Design and Implementation of distributed Web Services in Mobile GIS Application

Zheng Chang
February 2003

Design and Implementation of distributed Web Services in Mobile GIS Application

by

Zheng Chang

Thesis submitted to the International Institute for Geo-information Science and Earth Observation in partial fulfilment of the requirements for the degree of Master of Science in GEOINFORMATICS

Degree Assessment Board

Degree Assessment Board:
Chairman: Dr.Ir. R.A. de By
External examiner: Dr.Ir. M. van Keulen
Supervisor: Ms. Y. Sun M.Sc.
Second supervisor: Ir. R.L.G. Lemmens

INTERNATIONAL INSTITUTE FOR GEO-INFORMATION SCIENCE AND EARTH OBSERVATION
ENSCHEDE, THE NETHERLANDS

Disclaimer

This document describes work undertaken as part of a programme of study at the International Institute for Geo-information Science and Earth Observation. All views and opinions expressed therein remain the sole responsibility of the author, and do not necessarily represent those of the institute.

Abstract

In this dynamic world, information is essential for foot-travelers when they are traveling. For foot-travelers, there are great needs for personalized and Location Based Services. Nowadays, when application developers are trying implement proper applications for foot-travelers, dynamic data storage and software selection are real challenges. To help application developers avoid those problems, the suitable technology of the latest is "Web Service". A Web Service is simply an application that exposes a Web-accessible Application Program Interface, it stores relevant source data and functionality in server side, and it offer Web access to it. Moreover, a Web service is a set of standards that applications follow to achieve interoperability via the Web, it has programming language independent and platform independent characters.

In this research Web Service building components are studied first, then a suitable Web Service architecture is build which special for implanting GIS Web Services. Foot-travelers required services are classified into few major categories for consequence Web Services design.

Based on identified Web Service building components and architecture, the research implemented few GIS Web Services which for foot-traveler application which can help foot-travelers' application developers. Also, a simple client side application is build for testing implemented GIS Web Services. It has been found that Web Services is a very promising architecture for practical implementation of distributed geographical information systems, and Microsoft .NET and ERSI's ArcObjects are proper building components for such Web Services Architecture.

Keywords:

Web Services, Foot-travelers services, SOAP, Web Service Architecture, GIS

文章摘要

对于步行旅游者来说，个性化及的基于地理位置服务非常重要。当软件开发者开发步行旅游的应用软件时，动态数据存储和软件开发环境选择总是一个领人困扰的问题。“ Web 服务”是当今适宜帮助软件开发者避免或减少这方面的困扰的最新技术。“Web 服务”是发布在 Web 上的应用程序接口，它储存相关的原数据及其功能在 Web 服务器端并支持 Web 访问。此外，“Web 服务”是一整套遵循互用性的公开应用标准，它不受编程语言及操作平台的限制。

在本项研究中，在调查了“ Web 服务”框架的组成模块的基础上创建了“Web 服务”框架。此“Web 服务”框架借鉴标准的“ Web 服务”框架基础但特别应用于创建空间数据处理的“ Web 服务”。为有助于后期的空间数据处理的“ Web 服务”具体设计及开发，步行旅游者所需求的服务被分析并根据不同特性规划分类。

基于选定的 “Web 服务”框架组成模块及框架本身，本项研究开发了几个空间数据处理的“Web 服务” 事例，其将有助于步行旅游者需求应用开发。此外，简单的客户端应用程序被开发用于测试已完成的本项研究开发了几个空间数据处理的“ Web 服务” 事例。本项研究证实“Web 服务”框架适合开发分布式的地理信息系统应用 ，Microsoft .NET 和 ESRI 的 ArcObjects 适宜成为此框架的组成模块。

关键字：

“Web 服务”，步行旅游者，SOAP, “Web 服务”框架, 地理信息系统

Acknowledgements

I'd like to express my gratefulness to my supervisors Ms. Yuxian Sun and Mr. Rob Lemmens for their academic guidance they gave me throughout the project period.

Special gratitude will be given to the members of the GIP staff for their valuable comments contribute to develop of this thesis.

Contents

list of Figures

Chapter 1.

Introduction

1.1. Background

Travel, like many other aspects of daily life, is being transformed by technology revolution. People in field traveling are looking for easy and portable terminals for access to information relevant to the user's position, and there are many applications have been implemented towards this direction. In general, when travelers are visiting new places, they all need many Geo-related services before the trip or during the trip as the following. For example, when traveler on his way to his destination, he may check what's the shortest way to reach the destination place, and the traffic situation. When a traveler in site, he need local attractive activities information, such as parking place or restaurant

As a matter of fact, every individual traveler has different requirements for a particular travel application, but each other's basic needs have no much difference (as the problems mentioned above). Also, with the substantial progress in mobile communication technology achieved in the recent years, more and more travelers are equipped with mobile phones. So with Internet and WWW technologies attached to digital mobile phones and other wireless terminals, i.e., adapting the Web architecture to the wireless environment, travelers' activities can be benefit from it greatly. Meanwhile, the technical limitations of wireless system and mobile devices should be considered. Providing Internet and WWW services on a wireless data network presents many challenges. Most of the technology developed for the Internet has been designed for desktop and larger computer supporting medium to high bandwidth connectivity over generally reliable data networks. Hand-held wireless devices present a more constrained computing environment compared to desktop computers. Because of fundamental limitations of power and form factor, mass-market handheld devices tend to have less powerful CPUs, less memory (ROM and RAM), restricted power consumption and smaller display screen.

Based on travelers' requirement and available technology, many applications have been implemented to help travelers' needs. Traditional approach was highly integrated programs that allow little interaction with other related applications, thus an application can't benefit from other implemented applications. For example, Monternet Company [1] implemented such a mobile GIS application for its subscriber. By using its service, traveler can send a command to check his/her position information on his mobile phone, and the interesting sites (such as shops, restaurants or bus station around) [2]. Also, he

can send a command to request the position information of his friend with same service subscribed. Technically, there is a dataset including geography information in the server, and a communication interface between mobile stations system and this database. In this case, the GPRS (General Packet Radio System) is the protocol for the wireless communication between mobile device and mobile stations. As mentioned above, this kind application is isolated from other application, thus can't make use of other implemented application. Due to the fact that many applications share same characters, if an implemented application can be used by other applications, it will save a lot of time for develop an application and developers can easily combine several Web Services to create one complete solution.

As Web Service scheme been introduced and accepted by many major companies and organizations, it enables us make the Internet a platform for delivering services, not just data. The basic Web Services scheme is displayed in figure 1-1 below, which accepted by both IBM and Microsoft. Currently, there are many implemented applications (including GIS applications) can be access through Internet, by using Web Service technology, remote objects can be invoked by another application whenever that application needs it.

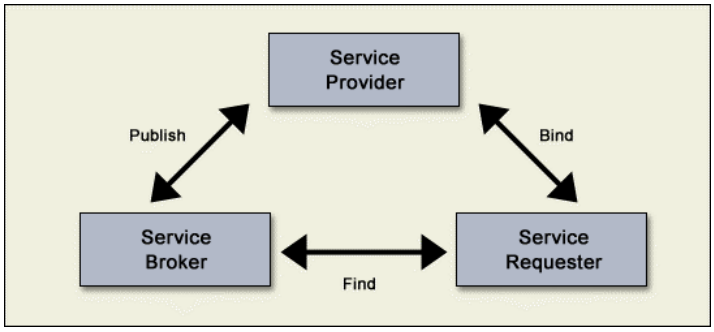


Figure 1-1: Basic Web Service scheme

- Publish: A service provider creates a Web Service and its service definition and then publishes the service with a service registry based on a standard called the Universal Description, Discovery, and Integration (UDDI) specification.

- Find: Once a Web Service is published, a service requester may find the service via the portal (UDDI interface).

- Bind and Consume: The service requester may then use this information to directly bind to the service and invoke it. Note: it is possible for a web application bind with a certain Web Service (or services) directly, without pass through UDDI.

As word about Web Services has spread in the last few years and actual implementations of Web Services appear on the scene, such implemented applications as Google Web APIs service [4] and OGC Web Services Initiative [5]. Like many other Web Services, they use SOAP to transfer information back and forth to clients (local applications). SOAP brings together two industry standard languages for communicating over the Internet: hypertext transfer protocol (HTTP) and Extensible Markup Language (XML). [6] In general, they are developing Web-based services and services accessible via the Internet. Each service supports a certain operation function that returns metadata describing the

nature of the service, the methods to register your own Web Service, the addresses to access the service or to contact its provider, the type of data provided by the service, and so forth. During the research, the process and relationship of Web Service publish; find and bind need to be investigated.

As the Internet becomes a ubiquitous feature of our daily life, we will use it to satisfy our information needs in much more powerful and yet simpler ways than having a PC at home and browsing web pages. Due to there are a great deal of irrelevant information on the internet which disturb users to find their specific targets, it will be more efficient if we just using certain Web Service application to get specific required result. This applies particularly to information with a significant real world context, such as traveler information. For example, a person planning a tourism trip may use their PC at home to access some tourism site that helps to prepare the visit and getting a general idea of what to expect. However, some types of information will be much more useful when that person is actually making the visit, e.g., information about local attractions, shopping places, or transportation. Through the Wireless Application Protocol, the traveler may make his/her dynamic search over traveling information relevant to his/her current location in real time, while his location information will be used as input data for Location Based Service.

This research needs to identify the typical traveling Geo-related problems first. Based on identified problems, technical ways should be investigated to find out which mobile GIS architecture (based on Web Services) can be used to facilitate traveler's needs. This research is going to study the components of a mobile GIS application model that used to link spatial data in order to make web GIS application more flexible. A detailed explanation will be presented in methodology section. The research will then discuss the implications of the architecture of mobile GIS services and develop a suitable prototype that can fit travelers' needs, such as site matching and route planning. Finally an evaluation of the achievements and limitations of the GIS project based on Web Service applications will be conducted.

1.2. Problem definition

For travelers (in mobile status) who equipped with mobile devices and capable for wireless communication connection, there is a great need for the Mobile GIS services which can solve their Geo-relating problems. Due to the fact that mobile devices don't have enough memory space to install GIS processing software and store dynamic spatial dataset, an important consideration should be given to what are the requirements and suitable architecture to implement GIS Web Services which Mobile GIS services will be benefited, and how to implement such architecture in the real world.

To reach above goals, it necessary to investigate what are the essential functional components (operations and data) in building Web Services and foot-traveler's application? And what functional components can reside on the client side and what on the server side, based on what criteria?

How can standards such as SOAP be used in the communication between client and server(s)? Besides SOAP, what is the role for this architecture of other standards in the Web Services protocol stack such as WSDL and UDDI?

What are the building blocks of Web Services and what is needed to build operational Web Services? Are Web Services best built from scratch, or is it better to let the Web Services provided by existing software? Can we use for instance ESRI ArcObjects to build GIS Web Services?

1.3. Research objective

The main objective of this research is to study current Web Service technology, and design a proper architecture of GIS Web Services that suitable for foot-traveler's application. The architecture will adopt open standard technology for an easy incorporation of new Web Service's providers and consumers. Then Based on the architecture designed, to implement several GIS Web Services in solving some typical foot-traveler's required services. Also, by using the architecture defined previously, implement a client side simulated application which make use of implemented GIS Web Services, which enable travelers query for their geo-relating services more easily with suitable interface.

1.4. Thesis outline

In chapter 2 some additional basic concepts that support this research will be introduced, such as Web Services, Web Services architecture and foot-traveler required services, together with some existing Web Services that on the Internet. In chapter 3 foot-traveler required services will be classified into four major categories, the classified requirements will facilitate the corresponded Web Services defining. In chapter 4 the research will implement several GIS Web Services based on selected software, GIS component and proper Web Services building architecture. Finally, in chapter 5 the conclusions and recommendations of the research will be explained.

Chapter 2.

Web Service Technologies

This section provides a general review of traveller requirement and Web Service technology. The Web Service architecture is presented for a better understanding of what the Web Service technology is and its advantages. In addition, Web Service programming stack will be introduced to explain how to implement Web Services. Then, two implemented Web Service applications for GIS and their functions will be introduced and described. Further more, an explanation of what travellers equipped with mobile devices can benefit from Web Service technology will be given, and limitations of Web Services will be considered as well.

2.1. Traveler requirement based on current technology

Travel is one of the most important aspects of human life, either for enjoying the life or being on business trip. Since people do not always repeatedly visit the same place, so the preparation for a trip and access on-site services are important for travelers. Decades ago, most travelers prepared their trips by study paper maps. When they are on-site, they need to be aware of issues like their relative position, direction to go and etc all the time. In addition, there are so many things travelers cannot predict or know in advance; such like weather information, transportation availability, interesting sites around etc. This is really a job for experienced travelers to manage carefully.

Later when the Information Technology (IT) emerged, travelers could manage their trip more conveniently. They can use some implemented applications to search the information they need for travel. One problem is that these implemented applications are characterized as tightly coupled applications and subsystems. Specific applications only help travelers solve certain problems in a specific region. These applications are written in different programming languages, and they are isolated from each other. In this way, when application developers are developing certain applications even if re are similar or same function or component in other application, they cannot use it directly, but have to develop it by their own. For the travelers, they require a complete solution to help them, especially in a new environment. They do not want search from many different sources for the information they need.

Also, we should be aware that most travelers even do not know how to search for the information they need.

In recent years, the Internet has been establishing itself as a privileged means for information publishing, communication and entertainment. It is forecasted that number of Internet-enabled mobile phones using WAP and its successors is soaring from 1.1 million at the end of 1999 to somewhere from 21 to 80 million worldwide in 2003 [12]. Travelers' behavior is changing along with the technology development. Nowadays, in the field of traveling, people are looking for easy and portable terminals for access to an on-line system capable of offering information relevant to the user's position in a user friendly way. Traveling Information Systems are characterized by distributed information sources, geographic information, mobile and non-expert users and languages issues. As information often has geographic attributes, access to that information must take into account user specific information, such as user position. In travel application, every traveler's position is different, and their requirements are different as well, they require different services like restaurants information, route planning or site search. In most services that travelers require, their geographic location is a key parameter for feedback result. Geographic location correctly handled and combined with other user preferences, can dramatically improve access efficiency and friendliness.

While bring the Web Service technology to travel applications, application developers can use specific Web Service functionality and do not have to maintain GIS application tools or the associated spatial data at the client side. Also, developers can easily combine several Web Services to create one complete solution. With this kind of approach, it benefits a great deal to application implementation. The time to develop an application will be shorter, and the Web Service can be shared by many custom applications. Also, many applications can be combined and to form a more complete application. For travelers, it means that they can get more suitable services from an application and they do not need to search from different sources for the information.

Access to information may be achieved either through dedicated information appliances installed in urban environments, e.g. a device on a bus stop with the respective timetables, or through the use of personal networked devices, e.g., a mobile phone. In the latter case, the effect of mobility may also be explored to support location-dependent applications (i.e. systems in which the behavior is adapted and mainly driven by changes in location). By adopting Web Service specification standards, it will be possible for browser-based ("thin") and desktop-deployed ("thick or full-functioned") applications to draw on the resources of geo-processing servers distributed across the Web. Web Services use data and related functionality to perform basic processing tasks such as address matching, map image display, and routing. As application developers, one can use Web Services to perform real-time processing on computers where Web Services are located and return the results to the local application all over the Internet. One does not have to maintain GIS application tools or the associated geographic data on the local system to use them in the custom application. Web Services can communicate with any web-enabled application, so as to mobile devices which are capable to communicate with web (Based on its wireless communication protocol functionality).

2.2. Web Service

A Web Service is a software application identified by an URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artefacts. A Web Service supports direct interactions with other software agents using XML-based messages exchanged via internet-based protocols [15]. Put it in a simple way, a Web Service is something one can call over the web from a program. Many organizations and companies (such as Microsoft, IBM, Sun and etc) are specifying the interoperability interfaces that enable software vendors to deliver services in the Internet's open, standards-based environment. For GIS application, this will make publishing, discovery, access and use of spatial data and geo-processing resources much easier and less expensive. For end users, it may not make any difference: "gets an input and displays the result". However, for developers/programmers, Web Services make their life easy.

For the end user it may not make any difference: "gets an input and displays the result". However, for developers/programmers, Web Services makes their life easy. The main point is that through Web Services you can achieve platform independency as well as language independency. For example, instead of writing COM object if one exposes one's methods (functionality) through Web Services using ASP.NET on Windows, it can access those methods in Java on Sun Solaris/Linux machine, as shown in figure 2-1 below. Because Web Services works on XML technologies, it makes developers life simple.

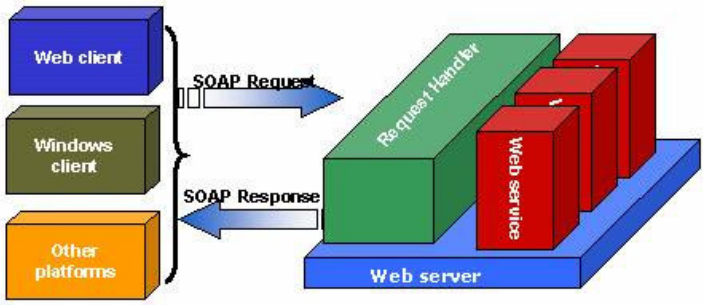


Figure 2-1: Typical Web Service architecture.

Web Services use XML to provide an application implementation of distributed application services and information based on multi-platforms and multi-languages. What this means is that applications and people can use the data and services of an application running on a remote computer regardless of differences between the application server technology running on the remote and local machines. Web Services make it possible for diverse applications to discover each other and exchange data seamlessly via the Internet. While it is perfectly feasible for a person to use a browser to take advantage of a Web Service running on a remote computer, the real potential of Web Services lies in its ability to allow applications to work with other applications without the need for any human intervention.

Unlike other technologies, such as Sun's Enterprise JavaBeans, and Microsoft's DCOM, Web Services are platform- and language-independent. They also don't rely on complex language translations like CORBA. Web Services are independent because the messages sent between applications are in an XML-based protocol (called the Simple Object Access Protocol or SOAP). This allows applications

written on one application server platform, such as ASP. Net, to be used by applications written in other platforms, such as ColdFusion or Java.

2.2.1. XML and SOAP

Although there are other XML-based Web Service protocols out there, such as XML-RPC, Jabber, and ebXML, SOAP seems to be the most popular one used with Web Services today. A discussion of the advantages and disadvantages of these competing can be found in XML Protocol Comparisons [16]. SOAP is a protocol that describes how messages are sent to and from Web Services.

SOAP is a simple XML based protocol to let applications exchange information over HTTP [16.1]. Within a SOAP framework, a client sends a request in XML over HTTP to the Web Service; in return, the Web Service sends a response to client in XML as well. In this way, it does not matter what language is used to create the request or response as long as it is wrapped in XML. This flexibility allows developers to integrate components built in different programming languages.

The Place Finder Web Service (developed by ERRI) provides is a good example of how SOAP works. Place Finder Web Service includes the following method:

Locationinfo findplace (string plave)
Returns the x, y location for a place name in any part of the world. [16.2]

If a similar "find a place" application was stored locally, a client could use a library and invoke one of its objects. The difference with the Place Finder Web Service is that the component (application tools and spatial dataset) is remote and across the Internet. Therefore, something must happen behind the scenes to transfer the findplace request to the remote object, and the remote object must somehow return the result. Without SOAP, a client would have to create an XML request, send it in a POST message to a URL over HTTP.

