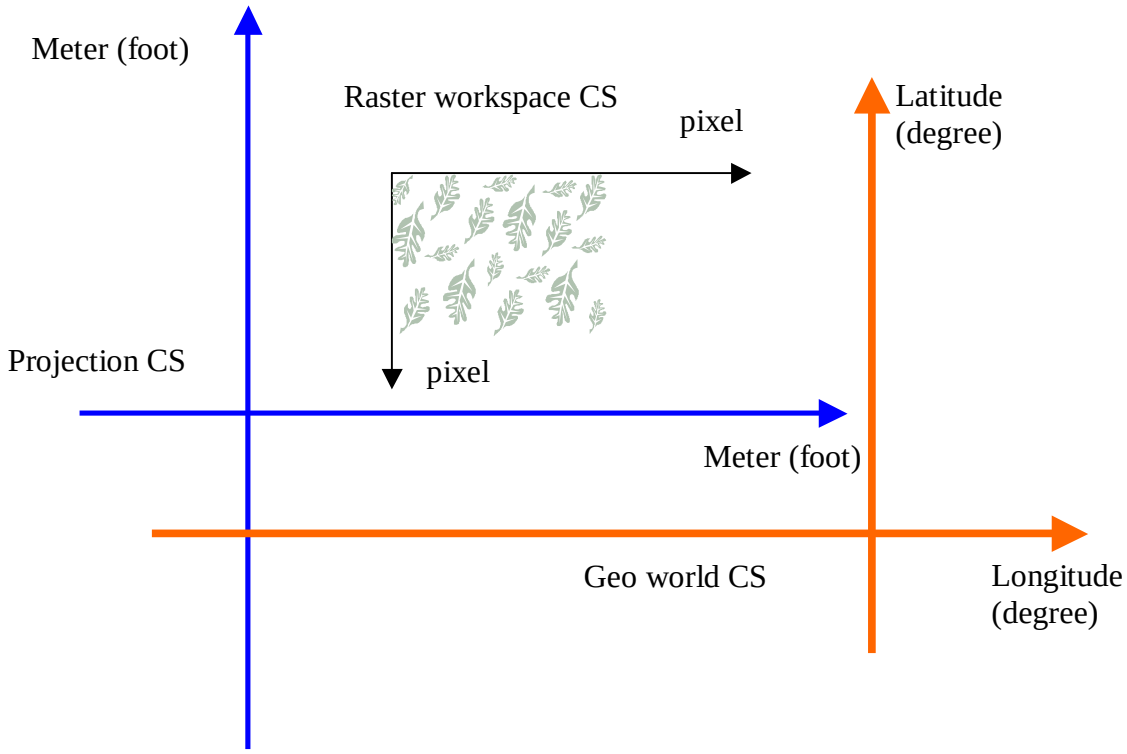


Figure 12: the relationship of the three coordinate systems

4.3.1.2 Resolution level

ArcSDE populates the raster blocks table according to a declining resolution pyramid. The height of the pyramid is determined by the number of levels specified by application. The application such as ArcToolbox or ArcCatalog may allow you to define the levels or it may request that ArcSDE calculate them, or it may offer both possible choices.

The pyramid begins at the base, or level 0 which contains the original pixels of the image. The pyramid proceeds toward the apex by coalescing four pixels from the previous level into a single pixel at the current level. This process continues until less than four pixels remain or until ArcSDE exhausts the defined number of levels. The apex of the pyramid is reached when the uppermost level has less than four pixels.

The additional levels of the pyramid increase the number of raster block table rows by one third. However, since it is possible for the user to specify the number of levels, the true apex of the pyramid may not be obtained, limiting the number of records added to the raster blocks table.

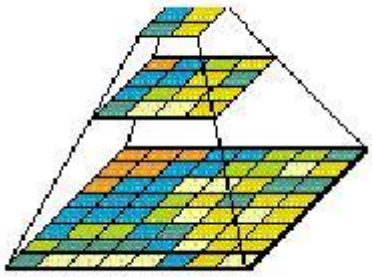


Figure 13: Raster pyramid in ArcSDE

When you build a pyramid, more rasters are created by progressively downsampling the previous level by a factor of two until the apex. As the application zooms out and the raster cells grow smaller than the resolution threshold, ArcSDE selects a higher level of the pyramid. The purpose of the pyramid is to optimize display performance.

The pyramid allows ArcSDE to provide the application with a constant resolution of pixel data regardless of the rendering window's scale. Data of a large raster transfers quicker to the client when a pyramid exists, since ArcSDE can transfer fewer cells of a reduced resolution.

In our raster adapter, we provide the users a functionality to choose the resolution level by themselves or automatically by the adapter shown in Figure 14. In order to implement selecting reasonable level automatically and enhance the performance, there is

internal method called getLevel in our raster adapter. We use the following formula to calculate the level we will select:

$$level = \max[\log_2(lc/ls), \log_2(wc/ws)]$$

lc , length of the selected clip

ls , length of the screen resolution

wc , width of the selected clip

ws , width of the screen resolution

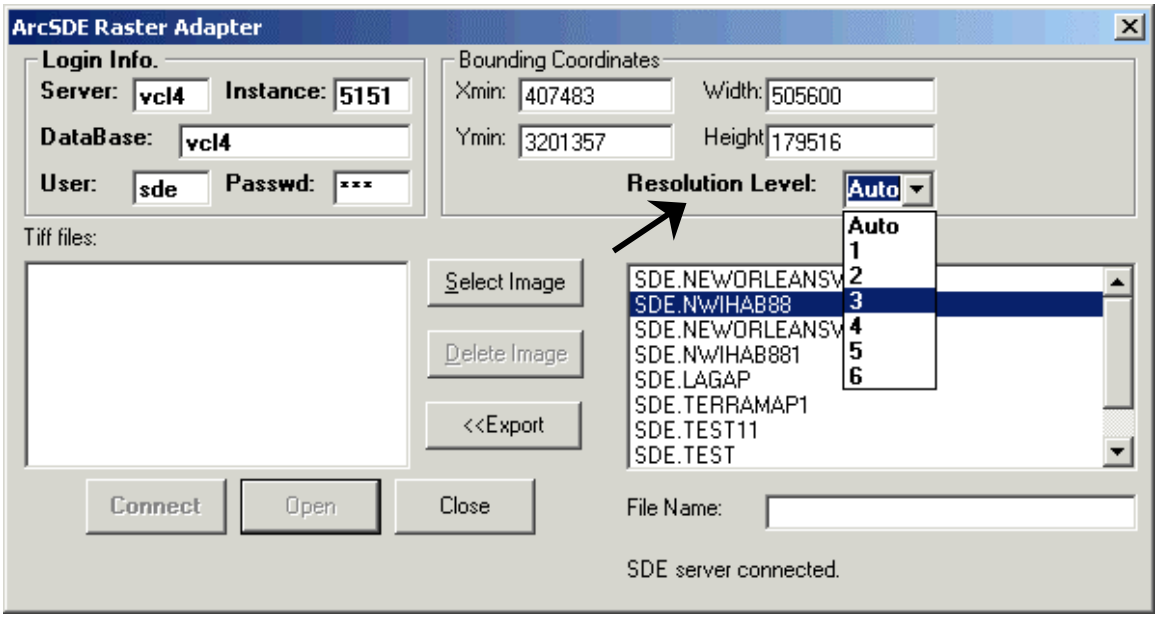


Figure 14: select resolution level in raster adapter

4.3.1.3 Functions provided by the C DLL

We have developed two version of raster adapter. One version is designed to return a single tile each time for end user's request. This solution has more advantages over the other version in that it provides more flexibility and achieves high performance by making the system scalable. This version can be easily migrated to Internet based

application. In our project, we set up the raster adapter as Web service and server end user's request through the standard Web Service interface.

The other version is designed for GIS desktop customized application. This version uses the client-server model to communicate with the ArcSDE server. The communicate takes advantage of the TCP/IP protocol to minimize the network communication overhead compare to the previous version. The raster adapter will be installed at the client desktop and act as an agent for the ArcSDE server. End user's request will be fulfilled and raster dataset will be returned as soon as the raster adapter agent finish assemble the spatial information into a raster file, there is no need for the end user to explicitly control the raster data retrieval process.

