- 150 CLIENT/SERVER COMPUTING AND DISTRIBUTED-COMPONENT FRAMEWORK
- Orfali, R., and Harkey, D. (1997). *Client/Server Programming with Java and CORBA*. New York: Wiley.
- Orfali, R., Harkey, D., and Edwards. J. (1996). The Essential Distributed Objects Survival Guide. New York: Wiley.
- Orfali, R., Harkey, D., and Edwards, J. (1999). *Client/Server Survival Guide*, 3rd ed. New York: Wiley.
- Pleas, M. (2000). Microsoft .NET. PC Magazine, December 5, pp. IP01-IP08.
- Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W. (1991). Object-Oriented Modeling and Design. Englewood Cliffs, New Jersey: Prentice-Hall.
- Schmidt, D. C., and Vinoski. S. (1995). Object interconnections: Comparing Alternative Programming Techniques for Multi-Threaded Servers. Column 5. IGSC++ Report Magazine, Feb. 1995. URL: http://www.cs.wustl.edu/~eschmidt/reportdoc.html.
- Schroeder, M. D. (1993). A State-of-the-Art Distributed System: Computing with BOB. In S. Mullender (Ed.) Distributed Systems. Wokingham, England: Addison-Wesley, Chapter 1, pp. 1–16.
- Seltzer, L. (1998). NT 5.0 Preview. PC Magazine, 17(20), pp. 100-130.
- Shan, Y.-P., and Earle, R. H. (1998). Enterprise Computing with Objects: From Client/ Server Environments to the Internet. Reading, Massachusetts: Addison-Wesley.
- Taylor, D. A. (1992). Object-Oriented Information Systems: Planning and Implementation. New York: Wiley.
- Thompson, C., Linden, T., and Filman, B. (1997). Thoughts on OMA-NG: The Next Generation Object Management Architecture. URL: http://www.omg.org/docs/ ormsc/97-09-01.html, May 11, 2000.
- Vckovski, A. (1998). Interoperable and Distributed Processing in GIS. London: Taylor & Francis.
- Vinoski, S. (1997). CORBA: Integrating Diverse Applications within Distributed Heterogeneous Environments. *IEEE Communication*, February 1997, 35(2), pp. 46–53.
- Weber, J. (Ed.). (1997). Special Edition: Using Java 1.1, 3rd ed. Indianapolis, Indiana: Que Corporation.
- Yang, Z., and Duddy, K. (1996). CORBA: A Platform for Distributed Object Computing. ACM Operating Systems Review, April, 30(2), pp. 4–31.

. .

## **CHAPTER 4**

# TECHNOLOGY EVOLUTIONS OF WEB MAPPING

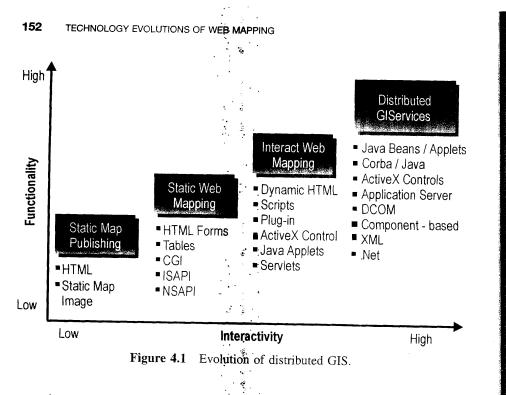
Technology does not stand still, even in this field. It is very likely that new methods will become available in the near and distant future . . . and allow you to include features not possible today. —Brandon Plewe (1997, p. 253)

#### 4.1 INTRODUCTION

The development of distributed GIS is following the progress of computer technologies and telecommunication networks. As we mentioned in the first chapter, it evolved from centralized mainframe GISystems to personal desktop GIS to distributed GIServices that include the applications of wired Internet GIS and wireless mobile GIS. Along with the progress of distributed GIS applications, the technologies adopted by distributed GIS are also changing constantly.

The technology evolution of distributed GIS is shown in Figure 4.1. It started with static map publishing and evolved to static Web mapping, to interactive Web GIS and to distributed GIServices. Static map publishing distributes maps on the Web page as static map images in graphic formats like Portable Document Format (PDF), GIF, or JPEG. It relied on the early stage of Web technology—a giant URL-based HTTP server—to hyperlink ready-made maps on the Web. Maps are usually part of the HTML document to enrich the contents of the document. Users cannot interact with the maps or change their display format in any way.

The second stage is static Web mapping. It involves the use of HTML forms and the CGI to link the user input on the Web browser with GIS or



mapping programs on the servers. Users make requests from the Web browser using customized HTML forms. The request is then sent to the CGI through an HTTP server to invoke GIS or mapping engines. The GIS or mapping engines create the map based on the user's request and generate an image map on the fly. The new image is sent via HTTP back to the user on the Web browser. However, the drawback of the static Web mapping technologies is that the performance of HTTP with CGI is slow, cumbersome, and stateless. Several variations of CGI were developed to improve the performance of CGI, such as Netscape's NSAPI, Microsoft's ISAPI and ASP, NeXT/Apple's WebObjects, Javasoft's servlets, and fast CGI. But the interaction between the user and the maps on the Web browser is still limited. The HTTP form is text based and allows limited user input. Users cannot define or draw a circle or a square on the image maps.