2.2.2. WSDL

A Web Services descriptor language (WSDL) file is an XML document that describes the arguments accepted by a specific Web Service as well as the methods and properties returned by that service. The WSDL specification provides the grammar and syntax rules for WSDL files. As the name implies, these files are used to describe the Web Service to potential consumers of the Web Service.

2.2.3. UDDI

Universal Discovery, Description and Integration (UDDI) is the means by which potential users of Web Services can locate and choose the Web Service that best meets their needs. UDDIs can be thought of as the marketplace for Web Services. They are the web sites where Web Service creators can offer and market their services and where users can locate the Web Service that meets their needs.

UDDIs provide information on the organizations providing Web Services, categorized lists of services registered with the UDDI.org web site and specific information on each service listed on UDDI.org.

Currently, there are three main public UDDI services on the web, but this number may change in the future. These include xMethods, the IBM UDDI and the Microsoft UDDI. There are numerous private UDDIs that are not publicized on the web as well. Software such like Dreamweaver MX gives you the ability to add or remove UDDIs from the Dreamweaver interface at any time.

2.2.4. Wireless Communication

In fact, for a web application, the set of protocols (include XML/SOAP, UDDI and WSDL) are enough for a web application implementation. But if want to use this kind Web Service architecture for mobile application, a wireless communication protocol is needed for connecting web application with mobile devices. Currently, there are several wireless communication protocols available, such like WAP (wireless application protocol), GPRS (General Packet Radio System), CDMA (Code Division Multiple Access) and etc.

Among these wireless communication protocols mentioned above, due to WAP emerged earlier than others, WAP has a relative closer relationship with W3C. WAP protocol is the leading standard for information services on wireless terminals like digital mobile phones. WML is used to create pages that can be displayed in a WAP browser. By using WAP and WML, it enable convert your HTML pages to pocket format, so that your information can be accessed from devices like mobile WAP Phones.

2.3. Web Service Architecture

Traditionally, a rich Windows client uses DCOM (Distributed Component Object Model) to communicate with the server and invoke remote objects; DCOM was developed by Microsoft for Windows Operating Systems. DCOM is an extension of COM (Component Object Model), It supports objects distributed across a network, much like IBM's DSOM protocol, which is an implementation of CORBA [17]. It is extremely complex that configure DCOM to work properly in a large network. Traditional systems architectures incorporate relatively tightness coupling between various components in the system. The traditional IT systems, including Web-oriented systems, can be characterized as tightly coupled applications and subsystems. Monolithic systems like these are sensitive to change. A change in the output of one of the subsystems will often cause the whole system to break. It is necessary to replace the current models of application design with a more flexible architecture, yielding systems that are more amenable to change.

Today’s application development is definitely shifted towards thin, browser-based clients, that because such clients eliminate the high costs of deploying an application to the desktop. Desktop applications are costly to deploy partly due to the issues of installing and configuring the application and partly due to the issues of communicating between the client and the server. An ideal solution to the

client-server communications problem is to use HTTP as the communications protocol. HTTP is a good choice because any machine that can use a Web browser is by definition running HTTP.

Despite HTTP's huge success as the Internet's application protocol, it is limited to fairly simple commands centered on requesting and sending resources. The result is that today we have millions of interconnected computers that leverage the Internet primarily for browsing the Web but cannot, despite of the connectivity, freely exchange data between applications. SOAP proposes to solve this problem by defining a standard protocol that any application can use to communicate and exchange data with any other application. SOAP enables application-to-application communication over any transport protocol including TCP. SOAP can be used over HTTP to enable application-to-application communications over existing Internet infrastructure with its firewalls and proxies.

The Web Services architecture describes three roles: service provider, service requester and service broker; and three basic operations: publish, find and bind. A network component can play any or all of these roles [18]. The current trend in the application space is moving away from tightly coupled monolithic systems and towards systems of loosely coupled, dynamically bound components.

The Web Services architecture describes principles for creating dynamic, loosely coupled systems based on services. There are many ways to instantiate a Web Service by choosing various implementation techniques for the roles, operations, and so on described by the Web Services architecture. The idea of Web Services is similar to Web content (in terms of simplicity of publishing and accessing applications), there are mechanisms for application providers to publish descriptions of their service and for application users to search for Web Services. WSDL is the mechanism for publishing the Web Service name. UDDI is a way to search for services.

Web Services in terms of a service-oriented architecture. Web Services reflect a new service-oriented architectural approach, based on the notion of building applications by discovering and orchestrating network-available services, or just-in-time integration of applications [18]. As depicted in Figure 4, IBM introduces this architecture; it sets forth three roles and three operations. The three roles are the service provider, the service requester, and the service registry. The objects acted upon are the service and the service description, and the operations performed by the actors on these objects are publish, find, and bind.

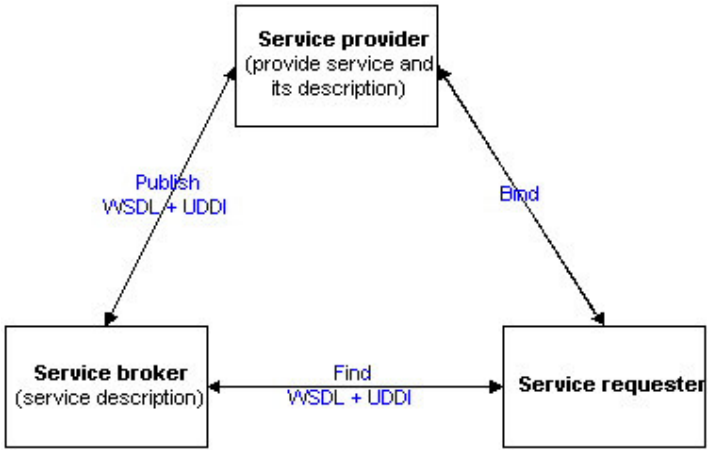


Figure 2-2: Basic Web Services Architecture

Service provider:

A service provider creates a Web Service and its service definition and then publishes the service with a service registry based on a standard called the Universal Description, Discovery, and Integration (UDDI) specification.

Service requester:

Once a Web Service is published, a service requester may find the service via the UDDI interface. The UDDI registry provides the service requester with a WSDL service description and a URL (uniform resource locator) pointing to the service itself. The service requester may then use this information to directly bind to the service and invoke it.

Service broker:

We have a situation in which multiple organizations are requesting information from multiple Web Service providers. A broker is a central medium that makes the information transfer happen: It is a common gateway/address for client applications to access a wide variety of services. It is a standard method for transforming applications into Web Services. These Web Services may require a broker to interact with only one server application or multiple server applications. Brokers perform the following functions:

- Receive SOAP requests in XML format from client applications.
- Authenticate the request and check for authorization.
- Publish all the services offered, or cancel the Web Service as service provider request.

In general, there are two different Web Services for consuming; they are public Web Service and Restricted Web Service.

- Public Web Services

No authentication steps are required; Web Services are accessible to anyone who accepts the license agreement. Users must agree on the intended use of the service. For example, the doGoogleSearch (a Web Service developed by www.google.com) Web Service count how may people viewed a web page) may be used freely in low-volume, non-commercial Internet applications. It cannot be resold as is or as part of a custom application without written permission from Google [35].

- Restricted Web Services

Restrict to users who have permission to access the service. Due to the sensitive nature of the information contained in them, some services are encrypted to users. Only authorized users have access to Restricted Web Services. These users may identify themselves with a user name and password. Authentication is done through the Authentication Web Service.

Example: Explain How Web Service's Components associate with each other

Problem: Check what restaurants are available within 2 kilometers; a list of local restaurants with their addresses is sending back. The restaurants Finder Web Service is an example of a simple service returning a single response. It takes a single integer as its input. "2 kilometers" for example and outputs basic restaurants location information.

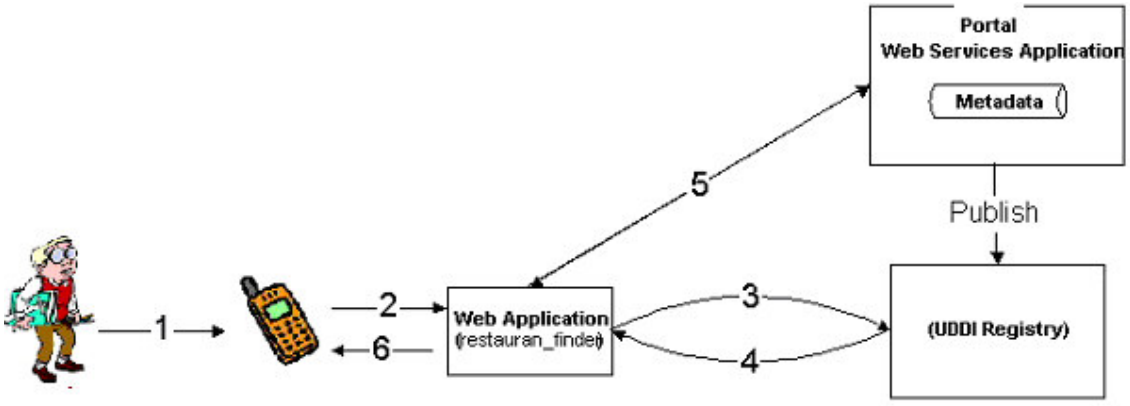


Figure 1-3: Explanation how to implement a restaurant search application

Step1: Through the mobile device, the traveler input the position information (based on mobile device's GPS functionality) and combined with traveler's specific request (Restaurants within 2 kilometers).

X, Y coordinates: A longitude (x) and latitude (y) coordinate of the current location.
Buffer distance: The distance from current position, 2 kilometers, e.g.

Step 2: The request send to web application. The communication between mobile device and portal should be use wireless communication protocol (such protocols like WAP, GPRS and BlueTooth are all capable.)

Step 3: Find the specific Web Service Restaurant_finder in portal. As application developers, you can use Web Services to perform real-time processing on the computers where Web Services are located and return the results to your local application all over the Internet. You do not have to maintain GIS application tools or the associated geographic data on your local system to use them in your custom application.

Step 4: Web Services are published on the portal (UDDI registry), a universal database of Web Services. Developers can search any UDDI site to discover services on the registry, making UDDI a powerful resource for publishers and consumers alike. The Restautants_finder Web Service looks like the following method:

Restautants_finder (integer DufferDistance) [11]

To use this Web Service, the client invokes the “Restautants_finder" methods, passes in a place name, and receives a location for the place.

Step 5: Through the portal, web application located the Restaurant Finder Web Service, and bind with Restaurant_finder that provided by a specific Web Service provider.

Step 6: The Restaurants_finder Web Service takes an integer and coordinates data, which defines the area you want to search (in latitude/longitude coordinates). The return is a location Information object that contains an array of location objects with the following content.

Name:	Restaurant name
x,y coordinates:	A longitude (x) and latitude (y) coordinate of the restaurants location.
Distance:	The distance from current position, smaller than 2 kilometers.
Score:	A number from 1 to 10 indicating the rank of restaurants.
Type:	A letter indicating the type of restaurants, for example, C=Chinese.

2.4. The Web Services programming stack

Web Services programming stack is a collection of standardized protocols and application programming interfaces (APIs) that lets individuals and applications locate and utilize Web Services [19]. In each layer of the Web Services stack, standards enable a Web Services client to speak to an application server or middleware component such as Common Object Request Broker Architecture, Java 2 Enterprise Edition or .NET. Clients also must have standard ways to discover what servers can do and how to communicate with them. Increasingly important roles will be played by emerging Web Service technologies, including Web Service Description Language; Universal Description, Discovery and Integration (UDDI).

Web Services applications are built on an architecture or software system design that can be illustrated as a "stack" of processing layers. The software components in these layers are "loosely coupled" components that interact with one another via standard protocols or libraries of subroutines that have standard interfaces. At the core of the Web Services stack there are the "tried-and-true" standards on which the Web is built today: TCP/IP, HTTP, HTML and XML.

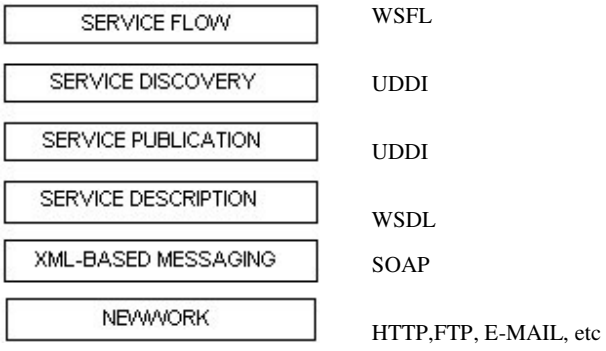


Figure 2-3: Web Services programming stack

The network is the foundation layer for the Web Services programming stack (Figure 2-3). All Web Services must be available over some network. The network is often based on an HTTP protocol.

On top of the networking layer is an XML-based messaging layer that facilitates the communications between Web Services and their clients. The messaging layer is based on SOAP. SOAP is an XML protocol that facilitates publish, find, bind, and invoke operations described previously. [20]

WSDL is a specification that describes available Web Services to clients. These descriptions take the form of XML documents for the programming interface and location of Web Services. [21]

The three layers described above are required in order to have interoperable Web Services. These layers also create a low-cost entry for leveraging Web Services by allowing these services to be deployed over the Internet. The remaining layers in the programming stack are optional and will be used based on specific requirement.

Publication of a service is really any action by the service provider that makes the WSDL document available to a potential service requester. Sending the WSDL (or a URL pointer to the WSDL) as an e-mail to a developer is considered to be publishing. Publishing is also advertising the WSDL in a UDDI registry for many developers or executing services to find. [22]

Likewise, discovery of a service is any action that gives the service requester access to the WSDL for a service. The action may be as simple as accessing a file or URL containing the WSDL or as complex as querying a UDDI registry and using the WSDL file(s) to select one of many potential services. The service flow layer of the stack facilitates the composition of Web Services into workflows and the representation of this aggregation of Web Services as a higher-level Web Service. Standardization activity at this level is ongoing, but IBM has produced the Web Services Flow Language (WSFL) as its input to the standardization process. [23]

2.5. Existing GIS Web Services

So far, Web Services technology still new, but Web Service is the rapidly emerging paradigm for World Wide Web-based distributed computing, and it can be tailor made for organizations that produce and use geo-spatial information. Due to Web Service technology's strong capabilities, it will be sure more and more application will adopt this technology. Following are some examples of current Web Services which for GIS application.

2.5.1. OGC Web Services 1.1 Initiative (OWS-1.1)

OWS-1.1, an Open GIS Consortium (OGC) Interoperability Program Test bed, began with a kick-off on September 24, 2001 and ended with a demonstration on March 7, 2002[24]. As an ongoing project,

OGC is developing Web-based geo-spatial services through the OGC consensus process and provide geographic information and services accessible via the Internet. Each service supports a certain GIS operation function that returns metadata describing the nature of the service, the methods to register your own Web Service, the addresses to access the service or to contact its provider, the type of data provided by the service, and so forth.

2.5.2. ERSI ArcWeb Service

ArcWeb Services offer a way to include GIS content and capabilities in your applications without having to host the data or develop the necessary tools yourself. The result is significant savings of time, expense, and computer resources. [25] ArcWeb Service enable combine multiple services and integrate them with your own application environment, leading to limitless possibilities for sharing geographic information. Applications can access Web Services through Web protocols such as HTTP and XML, without concern for how each service is implemented. [26] Application developers can combine Web Services, or use them with other tools, to perform a larger function or provide a complete solution. Figure 2-4 showing what ArcWeb Web Services are available so far.

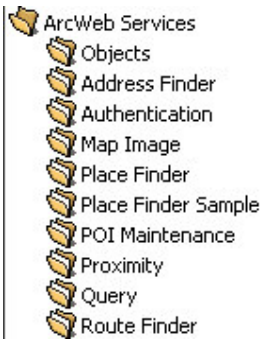


Figure 2-4: Available ArcWeb Services

ESRI's ArcWeb Services are collectively a type of Web Service that provides commercially hosted spatial data and geographic information system (GIS) functionality via the Internet to ArcGIS and custom Web applications. With ArcWeb Web Services, developers can include GIS content and capabilities in their applications without hosting the data or developing the GIS applications themselves. But so far most data and services are for USA domestic users. The result is significant savings of development time, expense, and computer resources. ArcWeb Services are available as prepackaged collections of related services or as individual services. They are developed using Web protocols so that different computers can easily communicate with each other.

ArcWeb Services communicate with any local application that can connect to the Web. ArcWeb Services, like many Web Services, use SOAP to transfer information back and forth to clients (local applications). SOAP brings together two industry standard languages for communicating over the Internet: hypertext transfer protocol (HTTP) and Extensible Markup Language (XML). [27] ArcWeb Services use SOAP to communicate so they are compatible with many Web Services toolkits.

2.5.3. Microsoft MapPoint .NET Web Service

Microsoft MapPoint .NET Web Service [28] is Microsoft's newest version of its platform for mapping and location services — delivers easier programming, a wider range of geographic coverage and the industry's most comprehensive feature set for developing a broad range of "location enhanced" applications, including wireless/mobility, customer relationship management, enterprise location services, business intelligence and many others. Web Services consumers could access Microsoft MapPoint .NET Web Service through:

http://staging.mappoint.net/standard-30/mappoint.wsdl

Figure 2-5 shown what Web Service Microsoft MapPoint .NET provided, it is viewed with XMLSpy software.

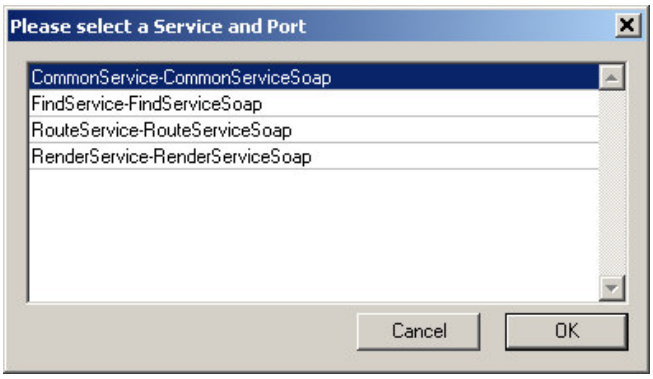


Figure 2-5: Microsoft MapPoint .NET Web Service

Microsoft MapPoint .NET is a hosted, programmable XML Web Service that allows application developer to integrate high-quality maps, driving directions, distance calculations, proximity searches, and other location intelligence into your applications, business processes, and Web sites. By using same Microsoft MapPoint .NET Web Service shown in figure 2-5, application developers can build many different customized application, as two different application shown in figure 2-6:

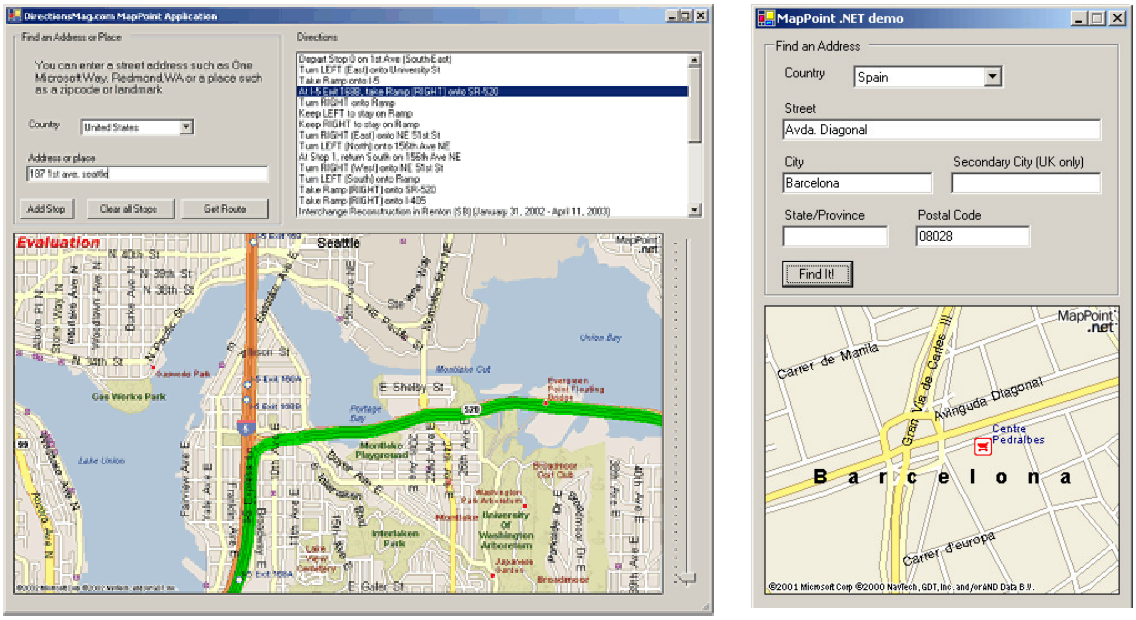


Figure 2-6: Different Applications which using same Web Service

2.6. Web Services' benefits and Real-worlds scenarios for using

2.6.1. Benefits of Web Services

- Web Services are easy maintenance. Web Services are located in Web Server side, won't conflict with client computer's setting and installation problems.