Function Name	**Explanation**
SdeConnect, sdeDisconnect	Establish and release the ArcSDE connection
getRasterFile(int prj_x_min, int prj_y_min, int prj_width, int prj_height, char* tablename, int selected_level)	The main function invoked by end user
getRasterInfo(char* tablename)	Look up the raster name in ArcSDE server and return metadata about the raster data
getRasterColumnInfoList()	Get additional information
int getColormapImage(int x_min, int y_min, int x_max, int y_max, char* tablename, int column_id, int band_id, int selected_level)	Use the collected metadata information and selected area information to get raster data for those who use color map
int getRGBImage(int x_min, int y_min, int x_max, int y_max, char* tablename, int column_id, int band_id, int selected_level){	Use the collected metadata information and selected area information to get raster data for those who use RGB raster band
long getPrjHeight() long getPrjWidth() long getMaxLevel() long getPrjX() long getPrjY() long getImageWidth() long getImageHeight() char * _stdcall getPCSString()	Functions used to deliver raster metadata information back to user
int getLevel(int xmin,int ymin,int xmax,int ymax)	Function that calculate the best resolution level for the current user request

Table 3: Functions provided by raster adapter

The basic functions of the first version adapter have been implemented. In Figure 15, we list a minimum subset of functions provided.

1. **Name:** sdeConnect ()
   Connect to the SDE Server with provided connection parameters.
   Usage syntax:
   int sdeConnect (char* sdeserver, char* sid, char* db, char* usr, char* pwd);
   Parameters:
   ```
   sdeserver:"XXX" (SDEserver name)
   sid: "5151" (SDE server port)
   db:   "XXX"  (Database SID)
   usr:  "xxx"  (SDE server user)
   pwd:  "xxx"  (SDE server password)
   ```
   Description:
   This function is to initialize a connection to the SDE server.
   Returns:
   Return 0 if success.
2. **Name:** getRasterFile()
   Get the max resolution level of the current image
   Usage syntax:
   Char[] getRasterFile (int x_tile_num, int y_tile_num, int selected_level);
   Parameters:

X_num	the tile number on the x coordinate
Y_num	the tile number on the y coordinate
Selected_level	the resolution level specified by the user. The user will select the level by choosing from a value list. When Selected level equal to –1, the selected level will be created automatically.

   Description:
   By entering the x, y tile number and resolution level to retrieve the raster dataset of a particular tile. To enhance display performance, we can build resolution pyramid for raster dataset. We can return the required image data with different resolution to satisfy the user's needs by setting the selected_level parameter.
   Returns:
   Return a char array contains the value of image dataset. Since it is RGB file, the array include 3 bands, each tile contains 128*128 pixels, thus the size of this array is 3*128*128
3. **Name:** sdeDisconnect()
   Disconnect from the SDE server.
   Usage syntax:
   int sdeDisconnect();
   Description:
   This function is to disconnect to the SDE server.
   Returns:
   Return 0

Figure 15: Basic Functions Provided by Raster Adapter

For the second version of raster adapter, we only list the functions provided and give explanation in table 3.

4.3.1.4 Implementation and performance analysis

We consolidate raster-based images using ArcSDE server. ArcSDE has APIs in Java, C and Visual Basic to access geo-database through ArcSDE. Although ArcSDE's Java API has not been completed by ESRI, ArcSDE's C API has been completed. It is truly a powerful API that supports both vector and raster data creation, management and query. Our raster adapter makes call to ArcSDE's C API. Using this API, we can query the selected image and its companion color map as well as other attributes. These functions are essential for raster file reconstruction, when we want to export the raster data to image files.

- **How the raster adapter works**

The raster adapter is designed to be flexible enough to user's request. Upon the connection to the ArcSDE server, The user will be provided by a list of the available raster image dataset names in the geo-database as well as metadata of each raster dataset like the size of the raster data by pixels and available resolution levels of this dataset as shown in Figure 14.

After given the information about the raster data, the user can request the spatial information they need. The input value for the raster adapter consists of five parameters. They are the coordinates of the upper left point of the requested area, the output image size specified by width and height and the resolution level that decide the display quality.

If the end user doesn't specify the output resolution level, Raster adapter will use an algorithm describe in section 4.3.1.2 to help decide which resolution level will be used. The raster adapter will try to produce an output area that fits the end user's display device with the best resolution level.

Table 4 provides the workflow of the getRasterFile function in the Raster Adapter.

No	Workflow	ArcSDE API
1	Create a stream for query	SE_stream_create
2	Allocate an SE_SQL_CONSTRUCT, query information is encapsulated in this structure	SE_sql_construct_alloc
3	Define the stream query	SE_stream_query
4	Fetch the first row	SE_stream_fetch
5	Obtain the image metadata in the current row	SE_rasterattr_get_image_size SE_rasterattr_get_max_level
6	Prepare to get raster data	SE_rasconstraint_set_envelope SE_rasconstraint_set_level
7	Send query to ArcSDE server	SE_stream_query_raster_tile
8	Loop to retrieve raster data from the stream	SE_stream_get_raster_tile SE_rastileinfo_get_band_id SE_rastileinfo_get_level SE_rastileinfo_get_rowcol SE_rastileinfo_get_pixel_data
9	Free resource hold	SE_sql_construct_free SE_rastileinfo_free SE_rasconstraint_free SE_stream_free

Table 4: Work flow of the getRasterFile() function

- **Tiff Lib & GeoTiff Lib for geo-referenced raster file**

TIFF format images are commonly used by GIS experts. We used the Libtiff open source library [25] to export TIFF files. There are two ways to implement the color schema of raster file. Some raster file use color map to index the distinct color with an image and represent each image pixel by the color index. Other raster files use the RGB color bands to store the color information. Each band record only one of the RGB information.

We use the Tiff Lib API to output the raster data to a Tiff file. We also use the GeoTiff Lib API to register geo-information into the Tiff file. The geographical

information will be save as a set of tags and assemble with the raster data to create geo-referenced raster file.

Figure 16 is an example of the geographical information to be registered into Tiff file.

```
Geotiff_Information:
Version: 1
Key_Revision: 1.0
Tagged_Information:
ModelTransformationTag (4,4):
399     0    0    407483
0   -399     0    3380874
0       0       1        0
0       0       0        1
End_Of_Tags.
Keyed_Information:
GTModelTypeGeoKey (Short,1): ModelTypeProjected
GTRasterTypeGeoKey (Short,1): RasterPixelIsArea
ProjectedCSTypeGeoKey (Short,1): PCS_NAD83_UTM_zone_15N
End_Of_Keys.
End_Of_Geotiff.
```

Figure 16: An example of the geographical information to be registered into Tiff file

This file tells us the projection coordinate system used. The "RasterPixelIsArea" keyword indicates that each pixel covers a certain area in the image. The "ModelTransformationTag" keyword gives us a transformation matrix to calculate the accurate coordinate for each pixel in the raster image. Our raster adapter will create such a geographical transformation file and take advantage of the geotifcp utility tool provided by the GeoTiff Lib to help register the geographical information. The registration processes require the customized application to call shell command to access the geotifcp utility.

- **Raster adapter performance analysis**

Upon receiving a XML request message, the raster adapter fetches the requested image from the shared database through ArcSDE, and writes the result images as a file in a specified directory. The adapter may reside anywhere in the network. To avoid adding computation burden to the ArcSDE server, we let this adapter reside on the GeoMedia workstation. The adapter communicates with the ArcSDE server in TCP/IP. The explanations of the links between components are listed in Table 5.

Compared to using local image files, fetching images from the ArcSDE server remotely seems subject to performance penalty. To assure that the database access delay is in tolerable range, we carried out a test using a 10.3MB compressed remote-sensing image file in TIFF format. Its uncompressed size was 138.5 MB. Since GeoMedia accesses raster data by files, the initialization time for GeoMedia to access this image was 40 seconds. The same TIFF file was imported into the ArcSDE server that connects to an Oracle database 8i server. The raster data were automatically saved as BLOB column in the Oracle database by the ArcSDE server. We then used our adapter to load parts with different sizes of this image from the ArcSDE server. Table 6 lists the performance for the adapter to get the parts from the ArcSDE server (in the same LAN). In the right column, adapter response time was measured from the request to the file is completely written. The computers that ran the GeoMedia Professional desktop and the ArcSDE server have the same configuration (850 MHz CPU, 512 MB memory, and the same SCSI disk drives). This simple test convinced us that our adapter approach is practical.