The third stage is the interactive Web mapping, where more interactivity and intelligence are added to the Web client side by using scripts like dynamic HTML and/or client-side applications like plug-ins, ActiveX controls, and Java applets. Some user queries can be processed on the client side without sending requests to the servers. But this approach still requires HTTP connections and the Web servers to mediate between software objects running on the client-side machines and the servers that store these objects.

The fourth stage is the distributed GIS ervices where GIS components on the Web client side can directly communicate with other GIS components on the server without going through an HTTP server and CGI-related middleware. Distributed GIServices rely on the communication between CORBA/ Java ORB or Microsoft's SOAP on the client side and the CORBA/IIOP and server-side Java or .NET/COM + technology in the Microsoft world (we will discuss these in detail late in the next chapter).

#### Key Concepts

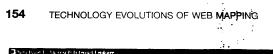
**Distributed GIServices** is a broad term for network-based geospatial information services. There are two major application of distributed GIServices, wired Internet GIS and wireless mobile GIS. This book uses the two terms Internet GIS and distributed GIServices interchangeably. Internet GIS emphasizes the aspect of physical networks and distributed GIS focuses on the distributed access mechanisms of information services.

**Distributed GIServices** refers to a specific software framework where GIS components on the Web client side can directly communicate with other GIS components on the server.

This chapter and the next will cover the underlying technologies that support the evolution of the distributed GIServices. This chapter starts with the early development of static map publishing on the webpage. It then introduces the static Web mapping technology, including HTML forms, CGI, servlets, and ASPs. It follows with the description of interactive Web mapping that covers dynamic HTML and client-side applications such as plug-ins, ActiveX controls, and Java applets. The technologies that constitute distributed GIS will be discussed in the next chapter, which includes the CORBA/Java ORB, CORBA/IIOP, COM+, XML, and Document Object Model (DOM).

## 4.2 STATIC MAP PUBLISHING

A static map publishing refers to embedding maps as graphic images like GIF, JPEG, and Portable Network Graphics (PNG) inside an HTML page. The map images are usually used as a visual presentation to illustrate the points inside the HTML text. But the map image itself is not intelligent. That is, the map image is a static image displayed on the Web browser. The user cannot click on it to zoom to a certain area or get more information. A static map publishing does not support feature data at the client side and does not have map-rendering tools. It is a very thin client application that only supports ready-made map images on the Web browser. To publish a static map image, you can save a ready-made map as a graphic map image format and embed it inside an HTML page. Figure 4.2 illustrates a graphic map example showing a map of a regional park.



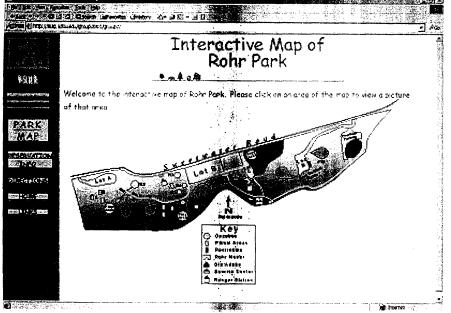


Figure 4.2 Example graphic map image. (Map generated by R. Thornberry, C. Cronk, and K. Hess at San Diego State University.)

In addition, Acrobat's PDF file is another popular method to publish maps on the Web. Figure 4.3 is an example of a PDF map that can be embedded inside an HTML document.

Static map publishing also includes a clickable map. That is, the whole map image is divided into different parts. If you click on one part, additional information on this part of the map will be displayed. For example, if you have a U.S. map and you click on a state on the U.S. map, information about that state or even the separate state map will be displayed. Additional information and maps for that state are separate HTML files or graphic image files that are stored in the Web server as separate files. Figure 4.4 illustrates the park example, which can allow users to click on different playgrounds or picnic area to show the actual photos.

Both the embedded static map images and the clickable maps are simple static map images. They both use the simple Web publishing technology, and no additional technologies are needed. So we treat them in the same category. We will now discuss how this static map publishing works on the Web.

## 4.2.1 Embedding Map Images in HTML Documents

To embed map images inside the HTML document or as separate static image files, you need to first make a map as one of the many graphic image formats,

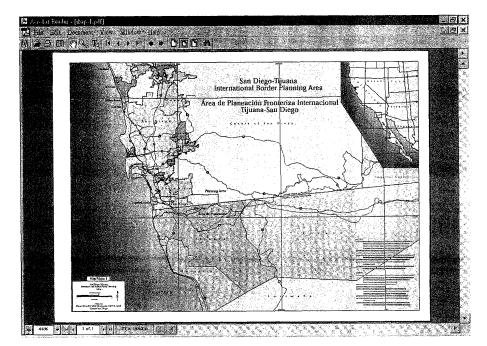


Figure 4.3 Example PDF map, San Diego-Tijuana International Border Planning Area. (Map generated by A. Perry and K. Wells at San Diego State University.)

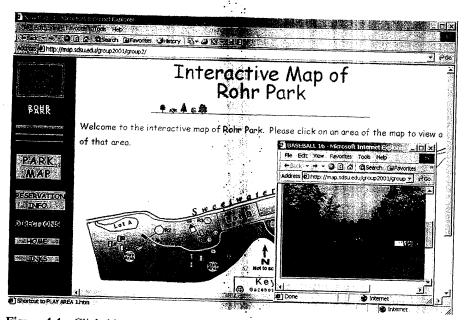
such as GIF, JPEG, and PNG or PDF. You then embed these map image files inside the HTML document using the  $\langle IMG \rangle$  tag or element in the HTML.