- Web Services are discoverable. Universal Description, Discovery and Integration (UDDI) is an open standard with broad industry support standard. When an implemented Web Service exposed in any Web Services portal, it can be discovered in any Web Services portal too.

- Web Services are self-describing. Once a Web Service is discovered, the developer can begin using it immediately. All they need is the full URL path to the services WSDL. Each method, parameter, property, and return value of the service is described in a standard way, allowing modern development tools to immediately allow access to the exposed functionality. Visual Studio .NET and XMLSpy are two of such development tools.

- Web Services conceal complexity. Web Services embed its complex data processing within itself in server side. Client side application developer make use of Web Services through standard interface which described in Web Services WSDL, therefore the application developer don't need dealing with Web Services internal process.

- Web Services are very independent. Device independent. Programming language Independent. Platform independent. Due to this character, Web Services are not Just for Web applica-

tions. A Web Service is just as comfortable as an ingredient in standard Windows applications, as it is in a browser-based Web application. Handheld devices, in-car telematics, Tablet PC's and any other platform or device that can make calls to a Web Service can benefit.

2.6.2. Real-worlds scenarios for using Web Services

This section showing you real-world scenarios where you can realize significant benefits by using Web Services, and other scenarios where it makes no sense to use Web Services and you really should not use them.

When To Use Web Services

- Communicating Through a Firewall. When you build a distributed application with hundreds or thousands of users spread over many locations, there's always the problem of communicating between the client and the server because of firewalls and proxy servers. Currently, due to security reason, most of web server only allows HTTP access.

- Software Reuse. Software reuse can take place in various forms and at different levels. The most basic form of code reuse is through reuse of source code modules or classes. Another form of code reuse is binary component-based reuse. There's a significant market today for reusable software components such as grids and other user interface. But software reuse has always been limited by one key factor: You can reuse the code but not the data behind the code. The reason for that is that you can easily distribute components or even source code, but you cannot easily distribute data unless it is fairly static data not expected to change much.

When Should Not To Use Web Services

- Single Machine Applications. There are still many desktop applications for business and personal use. Some of those applications might need to communicate with others running on the same machine. In this case, it is almost always better to use a native API rather than a Web Service. Software component technologies such as COM and .NET class libraries or components are very effective in this scenario because using such components requires relatively little overhead.

- Homogenous Applications on a LAN. One advantage of Web Services is programming language independent and platform independent, Web services will bring you benefits when you building application across many different programming languages or platforms. For example, you might have two server applications wanting to communicate with each other or, more commonly, a Win32 or Windows Forms client that wants to communicate with its server counterpart on the same LAN. It's much more efficient for two such applications to communicate using DCOM rather than SOAP/HTTP.

2.7. Summary

It may sounds complex, actually the Web Services revolution simplifies application development. The use of standard interfaces reduces complexity and time to implementation, supports multi-vendor "plug and play," and reduces product life-cycle risk. It's also good for the market, because it increases choice and reduces dependency on the more traditional single-vendor, monolithic application approach. This rapidly emerging paradigm for World Wide Web-based distributed computing is tailor made for organizations that produce and use geo-spatial information. Many GIS organizations and companies are specifying the interoperability interfaces that enable software vendors to deliver geo-processing services in the Internet's open, standards-based environment. This will make publishing, discovery, access and use of geo-spatial data and geo-processing resources much easier and less expensive. Also for the application developers, due to Web Service's dynamic, loosely coupled characters, they can implement the application much quicker.

However, Web Services are no heal-all medicine and you should not use them just because you can. There are still some limitations of Web Services technology; there are certain scenarios where using Web Services will cost you performance without buying you any benefit. For example, homogenous applications running on the same machine or on different machines on the same LAN should not use Web Services to communicate with one another, the connection speed will affect the efficiency of application; other one is the implemented application must to bind with Web Service providers and brokers, developer can't control the application performance totally.

When application developers are trying implement proper applications for foot-travelers, dynamic data storage and software selection are real challenges. To help application developer for design foot-traveler application, the suitable technology of the latest is "Web Services". In this research, I will implement several GIS Web Services that capable in solving some foot-traveler Geo-related problems.

Chapter 3.

Foot-traveler Services' classification and consequence Web Services design

Different types of travelers have different travel service requirements in certain degree. This implementation is trying to develop a solution for foot-travelers who are equipped with mobile devices and positioning functionality. Section 1 presents an analysis about what foot-traveller's requirements are and classify those requirements into a series of major categories. The classified requirements will facilitate the consequence Web Services design (refer to section 3). The end of this section there is a description of what a foot-traveller could achieve with these services. Section 2 discusses about different methods to implement Web Services; issues like developing Web Services with or without Web Services toolkit, what toolkits are available and a comparison of their advantages and disadvantages. Section 3 defines different Web Services with their consequence foot-traveller specific problems. Section 4 presents some considerations for implementing this kind of applications in reality.

3.1. Foot traveler requirement classification

Foot-travelers need location dependent information such as local yellow pages, traffic reports, directions, shopping information, local advertising etc. Information provided to Foot-travelers will change with the location of the user, i.e., vary in space. There are many examples such as national boundary data, which is valid at a nation's scale, while weather information will be relevant over smaller geographic regions, and traffic reports are of use at a more local level.

Yellow page:
A volume of telephone directory that lists businesses, services, or products alphabetically according to the field of service. There are many yellow page services existing in both national scale and local scale, even on Internet.

Traffic report:
Road network's traffic status information should be dynamic. When traveler is preparing the trip or during the trip, they could check the traffic status and make their judgment.

Direction:
When a traveler wants to visit a certain place, he needs to know the direction instruction first. The direction instruction should include the route, distance or transportation method information.

Shopping information:
Find local shopping opportunities. Many travelers like to buy some stuff related with their traveling activities, such as local specials, equipment for traveling.

Local advertising:
Temporary event and activities will hold in local, such as local special parties. Because the main purpose of travelers is to experience the scene or culture difference, the local focused event and activities will be extremely welcome.

Position of foot-traveler can be determined either with GPS attached to mobile devices or with mechanisms for identifying end users within a cell of wireless network. The information can be organized as a collection of pages, with a hyperlink-based "Web browser like" interface on the end users' side.

3.1.1. Service Categories definition

As a foot-traveler who does not know much about a city that he/she is in or plans to go, he/she will need many specific services in many different scenarios. Imaging myself as a foot traveler being the first time to visit Enschede (A small city of Netherlands) for the first time, I would have to prepare for the trip carefully. Before I reach Enschede, I need to check the weather information of Enschede for early preparing. When I arrive at Enschede, I need to have a suitable hotel to stay. When I have settled down in a hotel, I need to find a restaurant to eat. In addition, as a traveler, I want to visit some local attractions. In some circumstances, I want to be informed in time if I deviate from my route. Travelers need many different services, based on every service's character; different services can be classified into four major categories: Information services, Interesting Places Search, Road Navigation and Monitor Services. As shown in figure 3-1:

The definitions of Information services, Interesting Places Search, Road Navigation, Monitor Services are as following:

- Information services

Provide user with relevant information that are related to user's location.

Example: Traveler search for local events of Enschede, he may just input "500" (500 meters buffer distance), and all local events information (within 500 meters away from traveler) will send to traveller.

- Interesting Places Search

Provide user with the location information of his interesting place (nearest one) and his current position, it will help user to make his judgment for his planning.

Example: A traveler search for the location information of ITC. All he needs to do is to input "ITC", and the parcel dataset should contain the owner name of parcel, and coordinate of parcel.

- Road Navigation

In complex road network, based on user's location (or the location he defined) and the destination position, system generates a possible route solution to guide user to reach destination. In addition, user may add his preferred condition for output, such as shortest distance or must pass certain point.

Example: Traveler search for the route guidance from ITC to DISH hotel, all he need is input "DISH hotel" as destination. To implement this service, pavement network dataset is required by system, in this network, node information and length of two nodes (directly linked) are required, for the purpose of decide if a node is end node or not and different route result comparison.

- Monitor Services

User defined certain conditions previously, such like weather change (rain, snow), activity region boundary etc. Whenever those conditions occur, the system will give user an alert (or warning) with description.

Service Category	**List**
Information Services	▪ Weather forecast ▪ Local news ▪ Airlines ▪ Airline Ticket Agencies ▪ Airport Transportation & Parking Services ▪ Boat Excursions ▪ Cruises ▪ Ferries & Water Taxis ▪ Bus Lines ▪ Mass Transit ▪ And etc
Interesting Places Search	*Food & Stores* ▪ Bakers & Bakeries ▪ Cafeterias ▪ Candy & Confectionery ▪ Cheese & Dairy ▪ Convenience Stores ▪ Delicatessens ▪ Farmers Markets ▪ Foods - Carry Out ▪ Fruits & Vegetables ▪ Grocers & Supermarkets ▪ Health & Diet Foods

	▪ Ice Cream & Frozen Desserts ▪ Kosher ▪ Meat ▪ Natural & Organic Foods ▪ Produce ▪ Seafood ▪ And etc Government & Municipal Services ▪ Consulates & Foreign Offices ▪ Courts ▪ Embassies ▪ Police Departments ▪ Post Offices ▪ Public Libraries ▪ And etc Medical, Health, & Personal Care ▪ Ambulance Services ▪ Disability Services & Organizations ▪ Doctors & Practitioners ▪ Hospitals, Clinics, & Social Services ▪ Medical & Health Goods ▪ Personal Care ▪ And etc *Accommodations* ▪ Bed & Breakfast Accommodations & Inns ▪ Campgrounds ▪ Guest Houses ▪ Hostels ▪ Hotels & Motels ▪ Resorts & Vacation ▪ And etc *Tours, Information, & Places of Interest* ▪ Amusement & Water Parks ▪ Aquariums ▪ Bicycles - Tours & Renting ▪ Botanical Gardens ▪ Fishing Guides, Charters, & Parties ▪ Gambling & Casinos ▪ Golf Courses - Miniature ▪ Historical Places ▪ Museums

	▪ Parks ▪ Recreational Trips & Tours ▪ Sightseeing Tours ▪ Tourist Attractions & Information ▪ Tours - Operators & Promoters ▪ Travel Agencies ▪ Zoos ▪ And etc
Road Navigation	▪ Road Planning (based on distance) ▪ Road Planning (based on cost) ▪ Road Planning (based on user defined conditions) ▪ Traffic Warning ▪ Road Side Services ▪ Taxicabs & Transportation Services ▪ And etc
Monitor Services	▪ Check other traveler position ▪ Record traveler's track ▪ Safety service based on user definition ▪ Warning based on location ▪ Warning based on weather information ▪ Reference position record ▪ And etc

Figure 3-1: Traveler service category

3.1.2. Different services for different scenarios

The classified services (as shown in the figure 3-1) expects be able fulfil traveler's requirements. Those requirements such as weather information, restaurant services, locations and so on. With the help of Mobile GIS application and category services, figure 3-2 shows a description of my first journey in Enschede; different services will be required at different stages of scenarios:

Time	Scenario	Service
8:00	Step 1:Before I reach Enschede, I need to check the weather information in Enschede for early preparing.	Information Services
10:00	Step 2:As I arrive Enschede, I search for a hotel that within 5	Interesting

	minutes walk (500 meters) from railway station; the price ratio should be lower than 40 euro per day. As the result, I found the Mexican Hotel (in downtown area) that fits my requirement.	Places Search
12:00	Step 3: When I settle down in Mexican Hotel (I record the hotel coordinates information as a reference point in my mobile device), its time for lunch and I am getting hungry. I search for a restaurant that within 500 meters and provide Chinese dish. Its turn out that several restaurants fit my requirement, and I select one and make a reservation.	Interesting Places Search
14:00	Step 4: After I enjoyed my lunch, I fell full of energy again. I want to visit some local interesting places, the system tell me what are available in Enschede, such as museums, theaters, parks and etc. Because I want to know something about Enschede history, I select a museum, then the system tell me the museum is how far from my current local position, and generate a simple route map for my navigation guide.	Interesting Places Search, Road Navigation
18:00	Step 5: In the later afternoon, I want to see the night view of Enschede, but without specific purpose. As I knew Enschede is very near to Germany border, it will cause me unnecessary trouble if I cross national border without relevant identification. So I set up a warning function to my mobile device, to monitor where is my dynamic position from time to time, the system will give me warning signal if I am very near to the border.	Monitor services
20:00	Step 6: Because I have walked too far, I cannot remember the way back the Mexican hotel. Using the hotel coordinate data (refer to scenario 3) and current position (mobile device's position functionality); system generates the simple route map for me to go back.	Road Navigation
22:00	Step 7: In the evening, I lying on the couch and enjoy the memory of a one-day's journey in Enschede. Later I send today's traveling route data to my friends, and let them share my experience in a certain degree.	Monitor service

Figure 3-2: One day's journey in Enschede

3.2. Web Services Development and Consumption Environment

To create or access Web Services, the application developers have many different approaches. Application developers do not have to use a Web Services toolkit to create or use a Web Service. Because SOAP is simply an XML based protocol, it is straightforward to use the SOAP protocol with an HTTP POST, but it would be more complicated than using a toolkit.

Most Web Services toolkits are relatively new, and some are not fully functional [29]. Developers may be worry of using these new products on their production sites until they feel confident that they be stable and compatible with their systems. Currently, there are many Web Services toolkits, such as IBM WebSphere [30], Microsoft Visual Studio .NET [31] and GLUE [32]. Although they all provide solutions based on open standards, such as XML, SOAP, WSDL, UDDI (baseline specifications), there are many differences among them, for example, framework, programming languages, run time, service discovery, terminology and so forth.

In general, it is probably a matter of personal preference which tools to choose for development. Compare IBM Websphere with Mircosoft's .NET, one obvious difference is IBM Websphere supports Java as the primary language of choice, while Microsoft allows language independence, which means that one can choose the language of choice for coding, as long as .NET supports the language. Currently, .NET accommodates many languages, such as C#, J#, VB.NET, Jscript, C++, Perl and COBOL. Though Microsoft introduces Visual J# as a java-compatible language, it does not support Java [33].

One of Web Service's advantages is that Web Service allow application developer achieve platform independency as well as language independency. No matter what kind programming software or application software, if it has capability connect with Internet (support HTTP protocol), it will be allow consuming Web Services.

Overall, there are platform and server issues. If an organization were working with a WebSphere/Apache environment, then Application Developer would be the tool of choice. If an organization uses Microsoft products (such as BackOffice, Internet Information Server, Windows 2000), then Visual Studio .NET has advantages which leverage the existing infrastructure [3-6].

For the project's implementation under consideration, as the author is working on Window 2000 platform and his time is constraint, it is convenient for him to develop this project with VB.NET (has similar interface and syntax as VB, characterized by quick application developing).

3.3. Defining the Web Services for corresponded foot-traveler specific problems

To implement these services, there are different requirements for each type services with respect approach. For example, weather information service needs region's name as input; restaurant search service needs to indicate buffer distance and type of food and require restaurant dataset has corresponded information.

Basically, the principle of the client side application communicating with Web Services is: Client side sends SOAP request message (over HTTP) to Web Service, and then Web Service processes the request and replies a SOAP response message (over HTTP) to client side, as shown in figure 3-3 below:

Figure 3-3: Communication between client side and Web Service

In detail, client side and Web Service's structure display separately as figure 3-4 below. The function of the UML design is to make the relationship of different classes clear, which is very useful for later implementation.

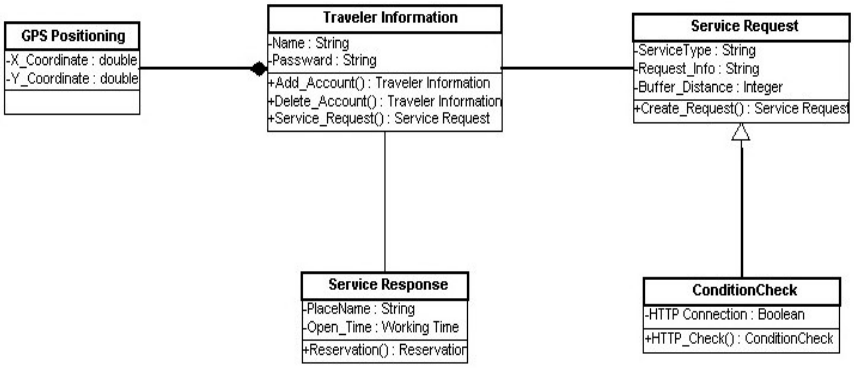


Figure 3-4: Client side class diagram

To make foot-traveller's search easier, many different services are classified into four major service categories, as shown in figure 3-5. Figure 3-5 is the UML diagram designed to illustrate the general requirements of different type services: Information Services, Interesting Places Search, Road Navigation and Monitor Services.

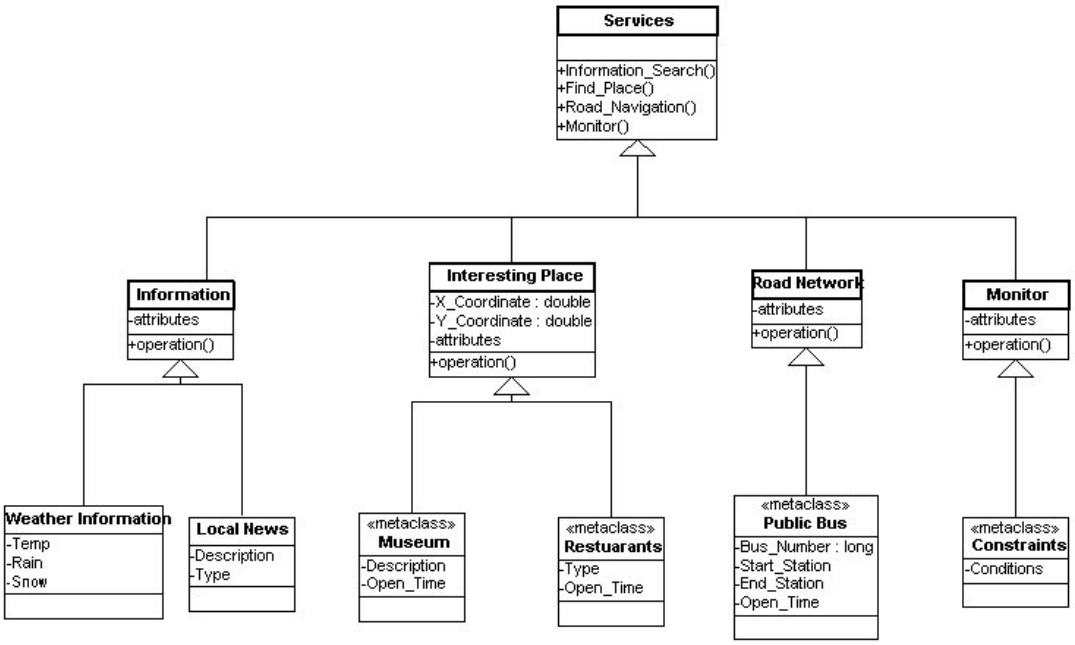


Figure 3-5: GIS Web Services class diagram

Note: Both UML diagram figure 3-4 and figure 3-5 are conceptual design, they provide better understanding for the require parameters and classes for later implementation. Because my dataset don't fit with above UML diagrams, I only basic structure of those UML diagrams but not detailed classes building.