No.	*explanation*	*protocol*
1	GeoMedia accessing geographic data	Oracle Net8
2	GeoMedia accessing tables holding attributes and metadata	Oracle Net8
3	ArcSDE's database connection, it may access every GIS information in the database.	Oracle Net8
4	ESRI products communicate with ArcSDE	TCP
5	Importing raster-based images into ArcSDE	(local)
6	Adapter fetch images from ArcSDE	TCP
7	GeoMedia receives images from adapter	(local)
8	GeoMedia sends requests to adapter	(local)

Table 5: Communications between components of the backend system

Image part size	Adapter response time
128*128	0.0945
768*768	0.026
1024*1024	0.375
20225* 7181 (full size)	35.0

Raster properties: ***Raster Dimension***: 20225 X 7181
Tile size: 128 X 128
Pyramids: True, 6 levels, nearest
Colormap: True
Data compression: LZ77

Table 6: Adapter accessing time

4.3.2 ArcSDE Vector Data Adapter

Since other GIS applications such as MicroStation cannot access spatial data stored as the type of Oracle Spatial Object Model so easily like GeoMedia. We developed a vector data adapter to support vector data retrieval through ArcSDE server. The vectors are a versatile and frequently used geographic data representation, well-suited for representing features with discrete boundaries such as wells, streets, rivers, states, and parcels. Typically, features are spatially represented as points, lines or polygons. ArcSDE stores vectors in a geodatabase which is a repository inside a DBMS and provides open, high-level C and Java APIs for querying and processing spatial information. Both C and

Java APIs provide plenty of functions to access vectors, while the C API is more powerful to process raster data than Java API. Since our vector data adapter is mainly called by WS server which is developed by Java, programming it using Java will simplified the integration and get high performance.

The vector data adapter has three methods as follow:

- SpatialQueryEx (String tablename)

  Establish a connection to a specified spatial table.

- QueryInBox (double minx, double miny, double maxx, double maxy)

  Retrieve the features in the bounding box and return a vector which contains all the geometry objects in the specified area. Every geometry object provide three kinds of information as follow:

  1. shape type: point, line, or polygon.
  2. attributes: what this geometry object presents, e.g. city name if it presents a city.
  3. coordinates of the points which this geometry object is made up of .

- SpatialQueryDisconnect ()

Disconnect from the SDE server.

4.4 Deployment of ArcIMS

4.4.1 Publish real data onto Internet

Authoring a map is the first step toward creating an Internet GIS application. ArcIMS Author can help us to add data layers, set layer symbology, create stored queries, and so on, then save the edited map as a map configuration file written by ArcXML

(AXL file). Publishing a map as a service involves using ArcIMS Administrator to register the map configuration file with the ArcIMS application Server. A map usually is published as an Image or Feature Service. ArcIMS Administrator let us can create and manage these services easily. The last step is designing a Web site using ArcIMS Designer for the map based on its services running on ArcIMS Spatial Server. Figure 17 and 18 show us two websites. One is based on Feature Service, while the other is Image Service.

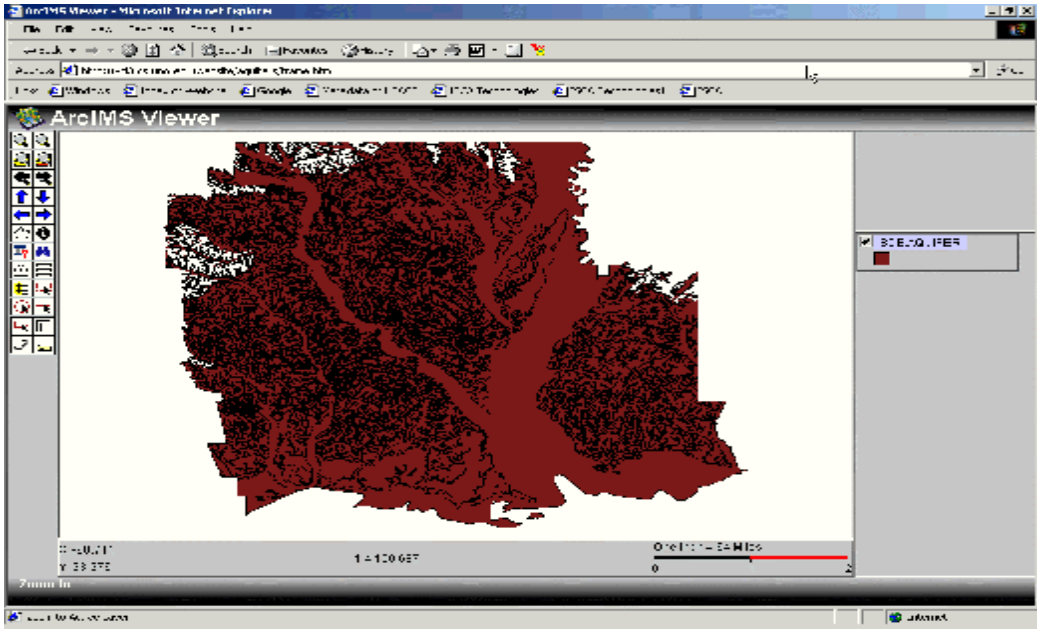


Figure 17: A Web site for Feature Service

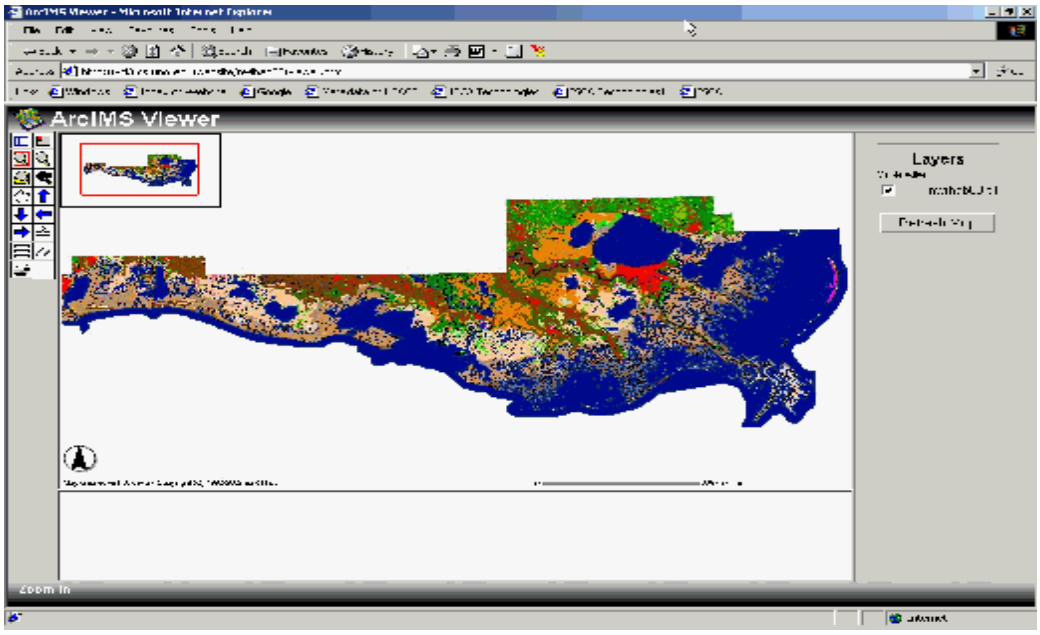


Figure 18: A Web site for Image Service

4.4.2 Web-based spatial data search on arbitrarily organized file system

There are three parts we need to develop in this application.

- Customize HTML Viewer using HTML and JavaScript
- Connect to and query information from database, create HTML Viewer dynamically using JSP and Java Servlet.
- Modify configuration file of ArcIMS service using ArcXML, so that the service can add the layers dynamically without changing itself.