The  $\langle IMG \rangle$  element in HTML includes an attribute "SRC" to indicate the file name and its location. For example,  $\langle IMG SRC="/maps/USAmap.gif" \rangle$  tells the Web browser that this is an image file "USAmap.gif" and it is located at the maps directory on the server. There are other parameters associated with the  $\langle IMG \rangle$  tag. For example, the ALLGN parameter tells the browser to place the map images at a certain place on the Web page. The ALT parameter displays the alternative text for nongraphic browsers. The ALT information is important for complying with the ADA (Americans with Disabilities Act) requirements. Here is a simple example:

(IMG SRC=''/maps/USAmap.gif'' ALLGN=''center'' ALT=''A USA map'')

The map image could also be linked with other map images or HTML pages. For example, in the case of

```
〈A HREF=''About_USAmap.html''〉 (IMG SRC=''/maps/
USAmap.gif'')
```



**Figure 4.4** Clickable static Web maps. Users can click on the location of playgrounds or picnic areas to show the actual photos. (Map generated and designed by R. Thornberry, C. Cronk, and K. Hess at San Diego State University.)

when the user clicks on the USAmap image on the browser, the server will return an HTML page (About\_USAmap.html) to the user.

## 4.2.2 Clickable Maps

A clickable map refers to a map that links to separate information about different parts of the map image. For example, if on the U.S. map you click on the state of Wisconsin. the map Wisconsin will be displayed as a new Web page; if on the Wisconsin map you click on Milwaukee, the city of Milwaukee map will show up. Here the terms *map image, clickable map*, and *imagemap* are often confusing. A map image is a geographic map in a graphic image format such as GIF or JPEG. A clickable map is a static map image but can be clicked and linked with other HTML or image files. An imagemap is simply a clickable image file that has hot links under different areas of the image, not necessarily a geographic map.

The clickable map can be created using imagemaps in the HTML page. Clickable images or imagemaps are similar to the static on-line GIF images. They are simply static images. The only difference is that clickable maps have hot spots or links assigned to them. Hot spots are areas of the map image that link to certain URLs.

## 4.2.3 Architecture of Static Web Publishing

Static map publishing uses the simple client/server architecture model as shown in Figure 4.5. It is a simple two-tier client/server model. The client is a Web browser such as Netscape or Internet Explorer, while the server is an HTTP server (or Web server), and the glue is the HTTP. The Web browser handles the presentation element for users to request information and for information to be displayed. The Web server receives users' requests and sends out the file in a user-requested URL. Therefore, this Web client/server model is simply a huge file server that serves files from URLs to all browsers.

**4.2.3.1 The Client: Web Browser** The client in this early Web stage is the simple Web browser with no client-side plug-ins or Java applets. The sole purpose of the Web browser is to interpret the contents of the HTML documents that were sent by the Web server and display them graphically. The Web browser also helps navigate from one page to another using the embedded hypertext links. The Web browser is incapable of interpreting any other documents or data formats except for HTML documents. The use of early Web browsers is similar to the use of a 3270 terminal, a simple display monitor. All the contents are prepared on the server side, and there is little intelligence on the client side. The partition point for this early Web model is at the Web browser, as shown in Figure 4.6. It is a thin-client and thick-server application. As mentioned before, this partitioning that is defined by a protocol (HTTP, in this case) is not flexible.

The client gets resources from the server by clicking a URI (Uniform Resource Identifier). The URI provides a global naming scheme to identify the names and the location of resources on the Web. It identifies the address of a resource and how to access it. A typical URI consists of four parts, as shown in Figure 4.7: the protocol scheme, the server name or domain name, the port number, and the location of target resources.

The protocol scheme specifies the type of protocol to be used to access the resources on a server. URI supports the following Internet protocols: HTTP, FTP, Gopher, Wide-Area Information Server (WAIS), News, and Mailto. A URI to identify a file for downloading would require the "ftp" protocol, such as this:

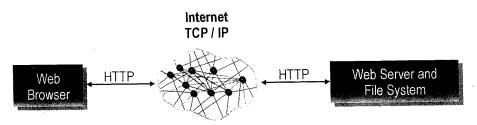
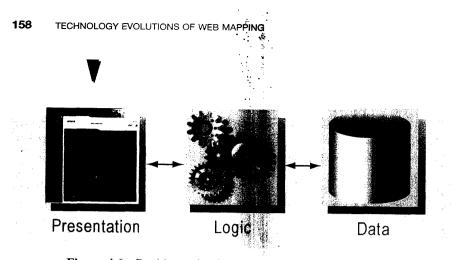
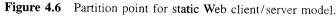


Figure 4.5 Static two-tier Web client/server model.





## ftp://www.GIScompany.com/downloadfiles/street.shp

This would result in a download of the shapefile "street.shp" from the server www.GIScompany.com under the directory of "downloadfiles."

The domain name is the server name to identify the Web server site. It could be a registered domain name such as www.yahoo.com or a numeric IP address such as 129.79.82.108.