Illustration: How to make use of UML in application designing

Below give a detailed description of Interesting Places Search service with an example respectively.

§ Interesting Places Search

Provide user with the location information of his interesting place and his current position, it will help user make his judgment for his planning.

Example: Traveler search for the location information of Chinese restaurant within 2 kilometers, all he to need is select "Restaurant Service" operation, and add buffer distance (2 kilometers) and restaurant type (Chinese) as constraints information. The whole procedures of request information passing and processing as show in the workflow diagram below:

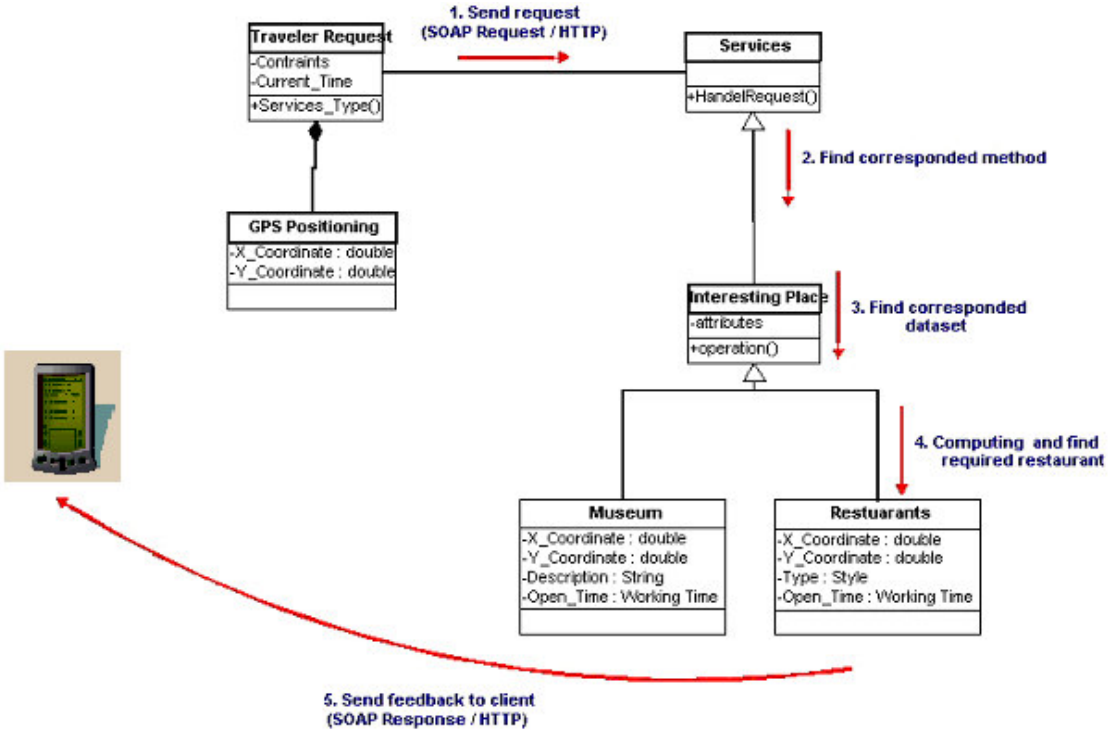


Figure 3-6: Workflow of Interesting Place Service

Step1: User sends his request by using his mobile device; user's location information and request service information contained in Request SOAP envelop and send to Web Service portal over HTTP protocol.

Step 2: The request send to Web Service portal. The function of portal is to find the specific Web Service (use pointed) and bind with it.

Step 3 and step 4: The Interesting place search service will process user's request with restaurants dataset. If the restaurant fit to user's constraints (such like buffer distance and restaurant type), it will be selected and send to user.

Step 5: The selected restaurants information will send to user. Again, the information will be contained in Response SOAP envelop and send to mobile device over HTTP protocol.

As result, user will get an image on screen, which shows both his location and restaurant location with distinguish sign respectively. In additional, a pavement network (contained in image) will help him find the way to reach restaurant.

3.4. Some considerations for application implementation

3.4.1. Accuracy

On-site positioning accuracy is very important. The accuracy level will affect the performance greatly. For general GPS positioning application, the deviation error is about 5 to 10 meters. Based on this error level that a traveller searches for the information of local weather condition is okay, but it may not necessarily be okay if he searches for a restaurant with a short distance. Therefore, as the services need more accurate positioning information than GPS could provide, we have to find a substituted method to approach the same result, use distinguished objects as the base (or original) position point. (Refer to section 3-4.2)

3.4.2. Landmark Selections

Based on traveller's position, select an object within a certain buffer zone (for example, buffer distance is 50 meters) as the base point. The object should be easily detected, such like a super market, church, railway station etc. Referring to the base point, the Web Service gives the navigation instruction.

Chapter 4.

GIS Web Services implementation and results

In this chapter GIS Web Services implementation technique is posed, with taking account into source data and GIS Web Services for solving foot-travelers' requirements. First give an introduction of selected software Mircosoft .NET and component ESRI's ArcObjects, and their relationship. Then four typical GIS Web Services will be developed. Also, for *Interesting Place Search Service,* one of four GIS Web Service implemented, a step-by-step explanation will present.

4.1. How to implement the GIS Web Services

Before start to implement GIS Web Services, the key point is to have proper components to build such architecture. As I mentioned previously, the source data is Geo-database format, an un-open format that developed by ESRI. Because I am familiar with both software VB.NET (part of Microsoft .NET) and ESRI's ArcObject, two products Microsoft's .NET and ESRI's ArcObjects have been selected to develop the Web Services for foot-travelers application. Note:

The two products are Microsoft's .NET initiative and ESRI's ArcObjects. The former will provide the user interface (UI), the network communications, the component framework and the basic glue to hold the entire system together. The latter will provide the three basic tenants of GIS: data storage, map rendering and spatial analysis. When coupled together, we have a very potent combination, one that will allow for massive multi-user systems that deploy as separate units but exists as a single, server based entity.

The concept behind a Web Service is nothing new and is really just a modern incarnation of old RPC (Remote Procedure Call) techniques. Computer A wants Computer B to do some chunk of work on its behalf because it does not have the necessary instructions or does not have access to data needed to complete the work. What is new in the process is an industry wide standard for exposing and consuming these services, a critical piece that was missing in older attempts at integrating computer systems. XML, via the SOAP protocol, has become the universal standard for the transfer of instructions, data and results across machine boundaries. Previous attempts at defining standards for remote calls like

DCOM or CORBA suffered from rigidity, proprietary protocols, and overly complex standards. SOAP replaces these with a simple, open architecture that makes extensive use of existing Internet protocols. It is these universal protocols that help make the XML/SOAP combination so effective, it is built on accepted and proven technologies. For example, Web Services traditionally use HTTP as the transport mechanism, which allows it to move freely (relatively) through corporate firewalls. In previous systems, piercing the firewall represented a major stumbling block for both administrative and security reasons.

Figure 4-1 presents a basic architecture of what components needed to build a GIS Web Service. In technique, Web Service is built in server side and associated with dataset. Any SOAP aware client could invoke remote Web Service by send SOAP message through HTTP protocol, and get SOAP message back in return (no SOAP message return in certain special occasion). ESRI ArcObjects.Net assemblies must be installed on the same server where the Web Service will be hosted. Make sure to make the appropriate references to the ArcObjects.Net assemblies before compiling any code.

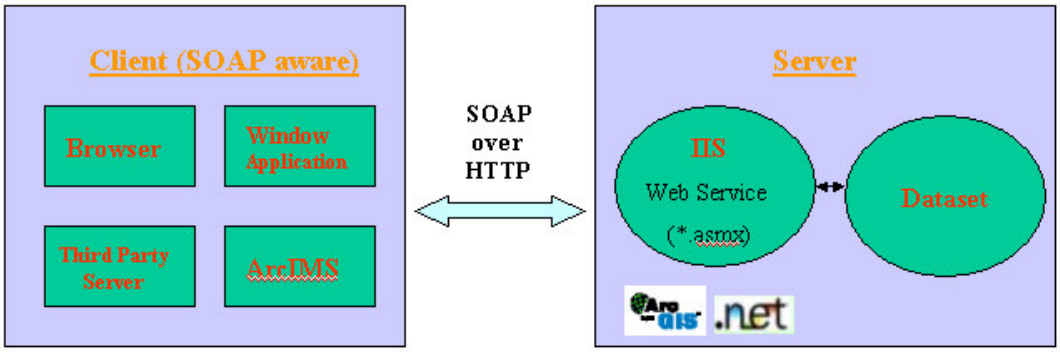


Figure 4-1 Any SOAP aware client can consume GIS functionality that is exposed by GIS Web Services

During the implementation, I have found out that ArcObjects does not support the creation of server-based GIS solutions – these include Internet Map services, Web Services, etc. [36] The "web enabling" ArcObject need a special license from ESRI, which is not available for public yet. Due to the face that I could not get "web enabling" ArcObjects to implement relevant GIS and still need use ArcObject's functionality, I modified this above architecture to fit with the Interactive Development Environment, refer to Figure 4-2. By using the following architecture to implement GIS Web Services, I could make use of ArcObject's functionality indirectly. Meanwhile, the tradeoff is that it doesn't allow multiple users to invoke the GIS Web Services simultaneity, the limitation of local application (*.EXE).

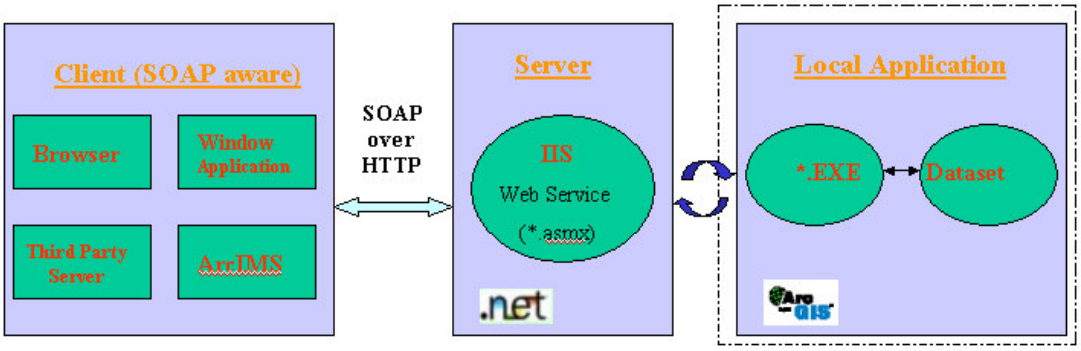


Figure 4-2 Modified Architecture for implementing GIS Web Services

.NET makes creating Web Services almost effortless. Using the Visual Studio.NET (VS.NET) Interactive Development Environment, you can create a simple Web Service in minutes, relying on the ASP.NET framework to publish and host the service. As an example, a simple GIS Web Service can be created that exposes basic ArcGIS spatial operations like Buffer or Union. We want to expose this functionality to other computers that do not have any GIS software installed on them but have a need for performing spatial operations. .NET allows us to wrap the GIS functionality and easily expose it to any client that can connect to the server using the HTTP/XML/SOAP combination.

To facilitate communication with another system that does not have ArcGIS installed, geometry data must be serialized to a format that both systems can understand. The proprietary ArcGIS format is obviously prohibitive so the next logical choice is ArcXML (or GML).

The calling application, can be javascript in a browser, a window Application or another ArcIMS server, sends valid ArcXML geometry representations to the server by calling the appropriate method. If the calling application is built using .NET, you just need to reference the Web Service at design time. You can then use the methods as if they were any other local resource, the .NET framework will execute the method using the appropriate protocols on your behalf. You can also use SOAP toolkits provided by a variety of vendors or by using a lower level, send HTTP GET/POST requests directly at the server. ASP.NET, which is hosting your service, will interpret them and automatically hook up the request to the service.

http://localhost/Restaurant/Restaurant.asmx?Current_Location=Enschede&Type_of_Place=(Restaurant)&

Figure 4-3 - Using HTTP GET to invoke a Web Service hosted in ASP.NET

The above example shows a relatively simple service that makes only cursory use of the ArcObjects components. In reality, more complex services will need to interact with a variety of components, some of which contain large object graphs themselves. A good example of this is the Map CoClass, which is primarily accessed through the IMap and IActiveView interfaces. From these interfaces, a

developer has access to many different ancillary objects like the individual layers, the display transformation or the graphics container. It is this incredible array of classes that makes ArcObjects such an attractive framework, however some parts of this design (as it exists today) can be problematic in multi-threaded containers like ASP.NET.

It is likely that in future versions of ArcIMS, ESRI will expose a large variety of GIS Web Services. Instead of exposing low-level geometry services, you will build on the ESRI services to create more specialized ones. For instance if you were a local municipality, instead of creating a generic Buffer service, you might create and expose a 200-foot abutters list service that accepts parcel identifiers or postal addresses. Other software that is not GIS enabled could simply use the service to query for a list of valid tax records. Unlike traditional html techniques, the Web Service provides a formalized interface (via SOAP) for the calling application to use; the call to the service is as natural as a call to a local resource. For the immediate future, using .NET and raw ArcObjects provides a way to quickly expose GIS functionality. Additionally, at this time, it is unclear how ESRI will expose the bulk of ArcObjects in the context of ArcIMS. Some services may need the fine grain control that only raw ArcObjects exposes, in these cases, using ArcObjects may be the only solution to the problem.

4.2. Implement foot-traveler’s GIS Web Services Information Service

Traveling on foot doesn't simply mean a leisurely walk. Those who wish to go on foot can learn much, become fit and sightsee at the same time. They experience their surroundings in a unique way. There are many interesting and beautiful places along or around travelers that welcome travelers, such like museums, restaurant, park and etc.

Many people are familiar with wireless Internet, but many don't realize the value and potential to make information services highly personalized. One of the best ways to personalize information services is to enable them to be location based. An example would be someone using their Wireless Application Protocol (WAP) based mobile devices (such like pocket PC) to search for a restaurant. The LBS application would interact with other location technology components to determine the user's location and provide a list of restaurants within a certain proximity to the mobile user.

The following Web Services are helping foot-travelers to find out what events or interesting places around him, and select a proper route to get there, and give alert message based on user’s location and pre-defined constraints.

4.2.1. Information Service

Let's look at my Interesting Place Web Service to see how SOAP works. Interesting Place Web Service uses a method called Interesting_Place_Service (String Current_Location, String Type_of_Place). To use this service, the client invokes Interesting Place Search Web Service, enters his current location (place name) and type of place he is looking for, and receives an image that marked his current location and the nearest place (type of place he identified). Because the Interesting Place Search is remote and resides across the Internet, something must happen behind the scenes to transfer the Interesting_Place_Service request to the rremote object, and the remote object must somehow return the result.

To invoke the application Information.exe though the server, below is the method used:

```
Shell ("D:\Information.exe", AppWinStyle.Hide)
```

The workflow and result at client side as Figure 4-4 below:

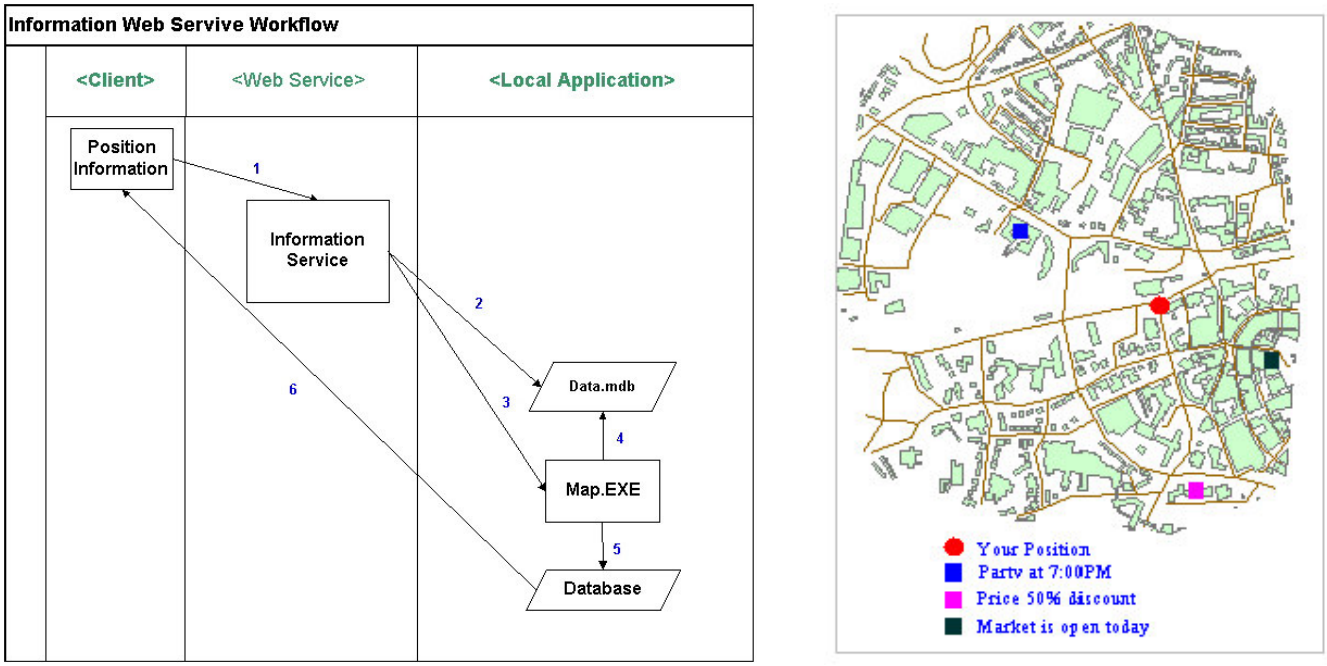


Figure 4-4: Information Service workflow (left) & The effect of Information Service (right)

1. Client invokes the Web Service; user's position information will be input variables.
2. Web Service take the variables sand save it in a Data.mdb file.
3. Invoke the local application Map.exe.
4. Map.exe takes the data in Data.mdb file as input.
5. The local application will be executed, based on user's position, it searches all related information in user's region.
6. Send the all related information of user's current region to Client side.