We take advantage of build-in HTML Viewer and modify its Text Frame. When the user selects an area on the backgraoud—draws a bounding box with mouse, the bounding coordinates of the rectangle show in the Text Frame (Figure 19). Click " Submit" button, a java servlet file will be activated. The servlet will connect to the database and query maps name in this given area, then rewrite the HTML Viewer dynamically. These maps' names and locations where they are stored are shown to the

user (Figure 20). These maps can be visualized on the background via clicking “Add Layer” button (Figure 21).

This tool can be used though standard browser. The URL is http://vcl3.cs.uno.edu/website/Lousisana/viewer.htm

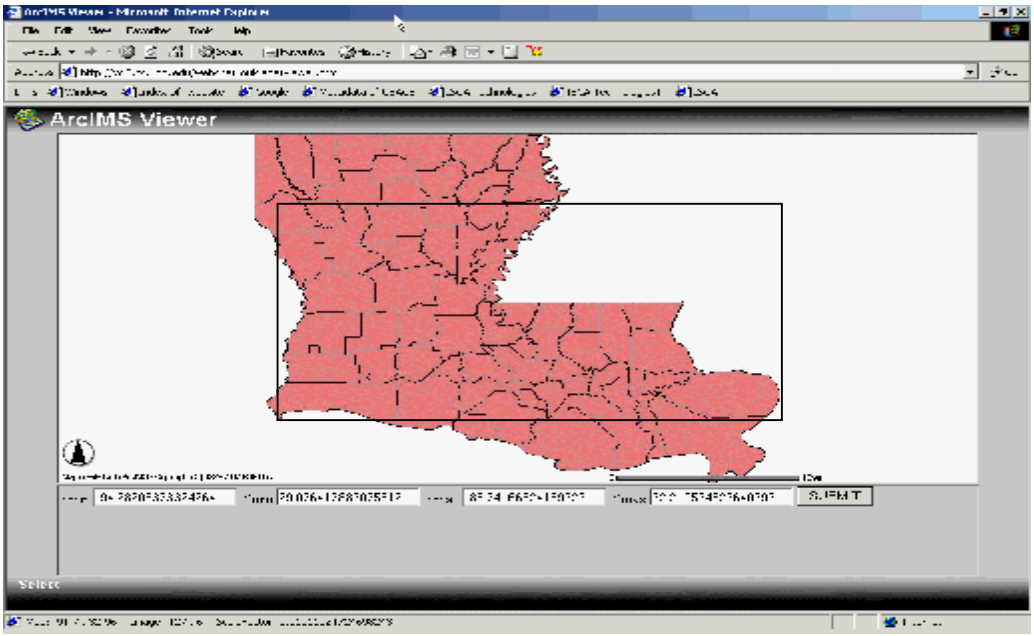


Figure 19: Select an area

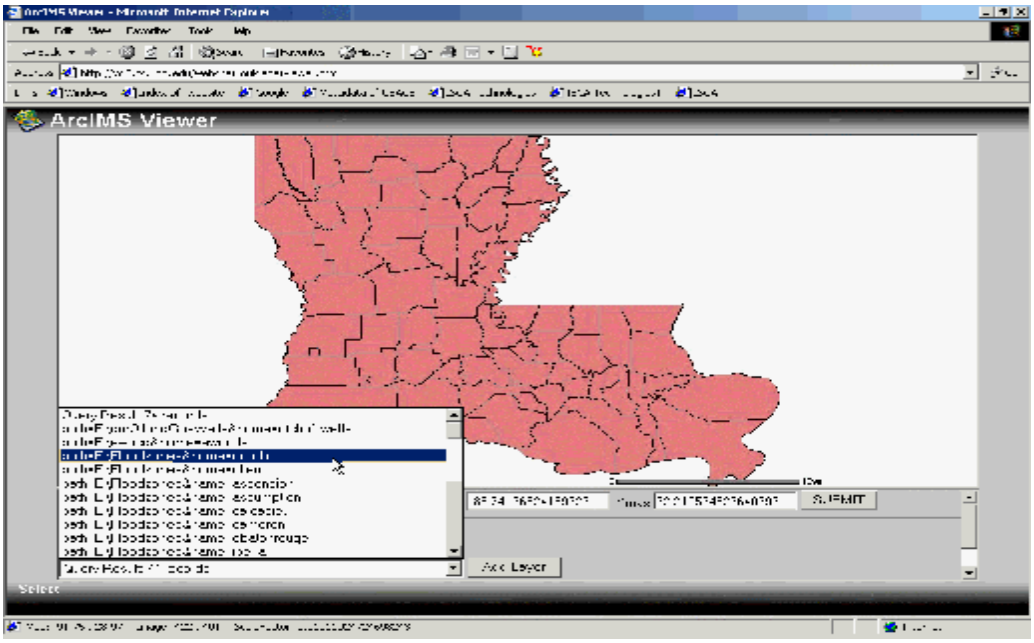


Figure 20: Select a map

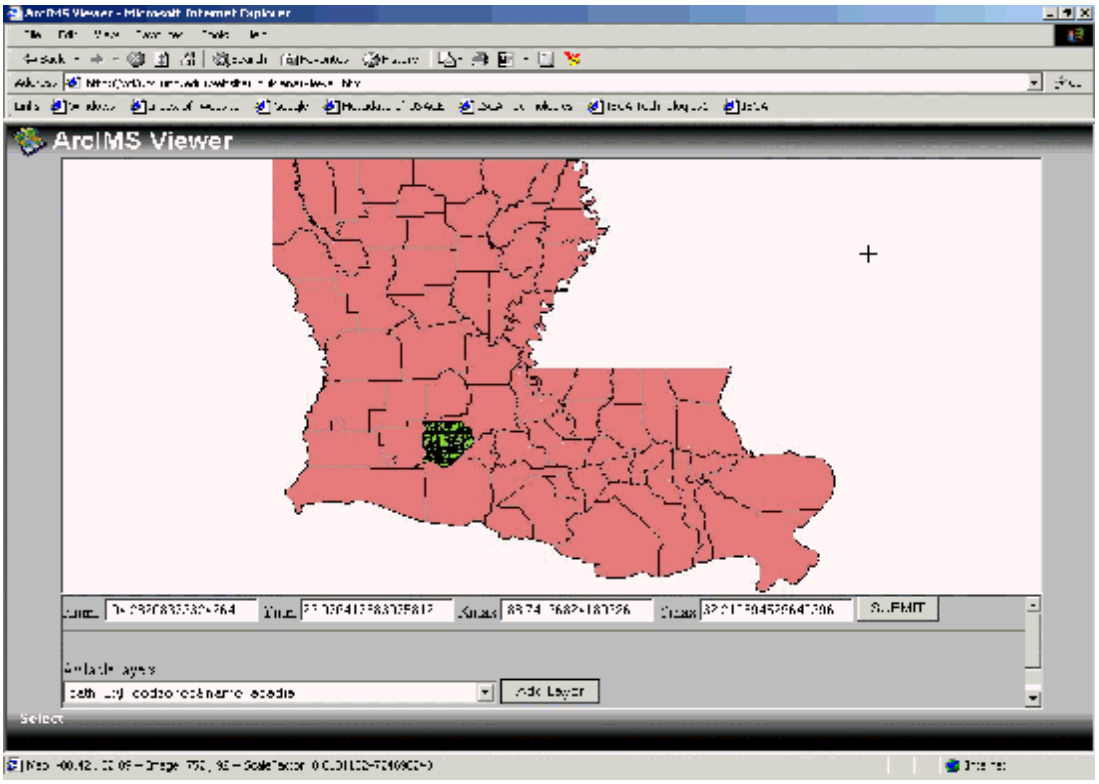


Figure 21: Map is added to Image Service as a layer

4.5 Metadata Service deployment

A Metadata service provides a catalog of metadata. The client might search, or browser Metadata Service when they need data. Similarly, the client can share, or publish data to a Metadata Service where others can see it.

ArcIMS is one part of the metadata system we created. ArcGIS and ArcSDE also play vital roles. Figure 22 shows how these components work together.