The port number is to identify the program that runs on a server. It is specified after the server name using a colon (:) as the separator. If no port number is specified, the browser will direct the request to a well-known port that is associated with a particular program. For example, 80 is usually a port number for accessing HTTP and 21 is a reserved port number for FTP.

#### Key Concepts

**URI** contains information on the specific location of target resources. It typically consists of four parts: the protocol scheme, the domain name, the port number and the location of target resources. **Protocol scheme** specifies the type of protocol to be used to access the resources on a

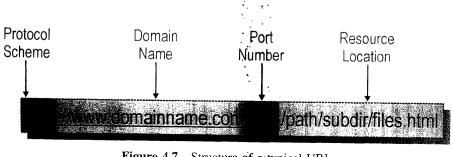


Figure 4.7 Structure of a typical URI.

server. The domain name is the server name to identify the Web server site. A port number identifies the program that runs on a server. The location of target resources specifies the path and the name of resources: document, images, titles, and so on. URL is an informal term associated with popular schemes: http, ftp, mailto, etc.

Finally, the URI contains information on the specific location of target resources. It is a hierarchical description of a file location on the computer. This usually includes a file directory, subdirectories, and files names. For example, /path/subdir/meeting.html indicates that the HTML file meeting.html is stored at the /path/ directory and /subdir/ subdirectory. The URI-supported Web resource can be HTML documents, image files, video clips, and programs such as a CGI or Java applet.

A URI for a program such as a forms-handling CGI script written in Practical Extraction and Reporting Language (PERL) might look like this:

http://www.getcomments.com/cgi-bin/comments.pl

It should be noted that URL is an informal term associated with popular URI schemes: http, ftp, mailto, telnet, and so on. It is no longer used in W3C's technical specifications (http://www.w3.org/addressing).

**4.2.3.2** The Glue: HTTP The glue, or middleware, between the Web client and Web server is the HTTP that is used by the Web to communicate between the Web client and the Web server. Like RPC, HTTP is a request-response oriented protocol. In fact, the HTTP is a *stateless* RPC on top of TCP/IP. That is, for each call from a client, the HTTP establishes a connection between the client and server; the server then fulfills a client request and hands over a reply to the client. The connection is then broken and the server forgets everything it sent out.

## Key Concepts

HTTP is a stateless RPC on top of TCP/IP. A cookie is a text file stored on the client machine to keep the state.

To keep the state, a text file called a cookie is stored on the client machine. For example, e-commerce applications often use cookies to store user-selected items and to remember a user's personal profile so that the next time the user visits the site the information is automatically retrieved. Cookies contain attributes that tell the browser to what servers to send them. But this is not sufficient to support state-oriented client/server conversation, especially the map-based graphic user interface in GIS. For example, the user cannot draw a rectangle on the Web browser because it requires clicking on two points, while the stateless nature of the HTTP does not allow the server to remember the first click once the mouse moves to the second point. Therefore, some client-side applications such as plug-ins, Java applets, and ActiveX controls are developed to enhance the capability of the Web browsers (We will discuss them later in the chapter).

How exactly does the HTTP help clients to communicate with the HTTP server? The answer is through a negotiation process of describing the data type by both the Web browser and the Web server. The HTTP allows Web browsers to inform their server about the type of files they could understand; in return, the server in its response informs the client about the type of data it sends out. Web browsers and HTTP servers use the Internet's MIME (Multipurpose Internet Mail Extensions) data representations to describe and negotiate the contents of the message. MIME is an extension of the original Internet e-mail protocol that specifies a standard to exchange different kinds of data files on the Internet, including audio, video, images, application programs, and others.

An HTTP request from the client consists of a request method (GET or POST), a URI, header fields, and a body (which can be empty). The GET method asks the server to send a copy of the file to the client. The POST method allows a client to send a form's data to the specific URI at the server. When you request a URI in a Web browser, the GET method is used for the request. When you send an HTML form, either GET or POST can be used. With a GET request the parameters are encoded in the URI; with a POST request they are transmitted in the HTTP message body.

The current common protocol is HTTP 1.0, which sets up a new connection for each client request and creates a separate TCP connection to download each URI. This may cause some communication problems while providing multi-tasking services. The newest version of HTTP is 1.1, which has been adopted by some advanced Web servers, such as Apache and Netscape Fasttrack, and some mobile application servers. The new HTTP 1.1 allows persistent connection. That is, HTTP 1.1 keeps the connection with the HTTP server open for multiple request-response interactions. This means that many images embedded in the same HTML document can be downloaded consecutively without breaking and reestablishing the link with the server. The second advance of HTTP 1.1 is pipelining; that is, the client can send multiple requests to the server before waiting for a response. In HTTP 1.0, the client has to wait for the response before sending another request. This means that HTTP 1.1 will allow users to click on two points in a map before sending the request to the server. Therefore, HTTP 1.1 offers the potential capability of drawing a box on a map on a Web browser. Lastly, HTTP 1.1 provides cache validation commands to help clients maintain a consistent local cache of documents. This can reduce the overall network traffic and improve the protocol performance.