4.2.2. Interesting Places Service

By using Arcobject and VB, an application (InterestPlace.exe) is made in local. The function of InterestPlace.exe is: searching where is the nearest place based on user selected place type and generate a JPG image with text description. When user invokes Interesting Place Service, the service will save user's position data and selected place type in database (Data.mdb), then the application Information.exe will execute base on user's position data and selected place type, which stored in database previously. To invoke the application Information.exe though the server, below is the method used:

```
Shell ("D:\Map.exe", AppWinStyle.Hide)
```

The workflow and result at client side as Figure 4-5 below:

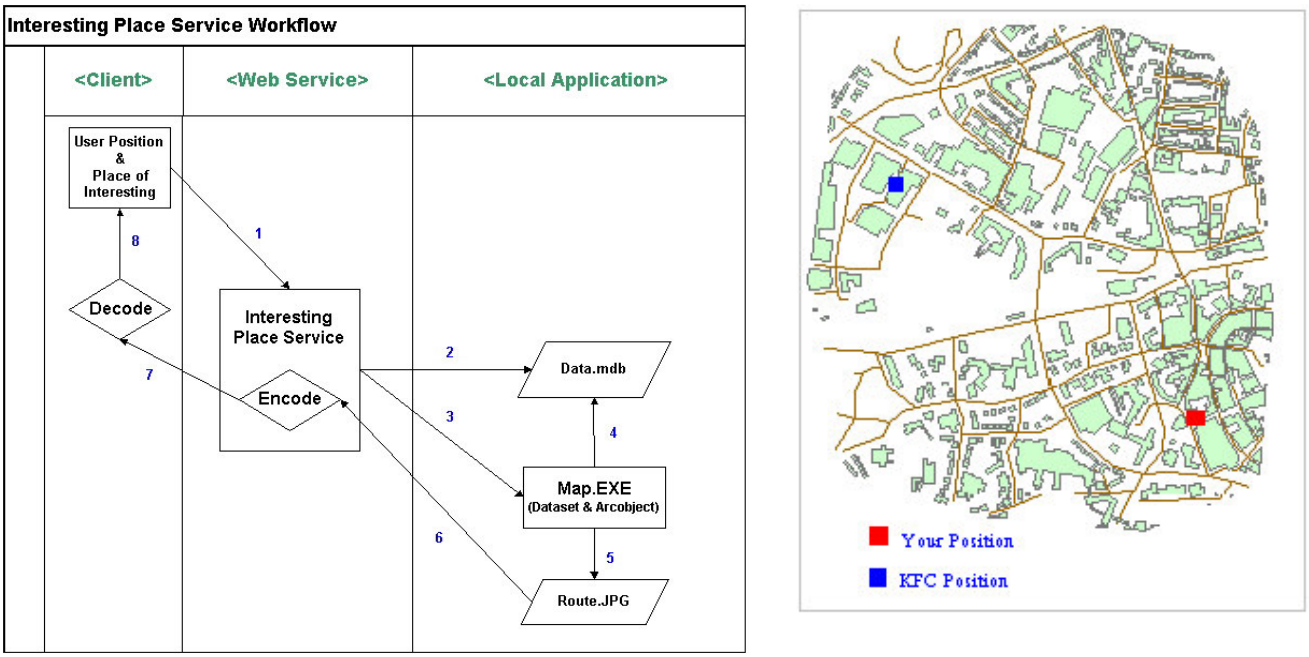


Figure 4-5: Interesting Place Service workflow (left) & The effect of Interesting Place Service (right)

1. Client invokes the Web Service, with input variables.
2. Web Service take the variables sand save it in a Data.mdb file.
3. Invoke the local application Map.exe.
4. It takes the data in Data.mdb file as input.
5. The local application saves the executed result Map.JPG in a certain place.
6. Due to Web Service cannot send JPG file to Client side (SOAP over HTTP) directly, the Web Service save IPG image to a memory stream, and then encode stream to a string with base64 format by Convert class
7. Client side decode base64 string to byte[ ] array, and save it to a JPG format.
8. Display the JPG image on Client side.

4.2.3. Road Navigation Service

By using Arcobject and VB, an application (Road.exe) is made in local. The function of Road.exe is: based on user's current location and user selected target place, to find a shortest path and generate it into a JPG image with text description. When user invokes Road Navigation Service, the service will save user's position data and selected place's position data in database (Data.mdb), then the application Information.exe will execute base on user's position data and selected place's location, which stored in database previously. To invoke the applicationRoad.exe though the server, below is the method used:

```
Shell ("D:\Pathfinder.exe", AppWinStyle.Hide)
```

(Note: This function is build with ESRI's Pathfind.DLL, have special requirement of road network structure. Due to time constraint, I haven't modified all the corresponding data structure yet.)

The workflow and result at client side as Figure 4-6 below:

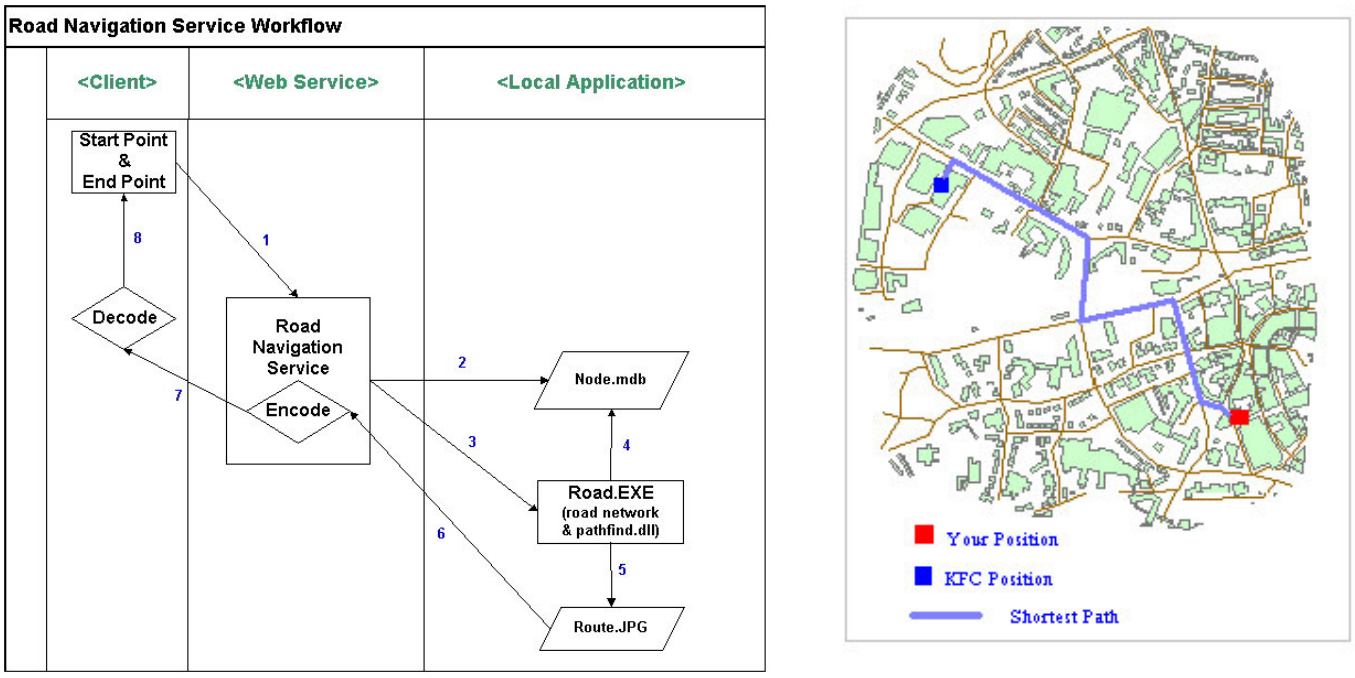


Figure 4-6: Road Navigation Service workflow (left) & The effect of Road Navigation Service (right)

1. Client invokes the Web Service, with input variables.
2. Web Service take the variables sand save it in a Node.mdb file.
3. Invoke the local application Road.exe.
4. It takes the data in Node.mdb file as input.
5. The local application saves the executed result Map.JPG in a certain place.
6. Due to Web Service cannot send JPG file to Client side (SOAP over HTTP) directly, the Web Service save JPG image to a memory stream, and then encode stream to a string with base64 format by Convert class

7. Client side decode base64 string to byte[ ] array, and save it to a JPG format.
8. Display the JPG image on Client side.

4.2.4. Monitor Service

By using Arcobject and VB, an application (Monitor.exe) is made in local. The function of Monitor.exe is: update the database and check if new data fit with user's defined constraints.

When user invokes Road Navigation Service, the service will save user's position data and user defined constraints in database (Data.mdb). Here have two different conditions that will result in sending warning message to user. One condition is the data in current dataset fit with user's defined constraints; the Monitor Service will send a warning message to user; the second condition is that updated new data fit with fit with user's defined constraints.

```
Shell ("D:\Monitor.exe", AppWinStyle.Hide)
```

The workflow and result at client side as Figure 4-7 below:

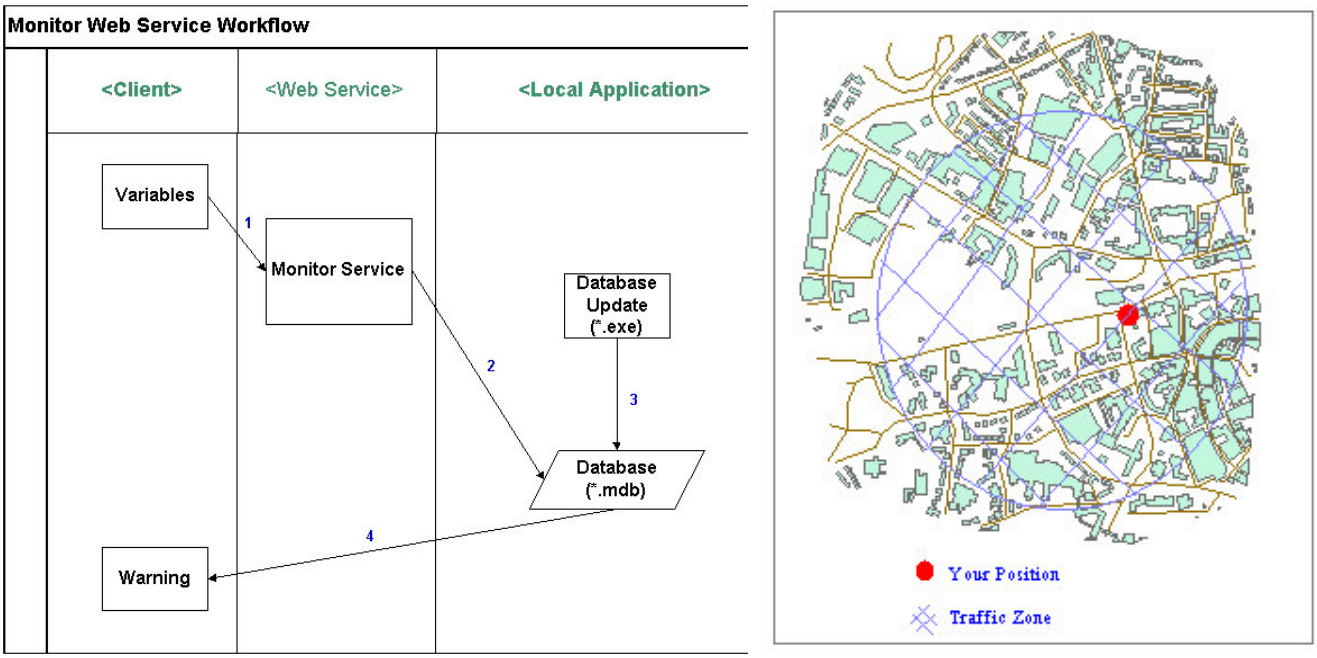


Figure 4-7: Monitor Service workflow (left) & The effect of Monitor Service (right)

1. Client invokes the Web Service, with input variables.
2. Web Service take the variables sand save it in a database (*.mdb) file.
3. Whenever update a database through local application, it will search if the updated information is related with client side constraints or not.
4. If the updated information is fit with client-defined conditions, it will send the warning message to client.

4.3. Example: explain implemented GIS Web Services Method and testing procedures

The research first identifies the proper GIS Web Services building components and architecture. Based on selected components and architecture, four typical GIS Web Services were implemented, the relevant Web Services' design method as descript in figure 4-4, 4-5, 4-6, 4-7. Below will explain what are the implemented GIS Web Services' interfaces and consuming methods.

The research build GIS Web Services in local server, which makes them easy manage and testing. With support of Microsoft .NET Framework, implemented Web Services can be visualized through Internet Explorer, as the figure 4-8 below. Web Services can be invoked by input its web address, in this case, it is http://localhost/Restaurant/GIS_WS.asmx. I will select Interesting_Place_Service to explain how its works.

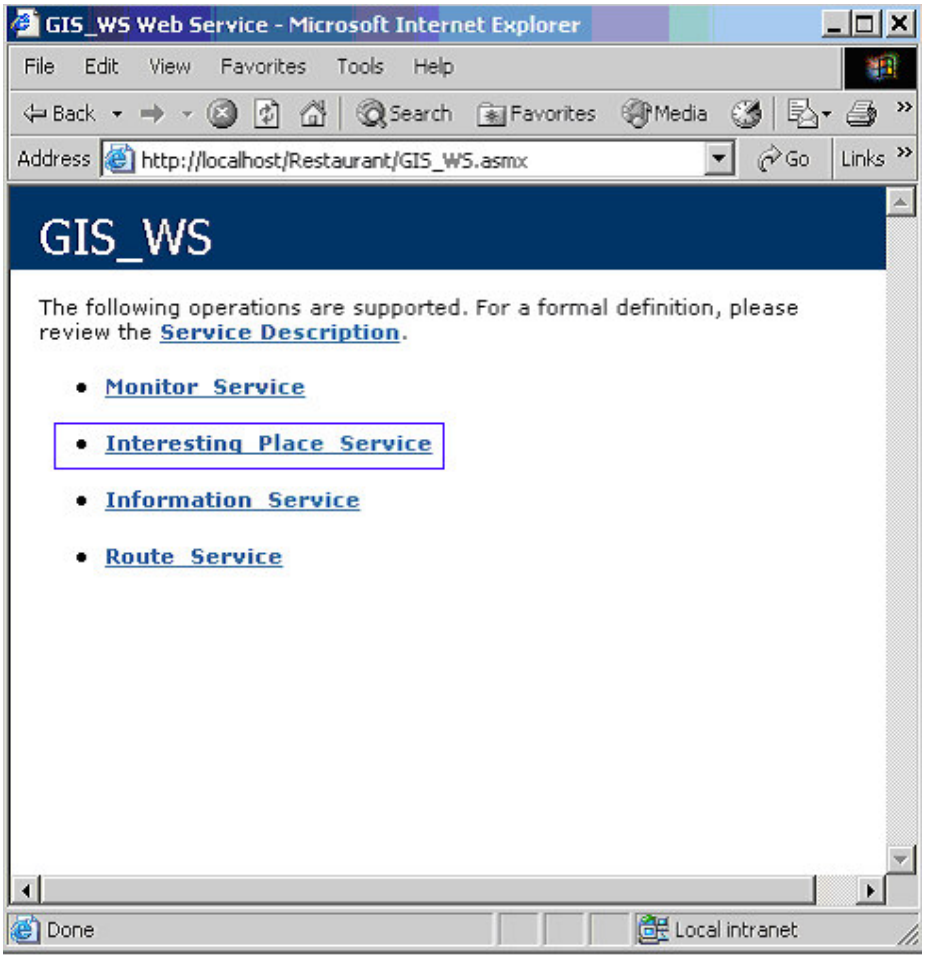


Figure 4-8: Invoke implemented GIS Web Service
(Refer to appendix A)

When invoke the Interesting_Place_Service, the will display the arguments of Interesting_Place_Service method, in this case, they are Current_Location and Type_of_Place. Current will simulate user's position, in real situation, it will be replaced by GPS reader; Type_of_Place will give user a choice for kind of place he/she is looking for.

<WebMethod()> Public Function Interesting_Place_Service(ByVal Current_Location As String, ByVal Type_of_Place As String) As String

The interface as figure 4-9 below:

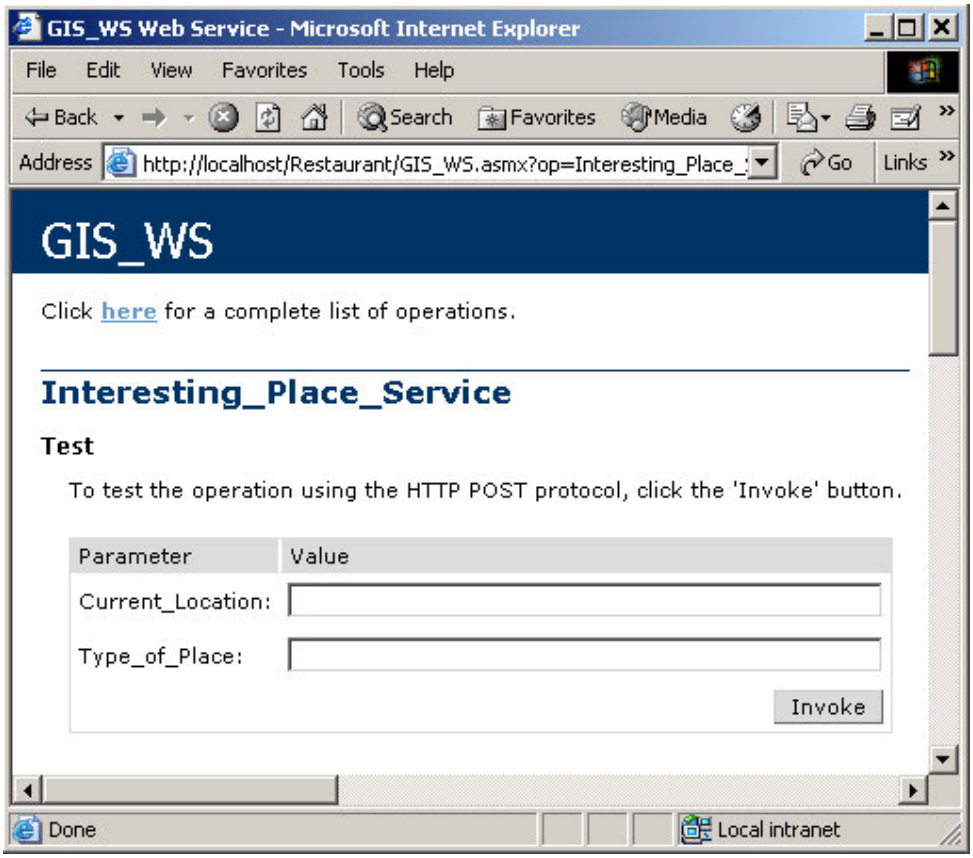


Figure 4-9: Interest_Place_Service Interface

3. When Invoke the Interesting_Place_Service, it will record the argument in a database, and leave the computing work to a local application. In this procedure, Microsoft AxtiveX Database Object was used, as the list below. It will save the argument Current_Location in the table Current_Location of Database WS_Data, as shown in figure 4-10.

```
'----------Current location--------

    Dim con1 As New ADODB.Connection()
    con1 = New ADODB.Connection()
    Dim str1 As String
```

```
    str1 = "Provider = Microsoft.Jet.OLEDB.4.0;" & "Data Source= D:\WS_Data.mdb;"
    con1.ConnectionString = str1
    con1.Open()

    Dim rcs1 As New ADODB.Recordset()
    rcs1 = New ADODB.Recordset()
    Dim strsql1 As String
    strsql1 = "UserPosition"
    rcs1.CursorLocation = ADODB.CursorLocationEnum.adUseClient
    rcs1.Open(strsql1, con1, ADODB.CursorTypeEnum.adOpenKeyset, AD-
ODB.LockTypeEnum.adLockOptimistic, ADODB.CommandTypeEnum.adCmdTable)
    If rcs1.RecordCount = 1 Then
        rcs1.Delete(ADODB.AffectEnum.adAffectCurrent)
    End If
    rcs1.AddNew()
    rcs1.Fields(0).Value = Current_Location
    rcs1.Update()
    rcs1 = Nothing

'----------end current location--------
```