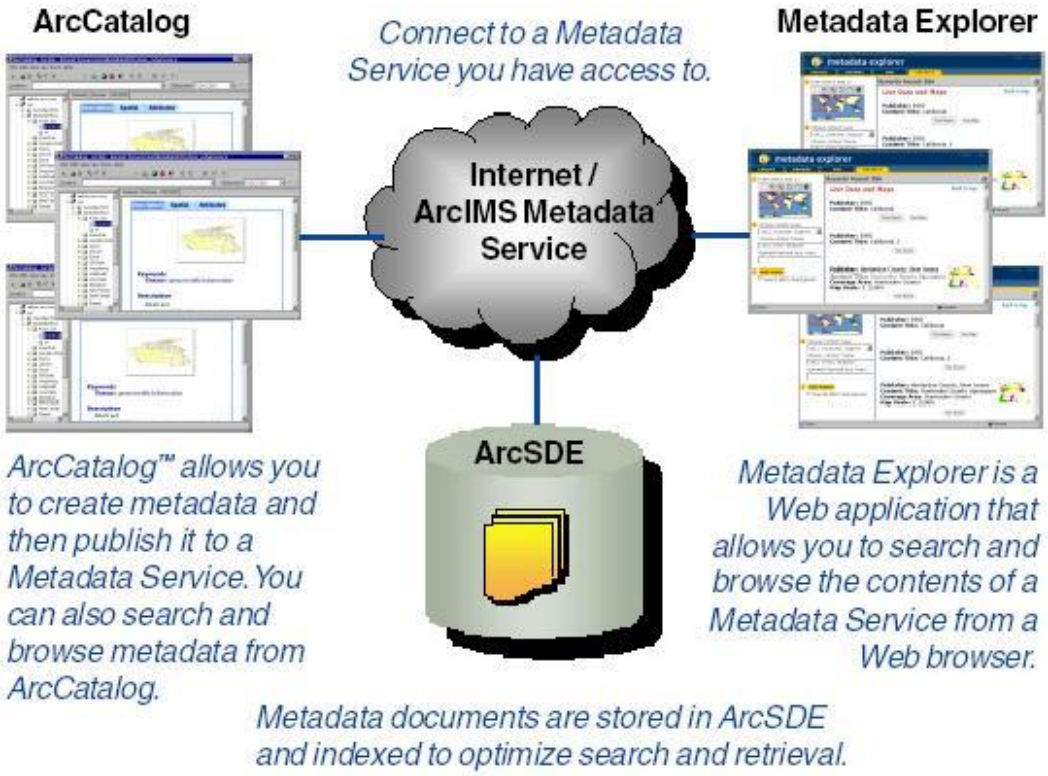


Figure 22: The components of Metadata System

ArcIMS provides the mechanism for hosting a Metadata Service, allowing clients to publish metadata to the service as well as search its contents. MetadataExplorer is a Web application included with ArcIMS that allows people to search and browse the contents of a Metadata Service from a Web browser. ArcGIS creates and author metadata in ArcCatalog then publish it to a Metadata Service. ArcCatalog provides built-in

editors—FGDC and ISO—to help user create metadata. When user is ready to publish his metadata, just connect to the Metadata Servic—with a username and password provided by the host of the Metadata Service—and copy his metadata onto the service. Now anyone who accesses the service will be able to see the published metadata.

ArcIMS utilizes ArcSDE and the relational database to store the published metadata documents. Each time a client publishes a metadata document to the service, a new record gets created in a database table. Metadata documents are also indexed to optimize searching and retrieving.

4.5.1 Setting up Metadata Service

Before we set up the Metadata Service, we have to install and configure ArcIMS. The steps of configuration for ArcIMS are different when we use different web server and servlet engine. In our ArcIMS system, we use Apache 1.3.20 and Jakarta Tomcat 3.2.

Four files must be created and configure properly when setting up a Metadata Service. They are

- MetadataServer.axl—configuration file for metadata service. This file is used to create Metadata Service.
- Aimsacl.xml—an access control list (ACL). This file list username and password used in ArcCatalog and Metadata Explorer. Figure 23 is a sample ACL file.
- Authenticate.properties—turning on authentication for Metadata Explorer. When the authenticate property is enabled, the pathname of ACL file must be set.
- Aimsmeta.properties—turning on the login page in Metadata Explorer.

```
<?xml version="1.0" ?>
<AIMSACL>

<USER name="*"
services="MetadataServer,SanFran,SearchMap,Gazetteer"/>

<USER name="browse" password="browse"
services="MetadataUSACE,MetadataServer,SanFran,SearchMa
p,Gazetteer,nwihab88,neworleanswestne"
roles="metadata_browser" active="1" />

<USER name="publish" password="publish"
services="MetadataUSACE,MetadataServer,SanFran,SearchMa
p,Gazetteer,nwihab88,neworleanswestne"
roles="metadata_publisher" active="1" />


<USER name="author" password="author"
services="MetadataUSACE,MetadataServer,SanFran,SearchMa
p,Gazetteer,nwihab88,neworleanswestne"
roles="metadata_service_author" active="1" />


<USER name="admin" password="admin" services="SanFran,
SearchMap,Gazetteer,nwihab88,neworleanswestne,moritor,m
oritor1"
roles="metadata_administrator" active="1" />

</AIMSACL>
```

Figure 23: A sample of ACL file

Figure 24 shows the Metadata Explorer we host for USACE. The URL is http://vcl3.cs.uno.edu.

This Meatadata Service also can be accessed via ArcCatalog shown in Figure 25.

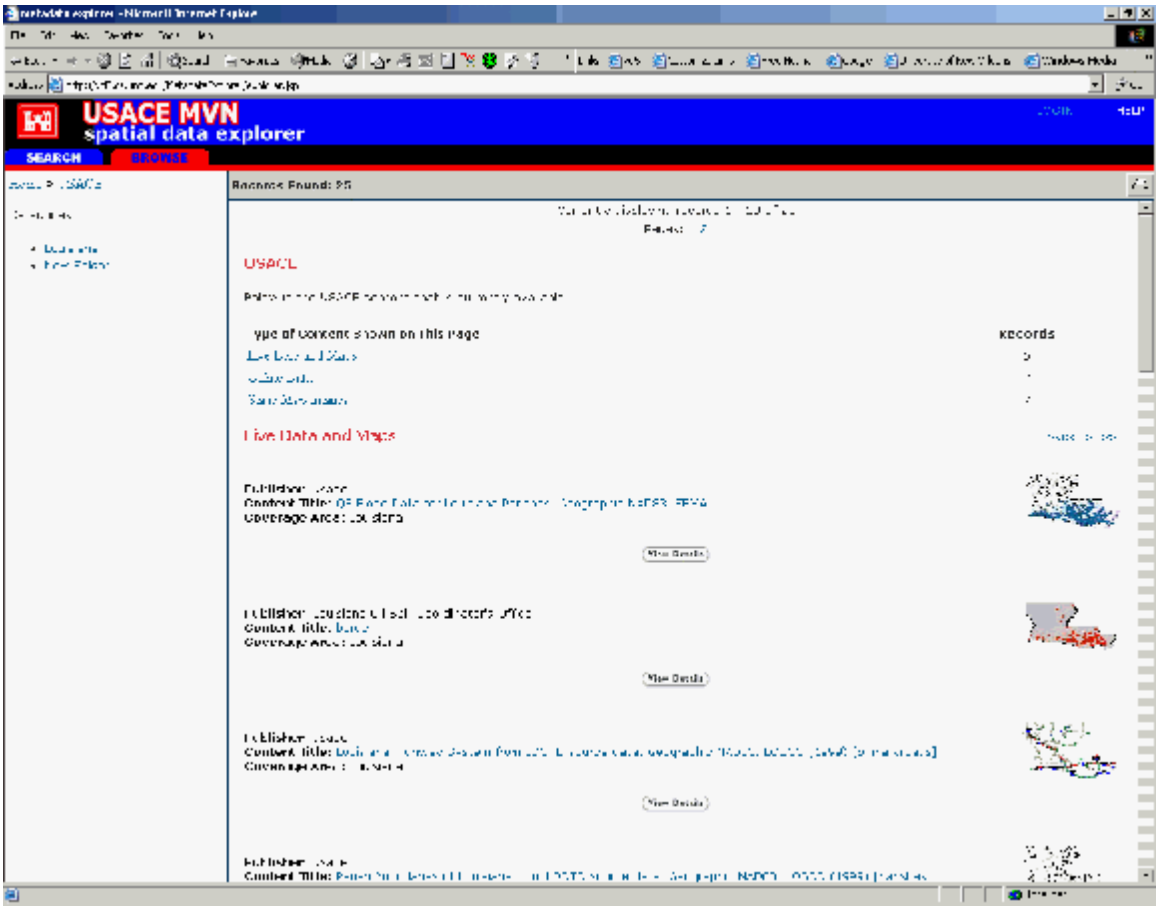


Figure 24: Metadata Explorer

4.5.2 How to create" Live" data

Live data and maps are dynamic geographic services that allow direct interaction with map content. This type of content is delivered in one of two ways: as a cartographic image or "snapshot" of a map, or as compressed vector features that are streamed to you. Streamed features allow for greater client-side interaction including dynamic labeling, feature symbolization, and MapTip creation. The client does not need to download anything to use live data—just add it to the map and begin exploring.