**1.2.3.3** The Server: HTTP Server As we discussed before, A Web server s often simply referred to as an HTTP server, which is a daemon that runs

continuously on the server machine. A daemon is a program that runs continuously to handle periodic service requests from other services or programs. The daemon program responds to simple requests or forwards the requests to other programs or processes. Each Web server has an HTTPD (HTTP Daemon) that is continually listening to requests that come in from Web clients and then serving pages out to clients. The role of the Web server is to listen to the client request and respond by sending a precomposed hypertext document or other documents.

## 4.3 STATIC WEB MAPPING

Static map publishing on the Web is simply an electronic copy of a paper map. The users can only take a look at the map images on the Web page and cannot interact with the map in any other ways. This is because the HTTP server in the two-tier Web client/server system cannot handle user requests other than serving ready-made files. To increase user interactivity, Web mapping emerged.

Web mapping refers to making maps, conducting queries, and doing some limited spatial analyses in the server while presenting the output on the standard Web browsers. The output presented on the Web browser is a copy of *static* map images that are generated by the programs in the server. So we call this kind of Internet GIS static Web mapping. The emergence of static Web mapping is the first true representation of the distributed GIServices on the Web.

## 4.3.1 Early History of Static Web Mapping

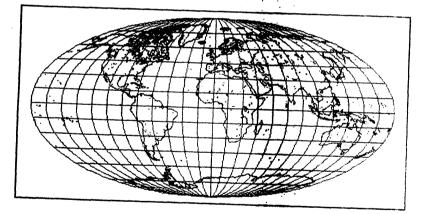
The early research of static Web mapping and distributed GIServices (Gardels, 1996; Plewe, 1997; Tang, 1997) has been motivated by the concepts of an open and distributed architecture. Three projects are of primary importance in the development of GIServices. They are important because these projects initiated the design of preliminary distributed GIServices frameworks and the adoption of early Internet technologies. They provided the GIS community with a glimpse of the potential of the Web and motivated the improvement of geospatial technologies. All three early examples of Internet GIS belong to the category of "static Web mapping" because the process of mapmaking occurred only in the servers.

**4.3.1.1 Xerox PARC Map Viewer** The Xerox PARC Map Viewer was one of the earliest prototype of static Web mapping and was created in June 1993. Map Viewer was developed at Xerox Corporation's Palo Alto Research Center as an experiment in providing interactive information retrieval via the WWW (Putz, 1994). Map Viewer is an interactive Web application that combines the

ability of HTML documents to display graphical images with the ability of HTTP servers to create new documents in response to user input (Figure 4.8).

Map Viewer used a customized server module (a CGI program) written in the PERL scripting language. Map images in GIF format were generated by two separate utility programs on a UNIX server. The first program, MAP-WRITER, produced raster map images from two public domain vector map databases. The second program, RASTOGIF, converted raster images to GIF format. In subsequent work, Xerox Map Viewer was integrated with U.S Gazetteer WWW services created by Plewe (1997) to provide a text-based query function, which is essential for a complete prototype of distributed GIServices. The design of Map Viewer introduced many innovative and advanced concepts in 1994, many of which have since been adopted by other Internet GIS projects. But Xerox did not expand its effort to further the Map Viewer development, and has taken it off line now. The screen shot of the viewer is still available at http://www2.park.com/istl/projects/www94/ mapviewer\_example1.html.

# Xerox PARC Map Viewer: world 0.00N 0.00E (1.0X)



Select a point on the map to zoon in (by 2), or select an option below. Please read About the Map Viewer, FAQ and Details. To find a U.S. location by mame, see the Geographic Name Server.

Options:

- Zoon In: (2), (5), (10), (25); Zoon Dut: (1/2), (1/5), (1/10), (1/25)
  Features: <u>Default</u>, <u>All</u>: <u>tborders</u>, <u>trivers</u>
  Display: <u>color</u>; Projection: <u>eliptical</u>, <u>rectangular</u>, <u>sinusoidal</u>; <u>Harrow</u>, <u>Square</u> Change Database to USA only (more detail) Hide Map Image, Retrieve Map Image Only, No Zoom on Select,
- Place mark at (0.00N 0.00E). Reset All Options

igure 4.8 Xerox Map Viewer. (Figure reprinted with permission from Palo Alto lesearch Center.)

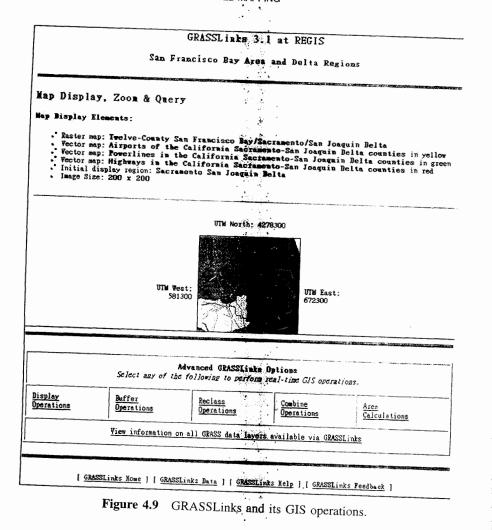
4.3.1.2 NAISMap and World Map Maker NAISMap was developed by the National Atlas Information Service (Natural Resources Canada) in September 1994. NAISMap allowed users to interact with map images (in GIF) on the Web to select map layers, order map layers, and even overlay map layers. The user could select a location from the map; the client application then passed the coordinate to the NAISMap server application, which then returned the requested map areas as a GIF image to the user at the Web client. NAISMap could provide maps in both a national view and regional view. It is an early operational and interactive web-based mapping service released on-line.