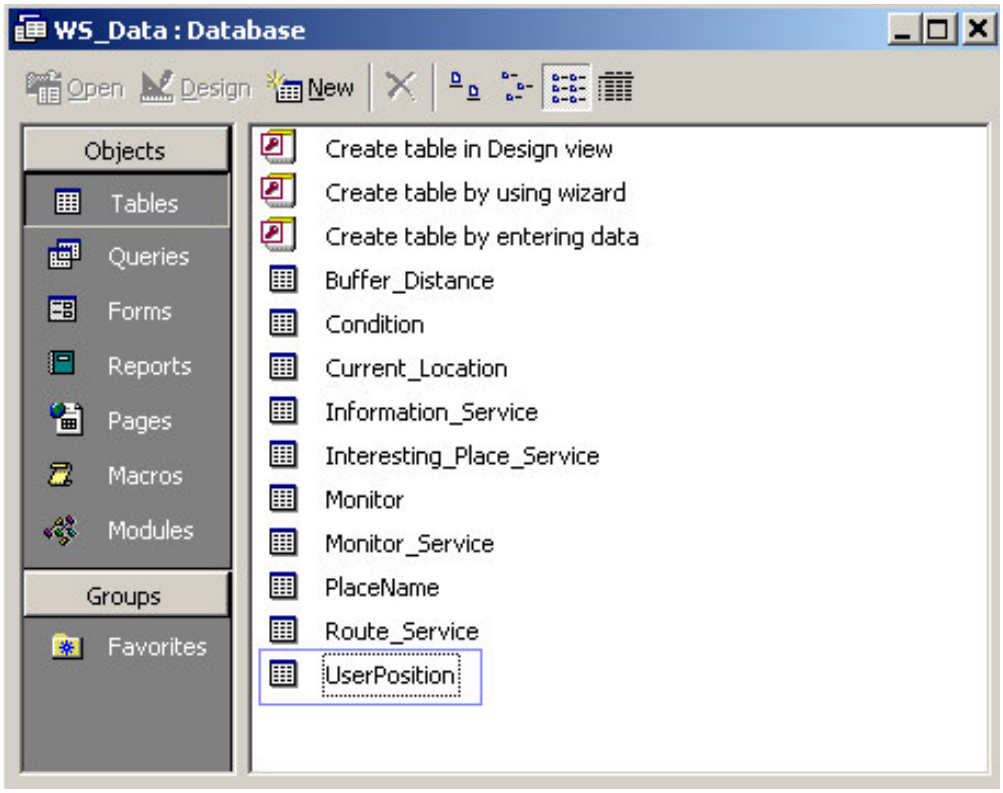


Figure 4-10: Database structure
(Refer to appendix D)

Next it will invoke the local application InterestingPlace.exe, with method:

```
Shell("D:/InterestingPlace.exe", AppWinStyle.Hide)
```

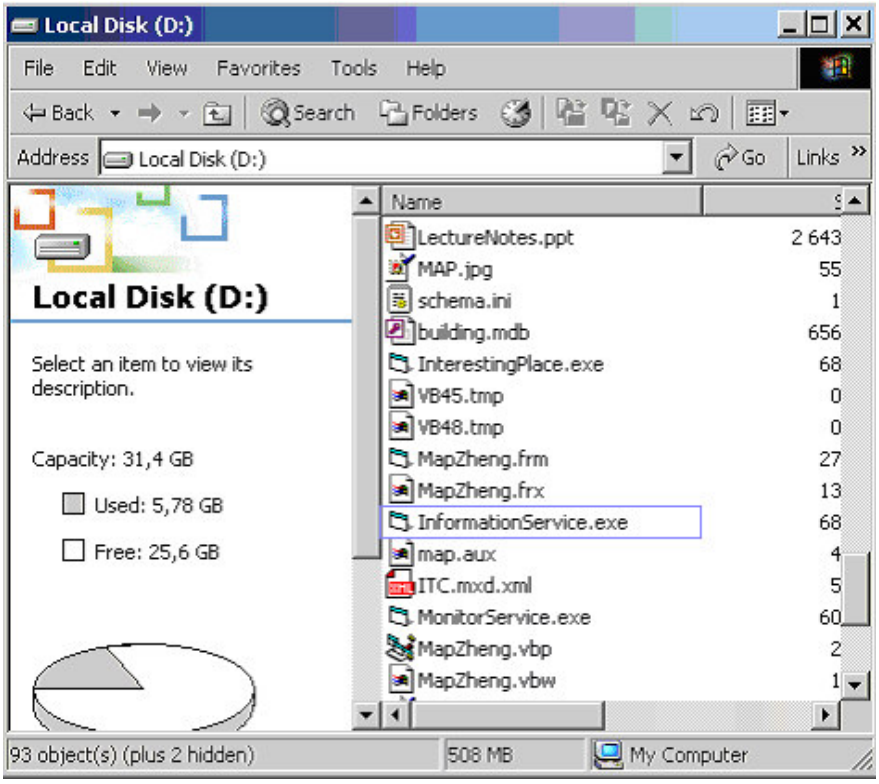


Figure 4-11: Local application associated with Intesesting_Place_Service
(Refer to appendix B)

This local application was build with ESRI's ArcObject. When it been invoked, it will load arguments from the database to receive User_Location and Type_of _Place values, as in figure 4-10. The local application executes the arguments, and then save the result InterestingPlace.jpg. The result looks like this:

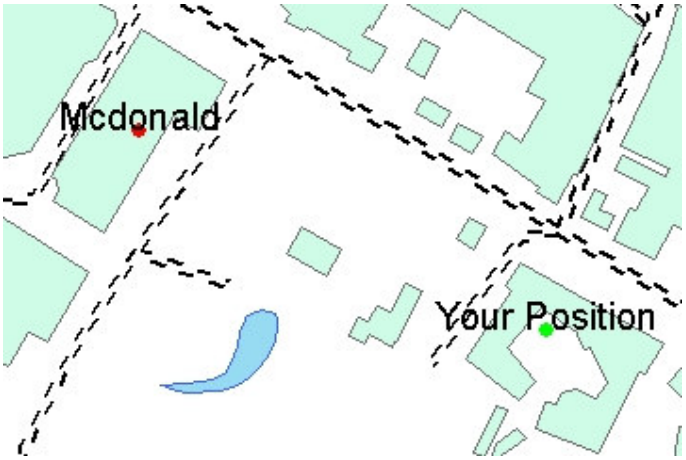


Figure 4-12: Executed result of Intesesting_Place_Service

Interesting_Place_Service's text output is SOAP message, an XML based language. As shown in figure 4-13 below:

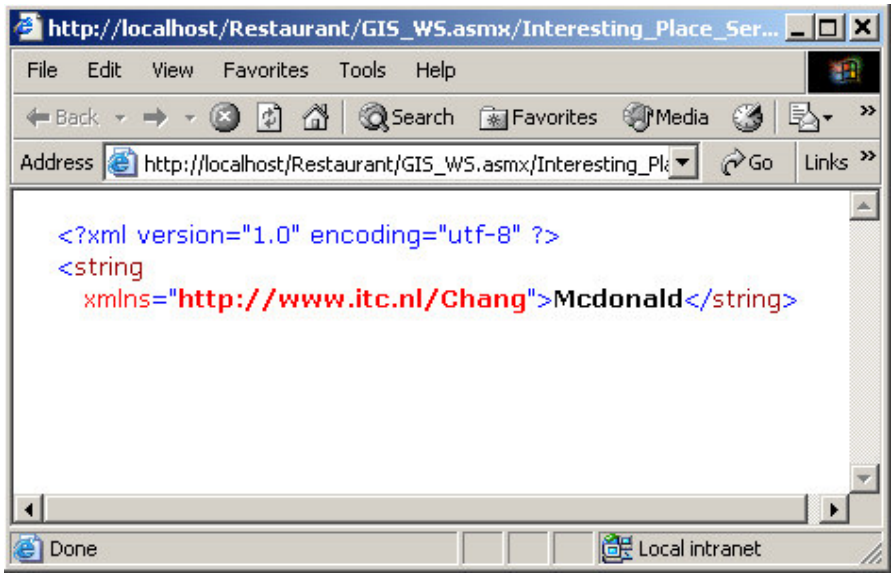


Figure 4-13: SOAP Response

Because Web Service cannot send JPG file to Client side (SOAP over HTTP) directly, so the image .jpg image have to send to client side through another Web Service: GetImage. Image Web Service save jpg image to a memory stream, and then encode stream to a string with base64 format by Convert class:

```
[WebMethod(Description="Get an Image using Base64 Encoding")]
        public byte[] GetImage(string Image)
        {
                Image= "D:\\" + Image + ".jpg";
                        return getBinaryFile(Image);
        }
```

It converts .jpg image to Base64 Encoding, text format. As figure 4-14 and figure 4-15:

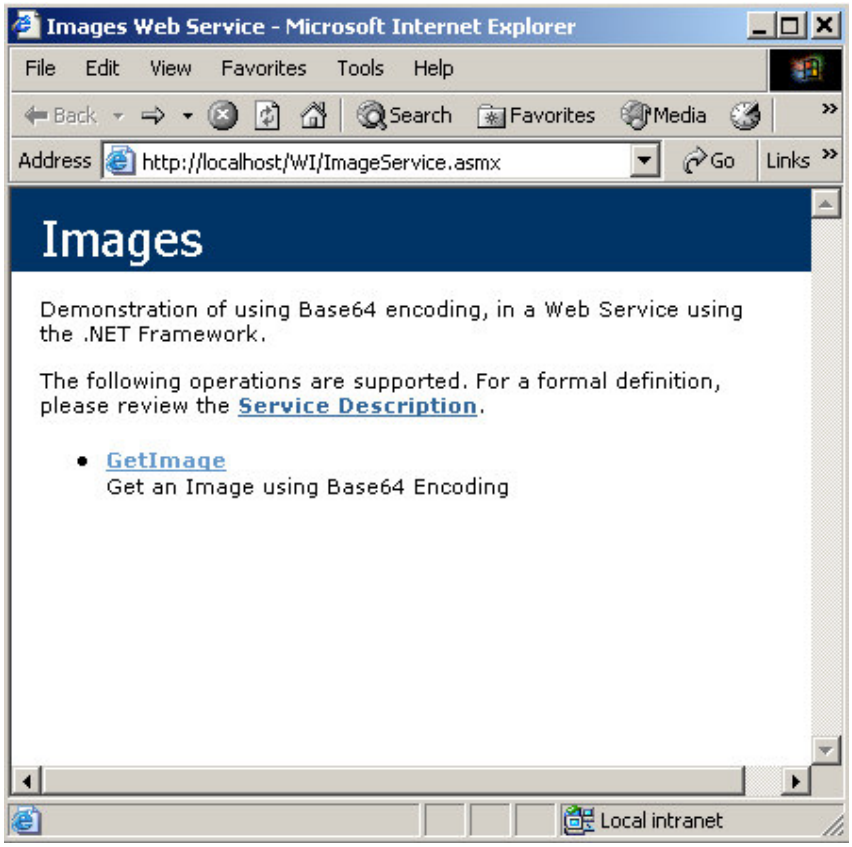


Figure 4-14: Image Web Service

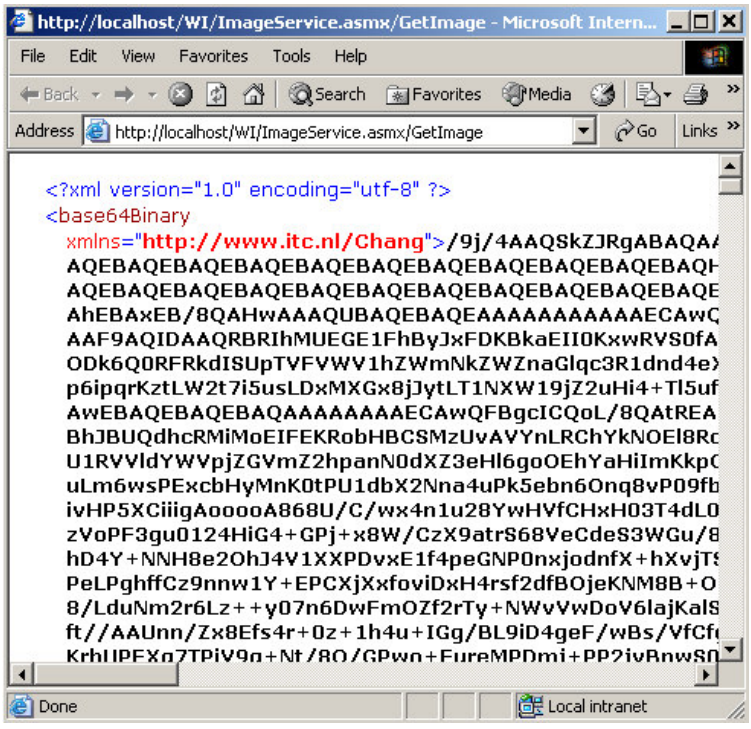


Figure 4-15: Result of image Base64 encoding

Finally, client side application developer can make use of the GIS Web Services, and develop an application for foot-travelers. As show in figure 4-16, only with argument "Restaurant" (User's position is received by GPS), the Interesting_Place_Service will do all the computing work in web server side and send the SOAP message (Text format) back. Client side application use Base64 decoding the SOAP message into Image. The method is the class as below, and the result as in figure 4-16:

```
Dim yy As New localhost.Images()
        Dim data() As Byte = yy.GetImage
        Dim sss As System.IO.MemoryStream = New Sys-
tem.IO.MemoryStream(data)
        'Dim ssss As New System.IO.FileStream()

        Dim bm As Bitmap
        bm = New Bitmap(sss)
        PictureBox1.Image = bm
```

Figure 4-16: Client side application which make use of Interesting_Place_Service
(Refer to appendix C)

Chapter 5.

Conclusions and recommendations

5.1. Conclusions

A Web Service is a software component that can be accessed over the Web for use in other applications. GIS Web Services, such as Web Service examples I developed during the research, provide hosted spatial data and GIS functionality via the Internet to packaged and customized Web applications. In essence, GIS Web Services let developers include GIS content and capabilities in their applications without having to host the data or develop the necessary tools themselves. The result is significant savings of development time, expense, and computer resources.

Web Services are significant because they're implemented as a new layer that integrates existing systems as opposed to a whole architecture. And while the basic computer components of a Web Services system are still clients and servers, it's important to recognize that the network can be any open or secure network. Web Services remove the requirement for permanent network connections using a stateless environment in which nodes interconnect only when necessary — a very promising architecture for practical implementation of distributed geographical information systems. As the GIS Web Services architecture evolves, its no doubt that we will witness wide spread support for more integrated applications such as location-based services.

In this age of significant telecommunications competition, mobile network operators continuously seek new and innovative ways to create differentiation and increase profits. One of the best ways to do accomplish this is through the delivery of highly personalized services. One of the most powerful ways to personalize mobile services is based on location, this kind personalize services are extremely useful for travelers in mobile status. During my study, I have classified Location Based Services (LBS) into for major categories, they are: Interesting Place Service, Information Service, Route Service and Monitor Service. As it is mentioned in previous chapters, there are many aspects connected with Web Service and Mobile GIS application, but the research objective here was concentrate on the design of a prototype of GIS Web Service architecture for Mobile GIS application, and then implement several typical GIS Web Services for foot-travelers' location based application. For this reason this research was focused on finding suitable building blocks of GIS Web Service architecture and

simulating the whole application in local server, and didn't consider the affects of web secure issue and run time error of local application.

GIS Web Services use data and related functionality to perform basic geo-processing tasks such as address matching, interesting place search, map image display, and routing. Application developers can use GIS Web Services to perform real-time processing on the computers hosting the Web Services and return the results to their local applications — all over the Internet. You don't have to maintain GIS application tools or the associated geographical data on your local system to use them in your custom application.

As you might expect, GIS Web Services depend heavily on Simple Object Access Protocol (SOAP) infrastructure. SOAP standardizes the way a Web Service communicates with a client and allows programs written in different languages and on different platforms to communicate. SOAP works with standard Web protocols including XML, HTTP, and TCP/IP, as well as emerging Web Services protocols such as Web GIS Web Services are published on the UDDI registry, a universal database of Web Services. Developers can search any UDDI site to discover services on the registry, making UDDI a powerful resource for publishers and consumers alike. Discovery happens either through a vendor-managed Web interface (https://uddi.microsoft.com/register.aspx, for example) or SOAP calls.

After study Web Service framework, Web Service applications and foot-traveler' requirements, a final prototype of GIS Web Service architecture was developed in chapter 4, with corresponded client application for consuming and testing implemented GIS Web Services. At end of the research, it is important to declare the contributions of this study, which consider might helpful for solving problems in the fields of distributed geo-information processing and foot-travelers application for Location Based Service.

One contribution of the research is: made general classification for foot-travelers' required services, spatial / non-spatial information is criteria in this services classification. They are Interesting Place Service, Information Service, Route Service and Monitor Service. Based on this general classification, new service can be sort into one service catalog. Because all the services in one catalog share same or similar characters, the classification method will make new service's implementation easier.

Another contribution of the research is: to show it is possible to divide GIS Web Service into separated components, each component capable for a certain function. As reader might noticed, each Web Service is linked with a local application that capable for solving a generic geo-problem, is related with a foot-traveler required service logically. The local applications are independent, with proper parameters, these local applications can be invoked and generate desired result.

- Web Services are a new, standard platform for building interoperable distributed applications. The strong point behind Web Services is cross-platform interoperability. To achieve this, the Web Services platform relies heavily on vendor and platform-neutral standards such as XML and XSD. The key advantage to using Web Services for GIS application is low-cost interoperability. By exposing your GIS functions as Web Services, you are enabling any authorized party to easily invoke those services regardless of the platform and programming language they use.

- Microsoft's .NET and ESRI's ArcObjects. The former will provide the user interface (UI), the network communications, the component framework and the basic glue to hold the entire system together. The latter will provide the three basic tenants of GIS: data storage, map rendering and spatial analysis. When coupled then together, it has a very potent combination, one that will allow for massive multi-user systems that deploy as separate units but exists as a single, server based entity. A system that can be fully administered (installations and updates) from a central location yet still provides a comprehensive UI that end users have come to expect from traditional desktop applications. The application develop environment use both .NET and ArcObjects because it is believed that they represent the best tools for build Web Services and GIS function. However, it should be noted that other frameworks might be used in their place.

- It is clear that Web Services offer the most benefit in cases where interoperability and/or remoting over the Web are desired. There are many scenarios where you don't benefit from building or using Web Services. For example, homogenous applications running on the same machine or on different machines on the same LAN should not use Web Services to communicate with one another.

5.2. Recommendations

In accordance to the research objectives, the essential characteristics of GIS Web Service components were first identified and built the architecture, and then implemented several typical GIS Web Service for foot-traveler application. Due to time limitation, there are other issues were considered but couldn't be implemented. But as a good start point, following recommendations can be useful for late GIS Web Service developers.