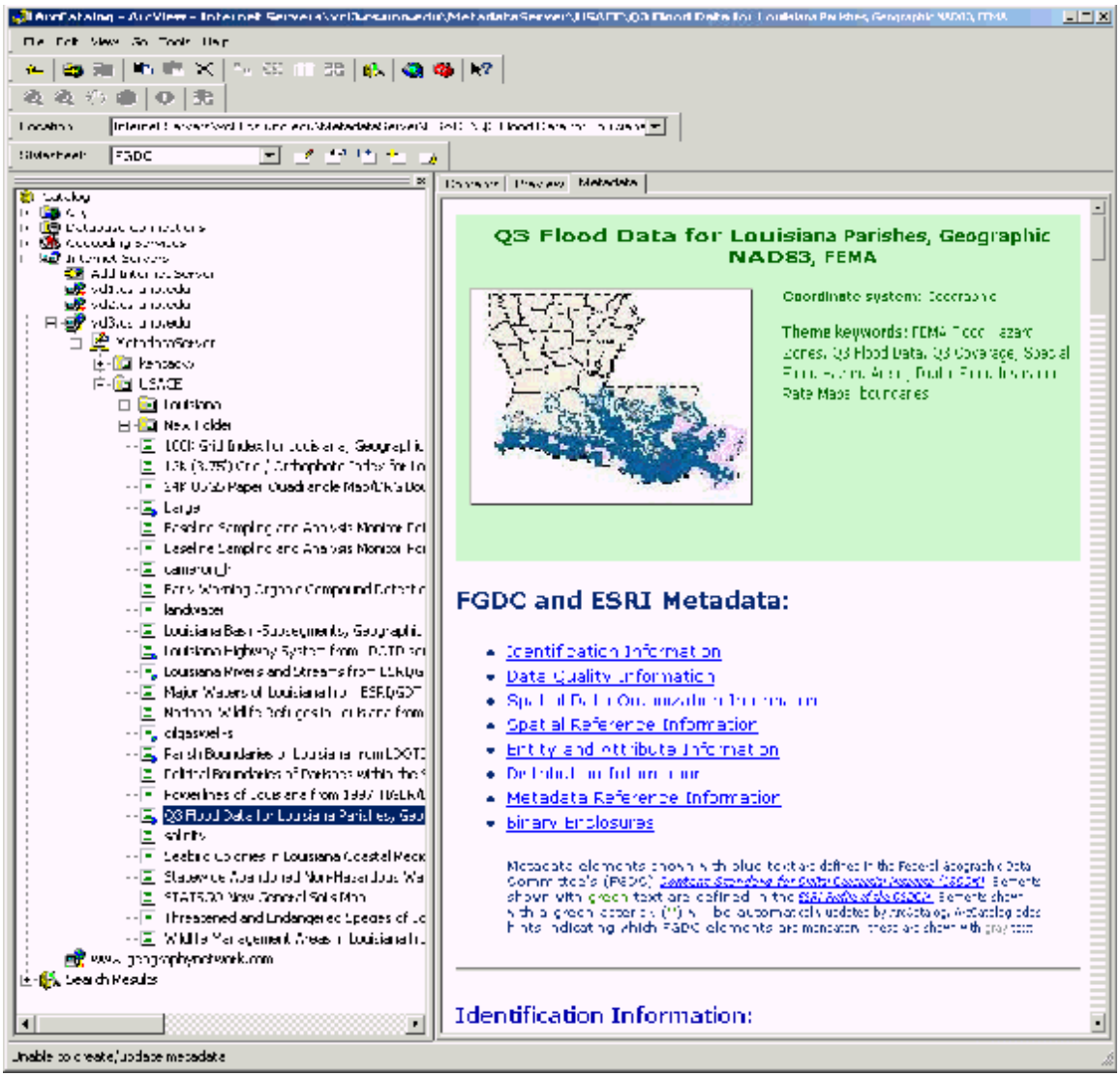


Figure 25: Administrate Metadata Service using ArcCatalog

If the client launchs a Metadata Explorer from ArcMap or ArcExplorer. The records of Live data should have an extra button named " Add to ArcMap" or "Add to ArcExplorer" shown in Figure 26. Click this button the map selected will be added to ArcMap or ArcExplorer as a layer. This is a very convenient tool for the clients to search data and use them as theirs. However, it is not so easy to define a map as "Live data and Maps" as the ESRI's user guide said that we just need to edit the content type as "Live data and Maps" using build-in Metadata Editor, then publish it. We tried many means to

solve this problem. The only way to get a “Live data” is create it for a specific service (Image or Feature) using ArcCatalog. In Figure 25, we can see the “Live data” created in ArcCatalog have a bule mark on the lower right corner of their icons. These data are the real “Live Data and Maps” in Figure 26.

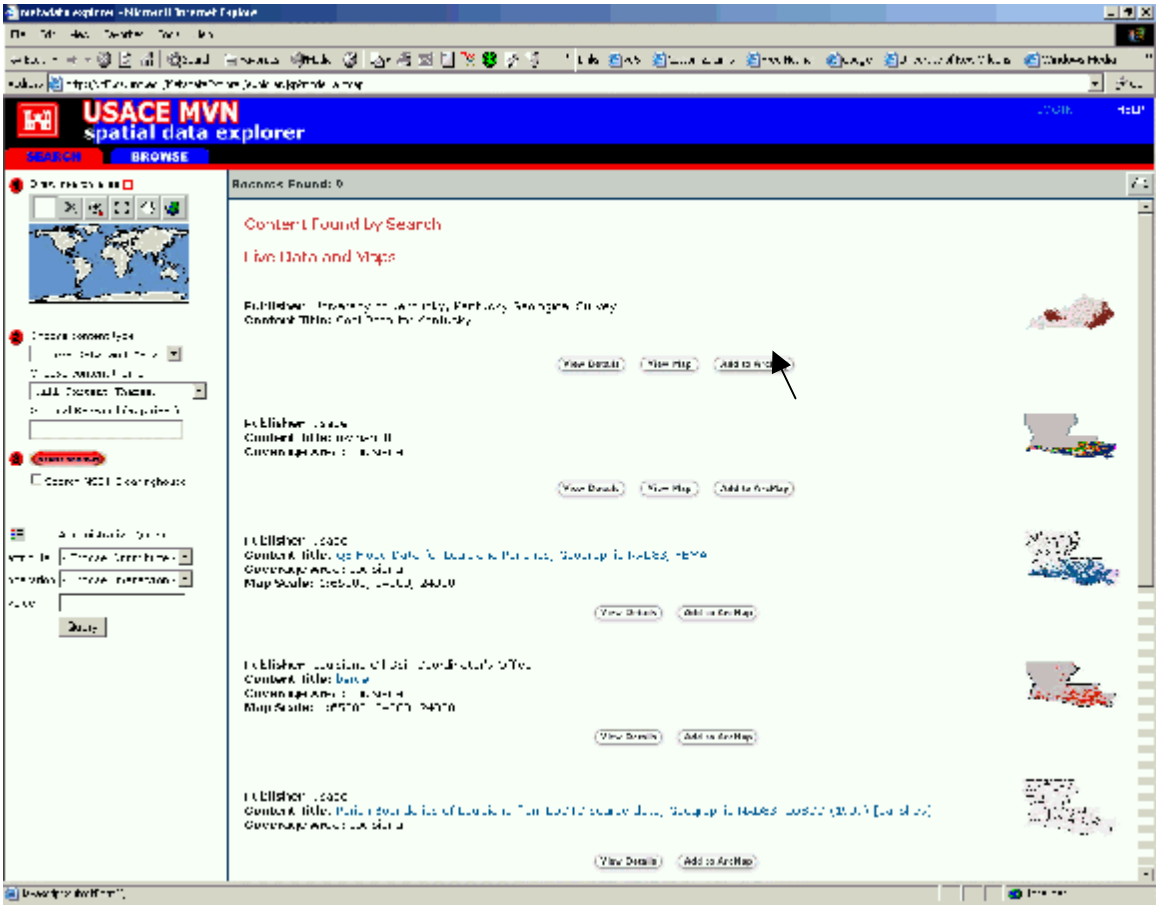


Figure 26: Metadata Explorer launched by ArcMap

4.5.3 Administration table

Administrative table for metadata server is an extension of the metadata service to help administrator manage the data. It can store any information that an administrator wants to have. We create an administrator table as a simply additional database table. Each entry in the table is linked to a published metadata document. The administrator