NAISMap has been operational for many years since its introduction in 1994. The current Atlas of Canada is its implementation (http://atlas.gc.ca/ site/english/index.html) (Reed, 2002).

Another operational online Web mapping is the World Map Maker that was developed at the Charles Sturt University by Paul Wessel and Walter Smith in 1995 as an integrated Web interface to the GMT 3.x (Generic Mapping Tools) package and a geographic database (http://life.csu.edu.au/cgi-bin/ gis/Map). With GMT, users can create maps on the Web using HTML forms and generate maps as PostScript output files. The World Map Maker is a typical implementation of HTML forms and CGI scripts. The user on the Web client specifies the mapping parameters, which are transferred to GMT commands. "The output of these commands is captured in a PERL scalar variable. The contents of this variable are then written to a temporary PostScript file, which is converted to the GIF format by the GhostScript Program" (Reed, 2002, p. 11).

4.3.1.3 GRASSLinks GRASSLinks was developed in 1995 by Huse (1995) based on her Ph.D. dissertation at the University of California at Berkeley. GRASSLinks was the first fully functional on-line GIService, connecting GRASS GIS software (from the U.S. Army Corps of Engineers) with the WWW. GRASS is a grid-based GIS package offering public domain access to environmental and geographical data. The development of GRASS-Links was supported by the Research Program in Environmental Planning and GIS (REGIS) at the University of California at Berkeley. To utilize GRASS-Links, a user only needed a Web browser to access GIS functions provided by GRASS (Figure 4.9).

The main goal of GRASSLinks was to encourage cooperation and data sharing between different environmental agencies. In traditional GIS applications, each federal/local government agency would maintain its own database as well as data obtained from other sources. GRASSLinks introduced a new model of data sharing where each agency could maintain its own data and access other agencies' data over the network as needed (Huse, 1995). GRASSLinks could perform many GIS operations, including map display, spatial query, overlay operations, reclassification, buffering, and area calculation. On-line users can save their work temporarily on the server and retrieve



saved files later. In general, GRASSLinks demonstrates an ideal prototype for high-end distributed GIS functions and provides an example of the first true on-line GIService (Plewe, 1997).

**4.3.1.4** Alexandria Digital Library Project The ADL project illustrated a digital library framework for heterogeneous spatially referenced information that can be accessed across the Internet. The ADL project was launched in 1994 concurrent with five other digital library projects (NSF, 1994). Many important collections of information, such as maps, photographs, atlases, and gazetteers, are currently stored in a nondigital form, and collections of considerable size and diversity are found only in large research libraries. ADL provides a framework for putting these collections on-line, providing search and access services to a broad class of users, and allowing both collections

and users to be distributed throughout the Internet (Buttenfield, 1998; Buttenfield and Goodchild, 1996; Goodchild, 1995).

The major contribution of the ADL project was to introduce the digital library services metaphor for distributed GIServices and to extend the types of GIServices to cataloging, gazetteer searching, and metadata indexing. Another contribution of ADL was its exploration of Internet-based interface design processes. ADL utilized three different technologies for the actual implementation. The first version ran as a customized ArcView project. The second version was based on HTML and CGI programs (Figure 4.10). The final version utilized Java applets and Java applications. However, the incompatible technologies between the three prototypes caused inconsistent problems of data integration and delivery of services. The ADL user interfaces proved difficult to migrate to each new version, and the task of redeveloping their functions and interfaces was time consuming and costly.

Overall, the ADL project explored different computer technologies and frameworks, identified major tasks of digital libraries, and became the first on-line library service to provide comprehensive metadata browsing, display, and query functions for geospatial information. The recent Alexandria Digital Earth Prototype (ADEPT) is a follow-up to the ADL project. ADEPT aims to use the digital earth metaphor for organizing, using, and presenting information at all levels of spatial and temporal resolution with a specific focus on geodata and images in California.

These early examples of static Web mapping represent milestone achievements for distributed GIServices. Xerox Map Viewer provided a preliminary technical solution for distributed GIServices by using HTTP servers and CGI programs. The technical framework of Map Viewer was followed by many other static Web mapping applications. The development of Xerox Map Viewer also indicated that a single GI service-map browsing-is not sufficient and other GIS functions should be provided. GRASSLinks illustrated a comprehensive prototype that provided many traditional GIS functions, such as map browsing, buffering, and overlaying. However, both Xerox Map Viewer and GRASSLinks were only built to mimic traditional GIS functions. The ADL project introduced a new type of GIService using a digital library metaphor and provided sophisticated library functions, including collections holding, catalog searching, and metadata indexing. Notwithstanding the different functions and interfaces, these three examples adopted the same architecture in developing the applications: server-centered, three-tier Web server architecture.