- The research develop several GIS Web Services for simulation and testing, it didn't worry about real-world scenarios too much, such like multiple users access web server and consume GIS Web Service. But for real-world scenarios, a real GIS Web Service may deal with large amount data, the data storage method and optimum algorithm are very important. It will speed up Web Service's response time. Also, when you build a GIS Web Service to address a specific problem of foot-travelers, human factors and environmental conditions should to be considered fully, which is essential for a successful application.

- For commercialized GIS Web Services, Web Services developer need to know that Web Services should be discoverable. So enable others can make use of the implemented Web Service, you need publish the Web Service at a Web Service register center. In this way, anyone can access Internet and any Web Service register center will able find your published Web Service.

- Web Service is self-describing, its very important to make other Web Services consumer clear what's the implemented Web Service's functionality, parameters and etc. So when publish the implemented Web Service, you should give the implemented Web Service a comprehensive description.

- To make use of Esri's ArcObjects functionality, this research implements a few GIS Web Services which associated with local applications. In real scenario, allow end users to invoke server side local application may cause security problem.

Reference:

[1] Monternet Company
http://www.monternet.com. Accessed on 15-10-2002.

[2] Place Search
http://www.monternet.com/image/gprs/tour.htm. Accessed on 15-10-2002.

[3] Web Services
http://www.w3.org/2002/ws/. Accessed on 15-10-2002.

[4] Google Web APIs – Home
http://www.google.com/apis/index.html. Accessed on 15-10-2002.

[5] IP2001 OGC Web Services Initiative Home Page
http://ip.opengis.org/ows/index.html. Accessed on 15-10-2002.

[6] ArcWeb USA tutorials
http://www.geographynetwork.com/services/webhelp/webservices.htm. Accessed on 15-10-2002.

[7] The World Wide Web Consortium
http://www.w3.org/. Accessed on 15-10-2002.

[8] W3Schools Online Web Tutorials
http://www.w3schools.com/. Accessed on 15-10-2002.

[9] developerWorks Web Services
http://www-106.ibm.com/developerworks/webservices/. Accessed on 15-10-2002.

[10] Web Services architecture overview
http://www-106.ibm.com/developerworks/webservices/library/w-ovr/. Accessed on 15-2-2003.

[11] Place finder Web Service
http://www.geographynetwork.com/services/v1/PlaceFinder. Accessed on 15-2-2003.

[12] Impact of information technology on personal travel and commercial vehicle operations: research challenges and opportunities
Thomas F. Golob, Amelia C. Regan, Transportation research part C 9 (2001) 87-121

[13] Write once, serve everyone
The future of E-Commerce is Web Service, by Martin Streicher, August 2002 Linux Magazine

[14] Web Services architecture overview
http://www-106.ibm.com/developerworks/webservices/library/w-ovr/. Accessed on 15-2-2003.

[15] The World Wide Web Consortium
http://www.w3.org. Accessed on 15-2-2003.

[16] XML Protocol Comparisons
http://www.w3.org/2000/03/29-XML-protocol-matrix. Accessed on 15-2-2003.

[16.1] SOAP Tutorial
http://www.w3schools.com/soap/default.asp. Accessed on 15-2-2003.

[16.2] ArcWeb USA tutorials
http://www.geographynetwork.com/services/webhelp/webservices.htm

[17] OneLook® Dictionary Search
http://www.onelook.com/. Accessed on 15-2-2003.

[18] Web Services architecture overview
http://www-106.ibm.com/developerworks/webservices/library/w-ovr/. Accessed on 15-2-2003.

[19] Web Services architecture : they stack up
http://www.webservicesarchitect.com/content/articles/webservicesarchitectures.pdf. Accessed on 15-2-2003.

[20] Simple Object Access Protocol (SOAP) 1.1
http://www.w3.org/TR/SOAP/. Accessed on 15-2-2003.

[21] Web Service Definition Language (WSDL)
http://www.w3.org/TR/wsdl. Accessed on 15-2-2003.

[22] UDDI.org
http://www.uddi.org/. Accessed on 15-2-2003.

[23] IBM Software Solutions Web Services by IBM Documentation
http://www.ibm.com/software/solutions/webservices/documentation.html. Accessed on 15-2-2003.

[24] Open GIS Consortium, Inc. (OGC)

http://www.opengis.org/pressrm/summaries/20011127.TS.OWS.htm. Accessed on 15-2-2003.

[25] ArcWeb Services
http://www.esri.com/software/arcwebservices/index.html. Accessed on 15-2-2003.

[26] Geography Network - Web Services
http://www.geographynetwork.com/webservices/index.html. Accessed on 15-2-2003.

[27] ArcWeb USA tutorials
http://www.geographynetwork.com/services/webhelp/webservices.htm. Accessed on 15-2-2003.

[28] MapPoint .NET Demos
http://demo.mappoint.net/. Accessed on 15-2-2003.

[29] ArcWeb USA tutorials
http://www.geographynetwork.com/services/webhelp/webservices.htm. Accessed on 15-2-2003.

[30] IBM WebSphere - middleware, application server, e-business, infrastructure software
http://www-3.ibm.com/software/info1/websphere/index.jsp?tab=highlights. Accessed on 15-2-2003.

[31] Visual Studio .NET
http://msdn.microsoft.com/vstudio/productinfo/default.asp. Accessed on 15-2-2003.

[32] GLUE, the Mind Electric™
http://www.themindelectric.com/glue/. Accessed on 15-2-2003.

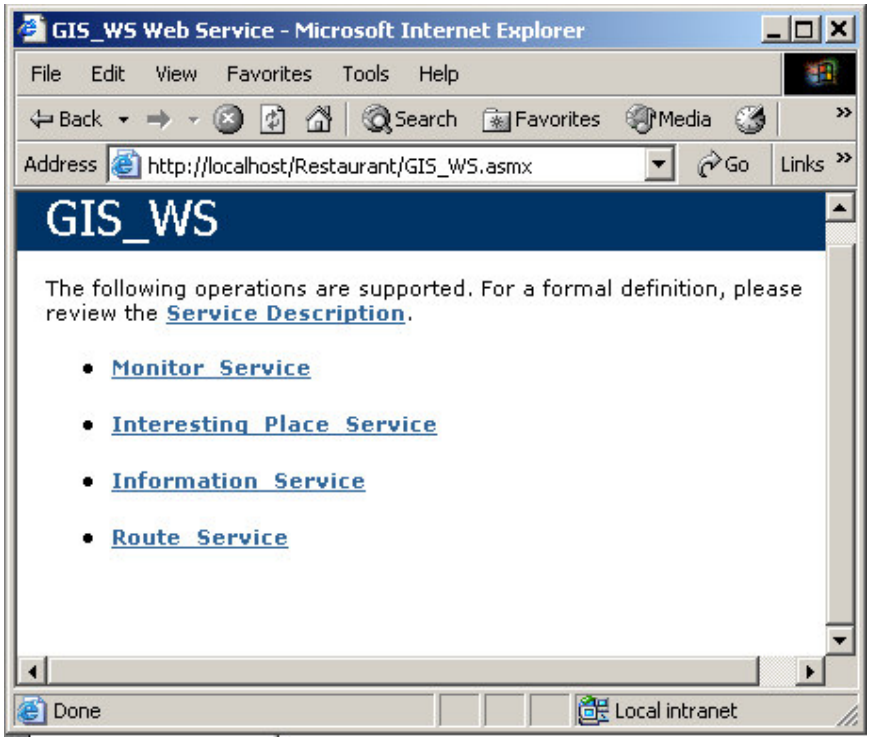
[33] How the IBM Web Services Development Environment and Toolkit Compares with Microsoft Visual Studio .NET
http://www7.software.ibm.com/vad.nsf/data/document4376?OpenDocument&p=1&BCT=66&Footer=1. Accessed on 15-2-2003.

[34] How IBM WebSphere Studio Application Developer Compares with Microsoft Visual Studio .NET -- Part 2 Implementation Differences
http://www7b.software.ibm.com/wsdd/techjournal/0204_kraft/kraft.html. Accessed on 15-2-2003.

[35] Google Web APIs - Home
http://www.google.com/apis/index.html. Accessed on 15-2-2003.

[36] ArcObjects Online
http://arconline.esri.com/arcobjectsonline/. Accessed on 15-2-2003.

Appendix A: GIS Web Services




```
Imports System.Web.Services

<WebService(Namespace:="http://www.itc.nl/Chang")> _
Public Class GIS_WS

    Inherits System.Web.Services.WebService

#Region " Web Services Designer Generated Code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Web Services Designer.
        InitializeComponent()

        'Add your own initialization code after the InitializeComponent() call

    End Sub

    'Required by the Web Services Designer
    Private components As System.ComponentModel.IContainer
```

```
    'NOTE: The following procedure is required by the Web Services Designer
    'It can be modified using the Web Services Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()

    End Sub

    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        'CODEGEN: This procedure is required by the Web Services Designer
        'Do not modify it using the code editor.
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

#End Region

    ' WEB SERVICE EXAMPLE
    ' The HelloWorld() example service returns the string Hello World.
    ' To build, uncomment the following lines then save and build the project.
    ' To test this Web Service, ensure that the .asmx file is the start page
    ' and press F5.
    '
    <WebMethod()> Public Function Interesting_Place_Service(ByVal Current_Location As String, ByVal
Type_of_Place As String) As String

        '-----------current location--------
        Dim userX, userY As Double

        Dim con1 As New ADODB.Connection()
        con1 = New ADODB.Connection()
        Dim str1 As String
        str1 = "Provider = Microsoft.Jet.OLEDB.4.0;" & "Data Source= D:\WS_Data.mdb;"
        con1.ConnectionString = str1
        con1.Open()

        Dim rcs1 As New ADODB.Recordset()
        rcs1 = New ADODB.Recordset()
        Dim strsql1 As String
        strsql1 = "UserPosition"
        rcs1.CursorLocation = ADODB.CursorLocationEnum.adUseClient
        rcs1.Open(strsql1, con1, ADODB.CursorTypeEnum.adOpenKeyset, AD-
ODB.LockTypeEnum.adLockOptimistic, ADODB.CommandTypeEnum.adCmdTable)
        If rcs1.RecordCount = 1 Then
            rcs1.Delete(ADODB.AffectEnum.adAffectCurrent)
        End If
        rcs1.AddNew()
        rcs1.Fields(0).Value = Current_Location
        rcs1.Update()
        rcs1 = Nothing

        '-----------end current location--------

        '-----------add or update type_of_place--------
        Dim con As New ADODB.Connection()
        con = New ADODB.Connection()
```

```
        Dim str As String
        str = "Provider = Microsoft.Jet.OLEDB.4.0;" & "Data Source= D:\WS_Data.mdb;"
        con.ConnectionString = str
        con.Open()
        Dim rcs As New ADODB.Recordset()
        rcs = New ADODB.Recordset()
        Dim strsql As String
        strsql = "Interesting_Place_Service"
        rcs.CursorLocation = ADODB.CursorLocationEnum.adUseClient
        rcs.Open(strsql, con, ADODB.CursorTypeEnum.adOpenKeyset, AD-
ODB.LockTypeEnum.adLockOptimistic, ADODB.CommandTypeEnum.adCmdTable)

        If rcs.RecordCount = 1 Then
            rcs.Delete(ADODB.AffectEnum.adAffectCurrent)
        End If
        rcs.AddNew()
        rcs.Fields(0).Value = Type_of_Place
        rcs.Update()
        rcs = Nothing
        '-----------End add or update type_of_place--------

        Shell("D:/InterestingPlace.exe", AppWinStyle.Hide)


        Dim Title As String

        Dim con3 As New ADODB.Connection()
        con1 = New ADODB.Connection()
        Dim str3 As String
        str3 = "Provider = Microsoft.Jet.OLEDB.4.0;" & "Data Source= D:\WS_Data.mdb;"
        con3.ConnectionString = str3
        con3.Open()
        Dim rcs3 As New ADODB.Recordset()
        rcs3 = New ADODB.Recordset()
        rcs3 = New ADODB.Recordset()
        Dim strsql3 As String
        strsql3 = "PlaceName"
        rcs3.CursorLocation = ADODB.CursorLocationEnum.adUseClient
        rcs3.Open(strsql3, con, ADODB.CursorTypeEnum.adOpenKeyset, AD-
ODB.LockTypeEnum.adLockOptimistic, ADODB.CommandTypeEnum.adCmdTable)
        Title = rcs3.Fields(0).Value
        Interesting_Place_Service = Title

    End Function


    <WebMethod()> Public Function Information_Service(ByVal Current_Location As String, ByVal Buffer As
Double) As String

        '-----------current location--------
        Dim userX, userY As Double
        Dim con1 As New ADODB.Connection()
        con1 = New ADODB.Connection()
        Dim str1 As String
        str1 = "Provider = Microsoft.Jet.OLEDB.4.0;" & "Data Source= D:\WS_Data.mdb;"
        con1.ConnectionString = str1
        con1.Open()
```

```
        Dim rcs1 As New ADODB.Recordset()
        rcs1 = New ADODB.Recordset()
        Dim strsql1 As String
        strsql1 = "UserPosition"
        rcs1.CursorLocation = ADODB.CursorLocationEnum.adUseClient
        rcs1.Open(strsql1, con1, ADODB.CursorTypeEnum.adOpenKeyset, AD-
ODB.LockTypeEnum.adLockOptimistic, ADODB.CommandTypeEnum.adCmdTable)
        If rcs1.RecordCount = 1 Then
            rcs1.Delete(ADODB.AffectEnum.adAffectCurrent)
        End If
        rcs1.AddNew()
        rcs1.Fields(0).Value = Current_Location
        rcs1.Update()
        rcs1 = Nothing

        '-----------end current location--------

        '-----------add or update buffer distance--------
        Dim con As New ADODB.Connection()
        con = New ADODB.Connection()
        Dim str As String
        str = "Provider = Microsoft.Jet.OLEDB.4.0;" & "Data Source= D:\WS_Data.mdb;"
        con.ConnectionString = str
        con.Open()
        Dim rcs As New ADODB.Recordset()
        rcs = New ADODB.Recordset()
        Dim strsql As String
        strsql = "Buffer_Distance"
        rcs.CursorLocation = ADODB.CursorLocationEnum.adUseClient
        rcs.Open(strsql, con, ADODB.CursorTypeEnum.adOpenKeyset, AD-
ODB.LockTypeEnum.adLockOptimistic, ADODB.CommandTypeEnum.adCmdTable)

        If rcs.RecordCount = 1 Then
            rcs.Delete(ADODB.AffectEnum.adAffectCurrent)
        End If
        rcs.AddNew()
        rcs.Fields(0).Value = Buffer
        rcs.Update()
        rcs = Nothing
        '-----------End add or update buffer_distance--------

        Shell("D:/InformationService.exe", AppWinStyle.Hide)


    End Function


    <WebMethod()> Public Function Monitor_Service(ByVal Current_Location As String, ByVal Condition As
String) As String

        '-----------current location--------
        Dim userX, userY As Double
        Dim con1 As New ADODB.Connection()
        con1 = New ADODB.Connection()
        Dim str1 As String
        str1 = "Provider = Microsoft.Jet.OLEDB.4.0;" & "Data Source= D:\WS_Data.mdb;"
        con1.ConnectionString = str1
        con1.Open()
```

```
        Dim rcs1 As New ADODB.Recordset()
        rcs1 = New ADODB.Recordset()
        Dim strsql1 As String
        strsql1 = "UserPosition"
        rcs1.CursorLocation = ADODB.CursorLocationEnum.adUseClient
        rcs1.Open(strsql1, con1, ADODB.CursorTypeEnum.adOpenKeyset, AD-
ODB.LockTypeEnum.adLockOptimistic, ADODB.CommandTypeEnum.adCmdTable)
        If rcs1.RecordCount = 1 Then
            rcs1.Delete(ADODB.AffectEnum.adAffectCurrent)
        End If
        rcs1.AddNew()
        rcs1.Fields(0).Value = Current_Location
        rcs1.Update()
        rcs1 = Nothing

        '-----------end current location--------

        '-----------add or update condition--------
        Dim con As New ADODB.Connection()
        con = New ADODB.Connection()
        Dim str As String
        str = "Provider = Microsoft.Jet.OLEDB.4.0;" & "Data Source= D:\WS_Data.mdb;"
        con.ConnectionString = str
        con.Open()
        Dim rcs As New ADODB.Recordset()
        rcs = New ADODB.Recordset()
        Dim strsql As String
        strsql = "Condition"
        rcs.CursorLocation = ADODB.CursorLocationEnum.adUseClient
        rcs.Open(strsql, con, ADODB.CursorTypeEnum.adOpenKeyset, AD-
ODB.LockTypeEnum.adLockOptimistic, ADODB.CommandTypeEnum.adCmdTable)

        If rcs.RecordCount = 1 Then
            rcs.Delete(ADODB.AffectEnum.adAffectCurrent)
        End If
        rcs.AddNew()
        rcs.Fields(0).Value = Condition
        rcs.Update()
        rcs = Nothing
        '-----------End add or update condition--------

        Shell("D:/MonitorService.exe", AppWinStyle.Hide)

        Dim con2 As New ADODB.Connection()
        con2 = New ADODB.Connection()
        Dim str2 As String
        str2 = "Provider = Microsoft.Jet.OLEDB.4.0;" & "Data Source= D:\WS_Data.mdb;"
        con2.ConnectionString = str2
        con2.Open()
        Dim rcs2 As New ADODB.Recordset()
        rcs2 = New ADODB.Recordset()
        Dim strsql2 As String
        strsql2 = "Monitor"
        rcs2.CursorLocation = ADODB.CursorLocationEnum.adUseClient
        rcs2.Open(strsql, con, ADODB.CursorTypeEnum.adOpenKeyset, AD-
ODB.LockTypeEnum.adLockOptimistic, ADODB.CommandTypeEnum.adCmdTable)

        Dim conD As String
```

```
        conD = rcs2.Fields(0).Value

        Monitor_Service = conD

    End Function
    <WebMethod()> Public Function Route_Service(ByVal Current_Location As String, ByVal Destination As
String) As String

        '-----------current location--------
        Dim userX, userY As Double

        Dim con1 As New ADODB.Connection()
        con1 = New ADODB.Connection()
        Dim str1 As String
        str1 = "Provider = Microsoft.Jet.OLEDB.4.0;" & "Data Source= D:\WS_Data.mdb;"
        con1.ConnectionString = str1
        con1.Open()

        Dim rcs1 As New ADODB.Recordset()
        rcs1 = New ADODB.Recordset()
        Dim strsql1 As String
        strsql1 = "UserPosition"
        rcs1.CursorLocation = ADODB.CursorLocationEnum.adUseClient
        rcs1.Open(strsql1, con1, ADODB.CursorTypeEnum.adOpenKeyset, AD-
ODB.LockTypeEnum.adLockOptimistic, ADODB.CommandTypeEnum.adCmdTable)
        If rcs1.RecordCount = 1 Then
            rcs1.Delete(ADODB.AffectEnum.adAffectCurrent)
        End If
        rcs1.AddNew()
        rcs1.Fields(0).Value = Current_Location
        rcs1.Update()
        rcs1 = Nothing

        '-----------end current location--------

        '-----------add or update type_of_place--------
        Dim con As New ADODB.Connection()
        con = New ADODB.Connection()
        Dim str As String
        str = "Provider = Microsoft.Jet.OLEDB.4.0;" & "Data Source= D:\WS_Data.mdb;"
        con.ConnectionString = str
        con.Open()
        Dim rcs As New ADODB.Recordset()
        rcs = New ADODB.Recordset()
        Dim strsql As String
        strsql = "Interesting_Place_Service"
        rcs.CursorLocation = ADODB.CursorLocationEnum.adUseClient
        rcs.Open(strsql, con, ADODB.CursorTypeEnum.adOpenKeyset, AD-
ODB.LockTypeEnum.adLockOptimistic, ADODB.CommandTypeEnum.adCmdTable)

        If rcs.RecordCount = 1 Then
            rcs.Delete(ADODB.AffectEnum.adAffectCurrent)
        End If
        rcs.AddNew()
        rcs.Fields(0).Value = Destination
        rcs.Update()
        rcs = Nothing
        '-----------End add or update type_of_place--------
```

```
        Shell("D:/InterestingPlace.exe", AppWinStyle.Hide)


        Dim Title As String

        Dim con3 As New ADODB.Connection()
        con1 = New ADODB.Connection()
        Dim str3 As String
        str3 = "Provider = Microsoft.Jet.OLEDB.4.0;" & "Data Source= D:\WS_Data.mdb;"
        con3.ConnectionString = str3
        con3.Open()
        Dim rcs3 As New ADODB.Recordset()
        rcs3 = New ADODB.Recordset()
        rcs3 = New ADODB.Recordset()
        Dim strsql3 As String
        strsql3 = "PlaceName"
        rcs3.CursorLocation = ADODB.CursorLocationEnum.adUseClient
        rcs3.Open(strsql3, con, ADODB.CursorTypeEnum.adOpenKeyset, AD-
ODB.LockTypeEnum.adLockOptimistic, ADODB.CommandTypeEnum.adCmdTable)
        Title = rcs3.Fields(0).Value

        Route_Service = Title

    End Function

End Class
```