table should be created before the Metadata Service starts. This will ensure that whenever a document is published to the Metadata Service, a corresponding record is inserted into the administrator table. In order to implement this query functionality on Metadata Explorer, we use Java Servlet and JSP to communicate with the administrator table stored in database and show the result to the client (Figure 27).

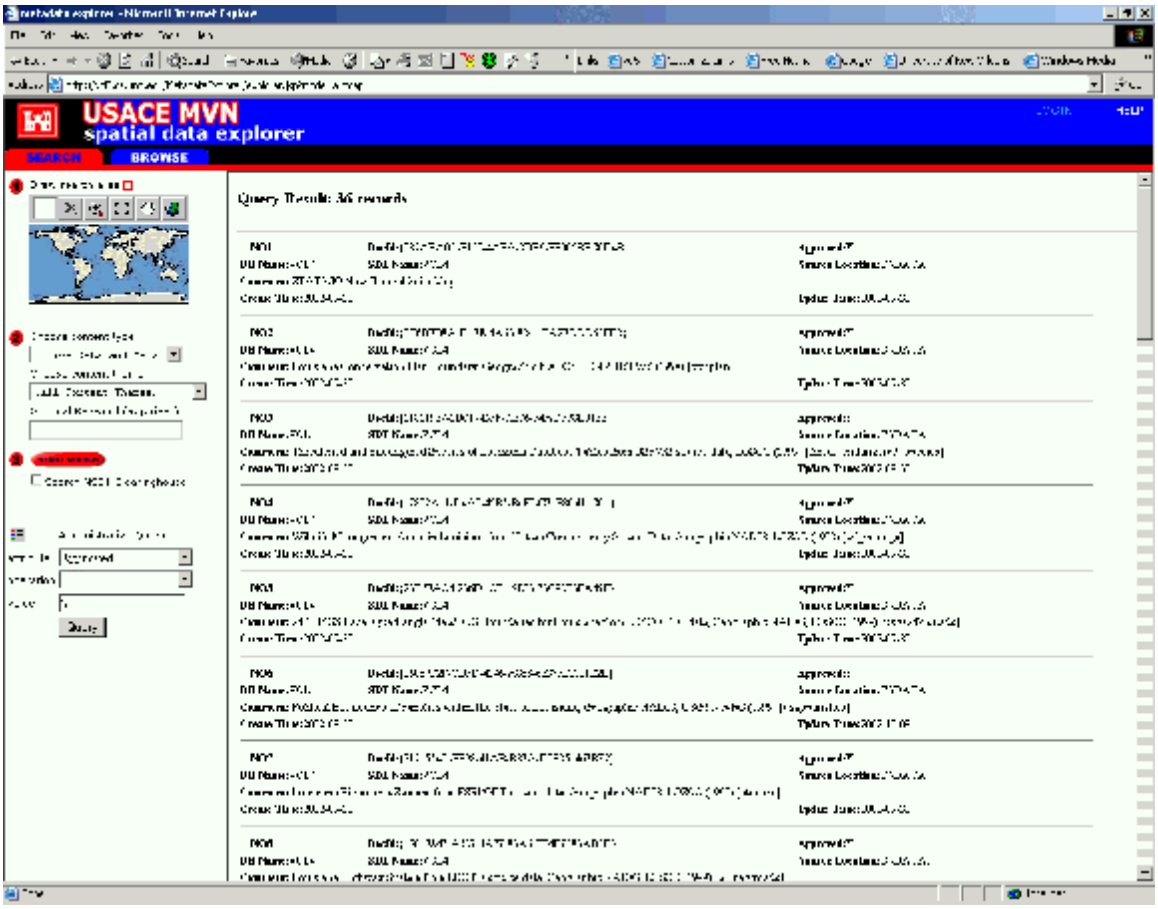


Figure 27: Administrative Query

4.6 Implementing SOAP-Based GIS Web Services for MicroStation

The web services implementation is a part of implementation of a GIS web service client for Bentley's MicroStation CAD program. The purpose of this work was to discover an efficient mechanism to allow the Bentley MicroStation software [26] to

access the E-GIS's central geo-spatial data resource, the ArcSDE servers. Such an achievement would harness the power and easy use of Microstation to draw geometries for its projects. Microstation is the program of choice for generating vector data for many GIS projects at USACE as well as many local governments' GIS workers.

A solution is to use the old Java capability in MicroStation Version 7 (called MicroStation/J) and the latest Java web services developer pack (WSDP) to implement the link between the "web services client agent" and the web services consumer, MicroStation/J.

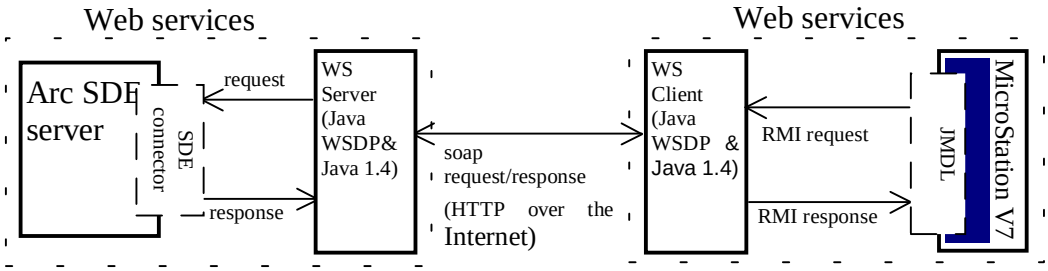


Figure 28: The web services for Microstation

The web services we implemented are the WS Server and the WS Client. The WS server connects to the ArcSDE server to get image tiles. This is implemented by using Java Native Interface (JNI) because the SDE connector was written in C. The WS Client is also programmed in Java. It uses RMI request and response to communicate with MicroStation. This is also a stand-alone client web services that can be used by other applications.

This wrapper web service is developed with the Java Language and uses SOAP message to stream the output image data. The Java web service accesses a DLL library using the JNI (Java Native Interface). The DLL includes the methods that can access the spatial dataset in the database through SDE server. There are only three methods in the

DLL, one is for establishing a connection to SDE server, the other is for disconnect. The main method is the getRasterFile(), this method will be designed to return an array of bytes to the caller. All the web services methods are in the main class sdeWS. The public methods of this class are the web services' methods. To configure the methods to be provided, you have to set the deploy.wsdd file correctly in the installation of the web services, this is documented in the Developer's guide. For example, Figure 29 is the method that connects to the database:

```
public synchronized String getSDEChar(int x_tile_num, int y_tile_num, int
selected_level)
{
   try {
      byte[] imgarray = new byte[16384];
      return RasterAgent.getSDEChar(x_tile_num, y_tile_num, selected_level);
   } catch (UnsatisfiedLinkError e){
      e.printStackTrace();
      return null;
   }
}
```

Figure 29: An example how to connect to the database

It is very simple. It calls the getSDEChar method (this method uses JNI to get the image from the database) and returns the result as a string (in fact this string is a byte array encoded as a string).