#### 4.3.2 Architecture of Static Web Mapping

Static Web mapping takes advantage of two advancements in Web technology: Web forms on the Web client side and the CGI on the server side. Web forms are created in the Web client to facilitate user input, and CGI is developed at the Web server to process the user requests. With the introduction of Web forms and CGI, the Web now essentially became a three-tier client/server

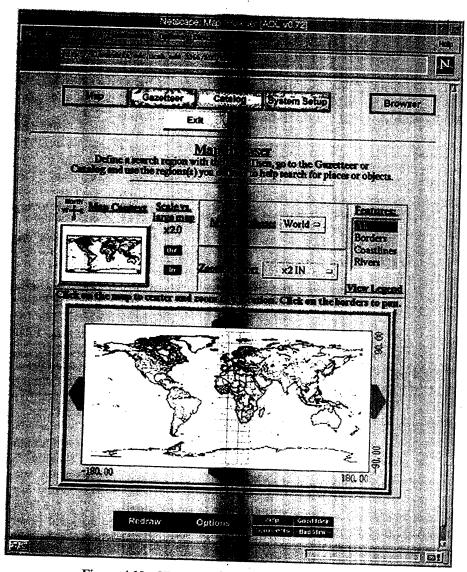


Figure 4.10 HTML/CGI version of the ADL Project.

model, as shown in Figure 4.11. The first tier is the Web client with the function of displaying HTML and forms. The second tier is the HTTP server coupled with a CGI. The third tier consists of traditional application servers such as map servers and DBMS servers.

The Web client is still a simple Web browser with the capability of handling HTML and Web forms. A Web form is an HTML page with data entry fields for user input. The user inputs are collected by the Web browser, which

#### **Internet GIS Showcase: Digital Libraries**

"Digital libraries basically store materials in electronic format and manipulate large collections of those materials effectively. Research into digital libraries is research into network information systems, concentrating on how to develop the necessary infrastructure to effectively mass-manipulate the information on the Net. The key technological issues are how to search and display desired selections from and across large collections" (from the Web site of the Digital Library Initiative (DLI) Phase I http://www.dli2.nsf.gov/dlione).

The DLI was supported by the (NSF) in 1994. There were six major DLI projects funded by four NSF awards during 1994–1998:

- University of California at Berkeley (http://elib.cs.berkeley.edu), Environmental Planning and Geographic Information Systems
- University of California at Santa Barbara (http://www.alexandria .ucsb.edu), Alexandria Project: Spatially Referenced Map Information
- Carnegie Mellon University (http://www.informedia.cs.cmu.edu), Informedia Digital Video Library
- University of Illinois at Urbana-Champaign (http://dli.grainger.uiuc .edu/idli/idli.htm), Federating Repositories of Scientific Literature
- University of Michigan (http://www.si.umich.edu/UMDL), Intelligent Agents for Information Location
- Stanford University (http://www-diglib.stanford.edu/diglib/index .html), Interoperation Mechanisms among Heterogeneous Services

In 1998, the NSF announced *DLI-Phase 2*, which focused on the following issues:

- Selectively build on and extend research and testbed activities in promising digital libraries areas.
- Accelerate development, management, and accessibility of digital content and collections.
- Create new capabilities and opportunities for digital libraries to serve existing and new user communities, including all levels of education.
- Encourage the study of interactions between humans and digital libraries in various social and organizational contexts.

There are currently more than 30 university and DLI projects that are funded under DLI-Phase 2.

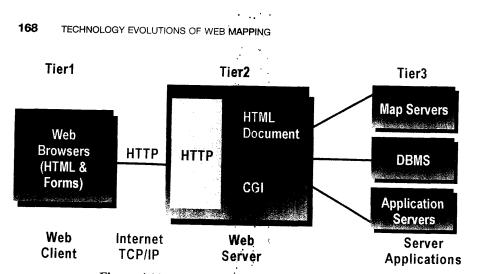


Figure 4.11 Architecture of static map publishing.

invokes a POST HTTP method and sends the user inputs to the server in an HTTP message.

The Web server receives the HTTP message but cannot respond to it because the Web server does not understand any requests other than those for an HTML or other MIME-type document. Therefore, the HTTP server passes the user requests to a back-end program specified in the URI. It uses a CGI to pass the method request and the parameters to the back-end programs.

The back-end programs are traditional server-side applications that do the actual processing. In the case of Web mapping, these back-end programs include map servers and DBMS servers. Any client/server-based GIS programs that work on the server can become a map server. The role of map servers is to fulfill the user requests and return the results to the Web server via the CGI protocol. The Web server then returns the results to the Web client. The Web server becomes middleware, connecting the Web client and back-end server applications.

This is an important and common architecture that dominates the earlystage Web mapping applications. An important characteristic of this architecture is that all user requests are processed by server-side applications. All output at the client side is merely mirrors of map images created by the server. Besides the three examples presented above, some more examples include Visa ATM locator (http://www.visa.com), MapQuest (http://www.mapquest .com), MapBlast! (http://www.mapblast.com), and many others.

## 4.3.3 The Client: HTML Viewers with Forms

#### Key Concepts

Forms are generally used in HTML 3.2 or later versions to gather information from users for a CGI-based server application. METHOD

. .

specifies invocation method for the data to be transmitted to the CGI server application. The GET method encodes the data input in the URI, while the POST method transmits the user input in the HTTP message body. The ACTION attribute specifies the URI where the data are processed.