Appendix B: Local Application


```
Private Sub Form_Load()
'--------------read inputs from database-------
Dim con As ADODB.Connection
Set con = New ADODB.Connection
Dim str As String
str = "Provider = Microsoft.Jet.OLEDB.4.0; Data Source= D:\WS_Data.mdb;"
con.ConnectionString = str
con.Open

Dim rcs As ADODB.Recordset
Set rcs = New ADODB.Recordset
Dim strsql As String
strsql = "Interesting_Place_Service"
rcs.CursorLocation = adUseClient
rcs.Open strsql, con, adOpenKeyset, adLockOptimistic, adCmdTable

Dim placeType As String

placeType = rcs.Fields(0).Value
```

```
Dim rcs5 As ADODB.Recordset
Set rcs5 = New ADODB.Recordset
Dim strsql5 As String
strsql5 = "UserPosition"
rcs5.CursorLocation = adUseClient
rcs5.Open strsql5, con, adOpenKeyset, adLockOptimistic, adCmdTable

Dim uPlace As String

uPlace = rcs5.Fields(0).Value

Dim rcs1 As ADODB.Recordset
Set rcs1 = New ADODB.Recordset
Dim strsql1 As String
strsql1 = "Current_Location"
rcs1.CursorLocation = adUseClient
rcs1.Open strsql1, con, adOpenKeyset, adLockOptimistic, adCmdTable

Dim Ux, Uy As Double

Do Until rcs1.EOF

If uPlace = rcs1.Fields(2).Value Then
Ux = rcs1.Fields(0).Value
Uy = rcs1.Fields(1).Value
Exit Do
End If

rcs1.MoveNext

Loop

'--------------end read inputs-------------

'--add point----
      Dim pGC As IGraphicsContainer
      Dim pArea As IArea
      Dim pPoint As IPoint
      Dim pElement As IElement
      Dim pMarkerElement As IMarkerElement
      Dim pSimpleMarkerSymbol As ISimpleMarkerSymbol
      Dim pRGBColor As IRgbColor
      Set pGC = MapControl1.ActiveView.FocusMap
      Set pArea = MapControl1.ActiveView.Extent
      Set pPoint = pArea.Centroid

      pPoint.X = Ux
      pPoint.Y = Uy
```

```
        Set pMarkerElement = New MarkerElement
        Set pElement = pMarkerElement
        pElement.Geometry = pPoint

        Set pRGBColor = New RgbColor
        pRGBColor.Red = 0
        pRGBColor.Green = 255
        pRGBColor.Blue = 0

        Set pSimpleMarkerSymbol = New SimpleMarkerSymbol
        pSimpleMarkerSymbol.Color = pRGBColor
        pSimpleMarkerSymbol.Size = 5

        pMarkerElement.Symbol = pSimpleMarkerSymbol
        pGC.AddElement pElement, 0



'----end add point--------

'---------------start measure distance----------------------

Dim pMap9 As IMap
Dim pFlayer9 As IFeatureLayer

Set pMap9 = MapControl1.ActiveView.FocusMap
Set pFlayer9 = pMap9.Layer(0)

Dim pFCursor9 As IFeatureCursor
Dim pFClass9 As IFeatureClass
Set pFClass9 = pFlayer9.FeatureClass
Set pFCursor9 = pFClass9.Search(Nothing, False)

Dim pFeature9 As IFeature
Set pFeature9 = pFCursor9.NextFeature

Dim disMin, disMax, cenX, cenY As Double
Dim num As Integer

disMax = 10000
disMin = disMax

Do Until pFeature9 Is Nothing

Dim name  As String
name = "(" & placeType & ")"

If Right(pFeature9.Value(pFCursor9.FindField("FUNCTIONS")), 12) = name Then
```

```
    Dim ProxOp As IProximityOperator
    Dim dis As Double
    Set ProxOp = pPoint
    dis = ProxOp.ReturnDistance(pFeature9.Shape)

    If dis < disMin Then
    disMin = dis
    cenX = (pFeature9.Extent.Envelope.xMin + pFeature9.Extent.Envelope.xMax) / 2
    cenY = (pFeature9.Extent.Envelope.yMin + pFeature9.Extent.Envelope.yMax) / 2

'--------------add data to database-------
Dim con8 As ADODB.Connection
Set con8 = New ADODB.Connection
Dim str8 As String

str8 = "Provider = Microsoft.Jet.OLEDB.4.0; Data Source= D:\WS_Data.mdb;"

con8.ConnectionString = str8
con8.Open

Dim rcs8 As ADODB.Recordset

Set rcs8 = New ADODB.Recordset

Dim strsql8 As String

strsql8 = "PlaceName"

rcs8.CursorLocation = adUseClient

rcs8.Open strsql8, con8, adOpenKeyset, adLockOptimistic, adCmdTable

If rcs8.RecordCount = 1 Then

rcs8.Delete adAffectCurrent

End If

rcs8.AddNew
Dim ss, InterestingPlace As String

ss = pFeature9.Value(pFCursor9.FindField("FUNCTIONS"))
InterestingPlace = Mid(ss, 1, Len(ss) - 12)
rcs8.Fields(0) = Mid(ss, 1, Len(ss) - 12)
rcs8.Update

Set rcs = Nothing
```

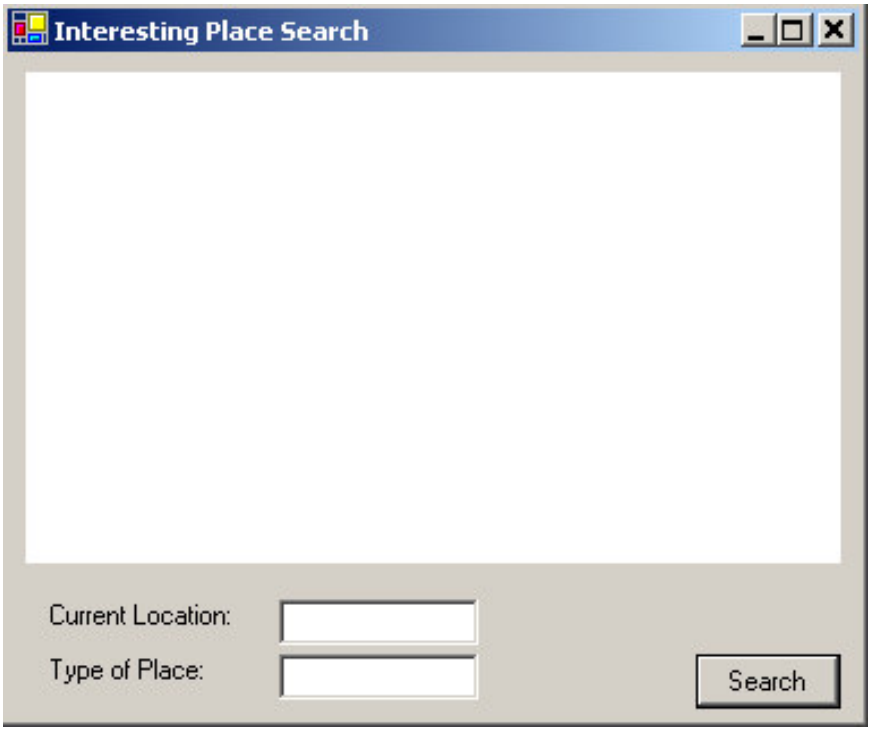
```
'--------------end add data to database-------

   End If

End If

Set pFeature9 = pFCursor9.NextFeature

Loop

'---------------end measure distance----------------------

'-------------- target------------------

Dim pGC1 As IGraphicsContainer
      Dim pArea1 As IArea
      Dim pPoint1 As IPoint
      Dim pElement1 As IElement
      Dim pMarkerElement1 As IMarkerElement
      Dim pSimpleMarkerSymbol1 As ISimpleMarkerSymbol
      Dim pRGBColor1 As IRgbColor
      Set pGC1 = MapControl1.ActiveView.FocusMap
      Set pArea1 = MapControl1.ActiveView.Extent
      Set pPoint1 = pArea.Centroid

      pPoint1.X = cenX
      pPoint1.Y = cenY

      Set pMarkerElement1 = New MarkerElement
      Set pElement1 = pMarkerElement1
      pElement1.Geometry = pPoint1

      Set pRGBColor1 = New RgbColor
      pRGBColor1.Red = 255
      pRGBColor1.Green = 0
      pRGBColor1.Blue = 0

      Set pSimpleMarkerSymbol1 = New SimpleMarkerSymbol
      pSimpleMarkerSymbol1.Color = pRGBColor1
      pSimpleMarkerSymbol1.Size = 5

      pMarkerElement1.Symbol = pSimpleMarkerSymbol1

      pGC1.AddElement pElement1, 0

'---------------end target----------------------------

MapControl1.ActiveView.Refresh
```

```
'-----------------------generate map-------------

'Dim fnum As Integer
'Dim myf As String
'Dim x1, y1, x2, y2 As Single
'myf = "D:\mydata.txt"
'fnum = FreeFile

'-----------------add label-----------
  Dim pTextElement3 As ITextElement
  Dim pElement3 As IElement
  Set pTextElement3 = New TextElement
  Set pElement3 = pTextElement3

  pTextElement3.Text = "Your Position"

  Dim pPointU As IPoint
  Set pPointU = New Point
  pPointU.PutCoords pPoint.X, pPoint.Y
  pElement3.Geometry = pPointU


  Dim pGraphicsContainer3 As IGraphicsContainer

  Set pGraphicsContainer3 = MapControl1.ActiveView

  pGraphicsContainer3.AddElement pTextElement3, 0
  pElement3.Activate MapControl1.ActiveView.ScreenDisplay

MapControl1.ActiveView.PartialRefresh esriViewGraphics, Nothing, Nothing

'---------------end add label----------

'-------------add interesting place ------------------------

  Dim pTextElement4 As ITextElement
  Dim pElement4 As IElement
  Set pTextElement4 = New TextElement
  Set pElement4 = pTextElement4

  pTextElement4.Text = InterestingPlace

  Dim pPointT As IPoint
  Set pPointT = New Point
  pPointT.PutCoords cenX, cenY
  pElement4.Geometry = pPointT

  Dim pGraphicsContainer4 As IGraphicsContainer
```

```
  Set pGraphicsContainer4 = MapControl1.ActiveView

  pGraphicsContainer4.AddElement pTextElement4, 0
  pElement4.Activate MapControl1.ActiveView.ScreenDisplay


  MapControl1.ActiveView.PartialRefresh esriViewGraphics, Nothing, Nothing


'------------end add interesting place-------------------------


Dim pEnvY As IEnvelope
Set pEnvY = MapControl1.Extent

Dim userX, userY, targetX, targetY As Double

userX = pPoint.X
userY = pPoint.Y
targetX = pPoint1.X
targetY = pPoint1.Y

       Dim xMin As Double
       Dim yMin As Double
       Dim xMax As Double
       Dim yMax As Double

      If userX >= targetX Then
      xMin = targetX
      xMax = userX
      Else
      xMax = targetX
      xMin = userX
      End If

      If userY >= targetY Then
      yMin = targetY
      yMax = userY
      Else
      yMax = targetY
      yMin = userY
      End If

pEnvY.PutCoords xMin - 50, yMin - 50, xMax + 50, yMax + 50
MapControl1.Extent = pEnvY


       Dim pActiveViewY As IActiveView
       Dim pExporterY As IExporter
```

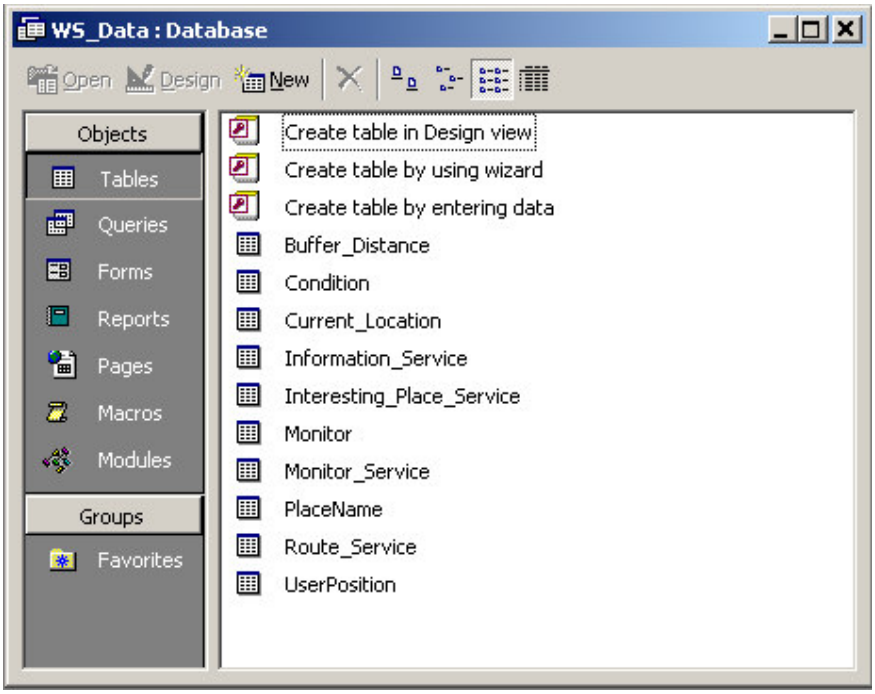
```
    Dim pJpegY As IJpegExporter
    'Dim pEnv As IEnvelope
    Dim exportFrameY As tagRECT
    Dim hdc As Long
    Dim dpi As Integer

    Set pActiveViewY = MapControl1.ActiveView
    exportFrameY = pActiveViewY.ScreenDisplay.DisplayTransformation.DeviceFrame
    'Set pEnv = MapControl1.Extent
    pEnvY.PutCoords exportFrameY.Left, exportFrameY.Top, exportFrameY.Right, exportFrameY.bottom

    Set pExporterY = New JpegExporter
    dpi = 120  'default screen resolution is usually 96

    With pExporterY
       .PixelBounds = pEnvY
       .ExportFileName = "d:\InterestingPlace.jpg"
       .Resolution = dpi
    End With

   Set pEnvY = pExporterY.PixelBounds

   pEnvY.QueryCoords xMin, yMin, xMax, yMax
   exportFrameY.Left = xMin
   exportFrameY.Top = yMin
   exportFrameY.Right = xMax
   exportFrameY.bottom = yMax


    'Do the export
    hdc = pExporterY.StartExporting
    pActiveViewY.Output hdc, dpi, exportFrameY, Nothing, Nothing
    pExporterY.FinishExporting

'----------------------end generate map-----------

Unload Form1

End Sub
```

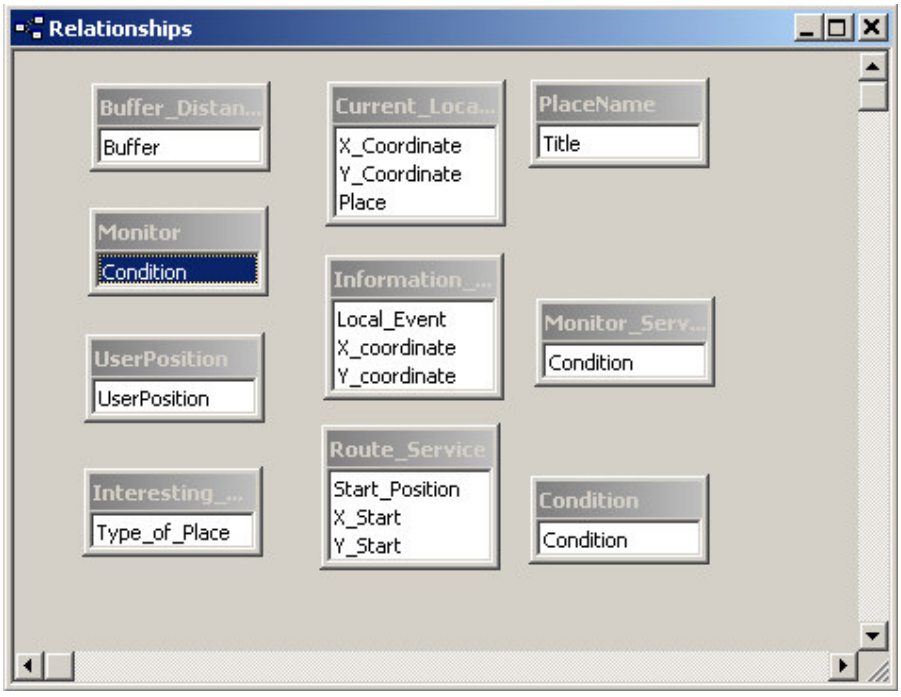
Appendix C: Client Side Application




```
Public Class Form1
    Inherits System.Windows.Forms.Form
    Private Sub ButtonClick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles But-
ton1.Click
        Dim x, y As String
        Dim Search As New localhost1.Service1()
        x = TextBox1.Text
        y = TextBox2.Text

        Dim yy As New localhost.Images()
        Dim data() As Byte = yy.GetImage
        Dim sss As System.IO.MemoryStream = New System.IO.MemoryStream(data)
        'Dim ssss As New System.IO.FileStream()

        Dim bm As Bitmap
        bm = New Bitmap(sss)
        PictureBox1.Image = bm

    End Sub

End Class
```

Appendix D: Database Contents





Filename: Thesis_ChangZheng.doc
Directory: D:\sun\ChangZheng_Thesis
Template: \\itcnt01\msappsuk\msoff2k\custom\templates\ITC Student\Thesis Layout.dot
Title: Thesis
Subject:
Author: chang
Keywords:
Comments:
Creation Date: 17.02.2003 14:16
Change Number: 7
Last Saved On: 17.02.2003 14:47
Last Saved By: chang
Total Editing Time: 8 Minutes
Last Printed On: 17.02.2003 14:47
As of Last Complete Printing
Number of Pages: 92
Number of Words: 19 295 (approx.)
Number of Characters: 109 985 (approx.)