The web service will use this String to construct the SOAP message. The SOAP Message is, in fact, as follows in Figure 30:

```
<soap-env:Envelope
xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap-env:Body>
  <ns1:getSDECharResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://soapinterop.org/">
                  <getSDECharReturn xsi:type="xsd:string">OutputStream
     </getSDECharReturn>
  </ns1:getSDECharResponse>
</soap-env:Body>
</soap-env:Envelope>
```

Figure 30: An example of SOAP Message

By using Apache Axis, the java programs do not have to implement these SOAP messages. SOAP is handled by Axis itself, which generates SOAP message. This message sent back to the user can then be parsed. The Output Stream is then used to display the image. The output stream is a string containing the value of image dataset.

For a RGB file, the array includes three bands, each tile contains 128*128 pixels, and thus the size of this array is 3*128*128.

The output stream is as follows in Figure 31:

…
UVt0kJlsOzpSYWNmaF1fZWlscnd2cHBucXV6dHRfQyodGxkeHyMxRlVTQT
QdHh0bHSYxMyYj
JCIhIiMsTnmMcUo3NTIyMSojICcuGRYcHRsdIB0eIBkUEhcgHyUpN1NMSE
1SSlNPRjk4PDZJ
V2ZvgIuIf4KPiIJxVURGSFBaa2RdaIScjUktJjRNYGBgZGNscG5xeHh0cG1vd
Xx3d3FlTzEe
…

Figure 31: An example of output stream

This is a base64 Binary string that can be decoded into a byte array on the client side.

Figure 32 shows the classes of the server side web services:

Base 64:	Encode the byte array into a string
RasterAgent:	Static methods
RasterReader:	Native methods (JNI) to connect, getData and disconnect of the database
SdeWS :	the web services methods (connect, getTile and getVector)
SerialRow:	used by SpatialQueryEx
SpatialQueryEx:	useful for the getVector method

Figure 32: the classes of the server side web services

CHAPTER 5

CONCLUSION

For large organizations whose core businesses are GIS related, using multiple COTS GIS suites is a common phenomenon, because different COTS products often have different strengths in various applications. On the other hand, deploying heterogeneous GIS software has the tendency to form fragmented data sets and to cause inconsistency. Data consolidation is an effective way to preserve data integrity. To accomplish this, we must achieve interoperability between different GIS tools. While vector spatial data have available standards and the implementations to support interoperability, raster based images have only one common ground -- files. Compared to the database approach, the file-based management for raster data is disadvantageous in terms of performance and flexibility. We implemented an adapter that allowed the file-based GIS tools to access the raster data storage in databases. Furthermore, we have enhanced this adapter according to the J2EE-CA standard. Using this kind of J2EE-CA adapter, any GIS can be embedded into the J2EE framework easily. The J2EE framework is an ideal server platform for building integrating software components inside the enterprise geographic information systems. This technology has been matured and stabilized in recent years.

Recently, the service delivery technologies for Internet computing have been converged to an open standard model, The SOAP-based Web Service is a promising technology in supporting the system integration tasks of GIS system in a distributed environment. Web Services represent a widely accepted point-to-point, cross-platform and standard based communication protocol that connects a pair of computational parties. Parrel to this general Internet computing model, the GIS community has developed GID specific standards, the Web Map Service (WMS) and the Web Feature Service (WFS). These two OGC standards have been implemented by the leading GIS software vendors. For nontrivial systems like GIS system, behind each web service is a backend system that fulfills the application specific tasks.

In our project, we have streamlined the COTS GIS server, the J2EE coordinator server, the web service provider components and the COTS web publishing tools into a hybrid web service architecture, in which the enterprise information system integration, the web publishing and the business-to business online services are uniformed.

REFERENCE

[1] Carnegie Mellon Software Engineering Institute, COTS-based systems initiative, http://www.sei.cmu.edu/cbs/cbs_description.html.

[2] J. Dean, A. Gravel (Eds.), *COTS-Based Software Systems*, (*Proceedings of the First International Conference on COTS-Based Software Systems* (ICCBSS 2002), Orlando, FL, USA, February 4-6, 2002), *LNCS 2255*, Springer-Verlag, Heidelberg, Germany, 2002.

[3] C. Albert and Lisa Brownsword, Forward, "Meeting the challenges of commercial-off-the shelf (COTS) products: the information technology solutions evolution process (ITSEP)", in [2].

[4] R. Seacord, "Replaceable components and the service provider interface", in [2]. [21] Rahul Sharma, Beth Stearns, and Tony Ng, *J2EE™ Connector Architecture and Enterprise Application*, Addison Wesley, Reading, MA, 2002.

[5] Berkeley. Web GIS and interactive mapping sites. http://sunsite.berkeley.edu/GIS/intergis.html.

[6] Maurizio Morisio, Marco Torchiano; Definition and Classification of COTS: A Proposal

[7] S. Tu, L. Xu, M. Abdelgurerfi, and J.J. Ratcliff, "Achieving Interoperability for Integration of Heterogeneous COTS Geographic Information Systems", Proceedings of ACM GIS 2002, McLean, VA, pp 162-167.

[8] GIS by ESRI, "What is ArcGIS "

[9] Intergraph Corporation, GeoMedia Professional, http://imgs.intergraph.com/gmpro/

[10] Bentley System Inc., MicroStation, http://www.bentley.com/products/

[11] Oregon State University, EPA Storm water management model, http://www.ccee.orst.edu/swmm.

[12] Boss International, the UNET model, http://www.bossintl.com/html/rms-unet.html.

[13] National Oceanic and Atmospheric Administration Coastal Services Center, Topographic LIDAR, http://www.csc.noaa.gov/beachmap/startup.html.

[14] United States Geological Survey, USGS Digital Elevation Model Data, http://edcwww.cr.usgs.gov/glis/hyper/guide/usgs_dem.

[15] The Hydrologic Engineering Center, US Army Corps of Engineers, http://www.hec.usace.army.mil/.

[16] ESRI, ArcGIS, http://www.esri.com/software/arcgis/index.html.

[17] Intergraph, Mapping and GIS solutions, http://www.intergraph.com/software/.

[18] Tcl developer exchange, Tcl Resources, http://www.tcl.tk/.

[19] Kitware Inc., Visualization toolKit, http://public.kitware.com/VTK/.

[20] Vlada Matena and Beth Stearns, Applying Enterprise JavaBeans: Component-Based Development for the J2EE Platform, Addison Wesley, Reading, MA, 2001.

[21] Dirk Slama, Jason Garbis and Perry Russell, *Enterprise CORBA*, Prentice Hall, Upper Saddle River, NJ, 1999.

[22] Shengru Tu, Xuefeng Li, Xiangfeng He, and Jay J. Ratcliff, "A Systematic Approach to Reduction of User-Perceived Response Time for GIS Web Services", *Proceedings of the 9th ACM International Symposium on Advances in Geographical Information Systems* (ACM-GIS 2001), Atlanta, GA, pp 47-52, November 2001.

[23] Z/I Imaging, Image management and distribution, http://www.ziimaging.com/Products/StorageAndECommerce/default.htm.

[24] Intergraph Inc., Intergraph Announces New GeoMedia 5.0 Product Suite, (announced on May 6, 2002) http://www.intergraph.com/gis/newsroom/press02/gm5_rlsf.asp.

[25] SourceForge.net, TIFF Software, http://www.libtiff.org.

[26] Bentley/ESRI ACE-GIS Interoperablility (White Paper), http://www.bentley.com/files/products/bentleyesri/bentley_esri_wp.pdf

[27] D.J. Reifer, V.R. Basili, B.W. Boehm and B. Clark, "Eight lessons learned during COTS-based systems maintenance", IEEE Software, Sept-Oct 2003.

VITA

Ying Wu was born in Tianjin, China in1973. Received bachelor degree from Mechanical and Electrical Engineering Department of Beijing University of Aeronautics and Astronautics (BUAA) in China (1992-1996). Then she began her gradate study in Robot Research Institute and got master degree of Robotics at 1999. During 1999-2000, she worked at Great Dragon Telecommunication Company in China as Software Engineer

From January 2001, she obtains a graduate assistant position in the Computer Science Department of University of New Orleans. Research interests include distributed system, database system and web-based system. Current research work on integration of COTS Geographic Information System which may employ multiple technologies like EGIS, Oracle Spatial database, J2EE, SOAP-based Web Services etc.