The client of static Web mapping is the HTML viewer with Web forms. In order for users to interact with back-end map servers, the HTML viewer needs to have two basic functions. The first one is a mechanism for users to enter text, such an address, and/or to select different options, such as selecting different display layers. The second function is to submit the user input and selections to the server. For example, the user may enter an address and then submit the request to the server to return with a map showing the location of the address. After seeing the returned map, the user may decide to zoom in (submit another request) to see a more detailed map in a larger scale.

These two functions can all be made available in Web-based *forms*. Forms are generally used in HTML 3.2 or later versions to gather information from users for a CGI-based server application. A form in HTML starts with a  $\langle FORM \rangle$  tag and ends with the  $\langle /FORM \rangle$  tag. The  $\langle FORM \rangle$  tag has two mandatory attributes: METHOD and ACTION. METHOD specifies the invocation method for the data to be transmitted to the CGI server application. There are two methods, GET or POST. The GET method encodes the data input in the URI, while the POST method transmits the user input in the HTTP message body. The ACTION attribute specifies the URI where the data are processed. The URI is the name of a server and the location of the CGI program or scripts. For example,

(FORM METHOD=''POST''ACTION=HTTP://WWW.dgis.edu/cgibin/geocode)

where the CGI program is the *geocode* program that resides in the cgi-bin directory at the server of WWW.dgis.edu. All CGI programs or scripts are located in the cgi-bin directory. This ACTION attribute tells the Web server that the incoming request is for a CGI program that is located at the cgi-bin directory, so the Web server will invoke the geocode program via the CGI protocol.

A form in HTML generally has three types of interface elements: the *IN-PUT field* for the user to enter data and submit requests, the *SELECT field* to select one option from a list of options in a dropdown list box, and the *TEXT AREA field* to enter multiple-line text input, such as comments. An HTML document can contain one or more forms, but a form cannot have another form nested within it.

A general syntax of an INPUT field is as follows:

(INPUT TYPE=''field-type'' NAME=''variable name''
VALUE=''default value'')

where INPUT indicates this is an input field; it has three properties: TYPE, NAME, and VALUE. The TYPE property indicates an input type. There are eight input types, including text, password, hidden, checkbox, radio, reset, submit, and image. NAME specifies the name of the variable (it is not the displayed name); and VALUE is the actual data value of the variable NAME. NAME and VALUE are a name-value pair to be sent to the server. The VALUE property could have a default value.

The HTML viewer with forms is a thin client; there is no restriction on the operation platform. There is also no requirement on the Web browser or the computing power of the client computer. Therefore, it is mostly applicable to the vast majority of the audience. However, there are some drawbacks of the HTML viewer. Notably, the interactivity between the user and the map image is very limited. The user cannot select a spatial feature or draw a box or a circle on the map due to the *stateless* nature of the HTTP. To improve the interactivity, DHTML or JavaScript could be used. The use of DHTML, VBscript, and JavaScript can allow the user to better interact with the map images.

## 4.3.4 HTTP Server with CGI

The data input from the HTML form is passed to the map server through the HTTP server and CGI. When the user fills the form and clicks on the Submit button, the data input is sent to the HTTP server, which relays the information to the CGI program. The CGI program then interacts with other applications in the server such as a map server and a DBMS server. The map server does the work and returns the results to the CGI program, which reformats the result in an HTML format and sends it to the HTTP server. The HTTP Web server then forwards the results to the Web client.

It can be seen that the CGI is an important middleware to link the Web client and server with a back-end external server application such as a map server. It is a simple language-independent standard interface that runs on top of the operating system to interface external applications with Web servers. It can be used to process user requests that involve computation or invoke other applications on the server. It works on any type of Web server and allows a server to start an external process. The CGI scripts handle the information exchange between the Web server and other server applications such as the map server (Figure 4.12). Basically, CGI is a message-handling protocol or interpreter that receives user inputs and parses them into param-

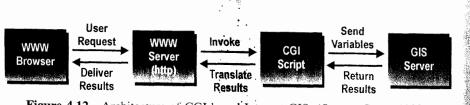


Figure 4.12 Architecture of CGI-based Internet GIS. (Source: Peng, 1999.)

eters of variables to be used in map servers or other GIS programs. It can invoke running map servers and/or other GIS programs, reformat output, and send it back to Web browsers.

A CGI program can be written in any language, such as C or Perl. It can access external resource managers such as files and databases. It is thus a native program. The CGI program can also connect with other applications using any of the communication middleware such as RPC and object messages. CGI programs add more functionality and interactivity to the Web page.

To illustrate how CGI works with the Web server and the map server, we will look at an example step by step [adapted from Orfali et al. (1999)]:

1. A Web user requests a map by typing a street address or a city name and clicking on the Submit request button. The Web browser collects the data within the form and assembles it into a string of name-value pairs that are separated by an ampersand (&). For example,

Street=''2131 E. Hartford Ave.''
&city=''Milwaukee'' &State ='' WI'' &Zip=53211

- 2. The Web browser makes an HTTP request that specifies a POST HTTP method, the URI of the target program in the cgi-bin directory, and the typical HTTP header. The HTTP message body or entity contains the forms data as mentioned in step 1, that is, the string of name-value pairs.
- 3. The HTTP server receives the HTTP request via a socket connection. The server parses the HTTP message and discovers that it is a POST for the cgi-bin program. It then starts the interaction process with the CGI.
- 4. The Web server sets up environmental variables to send parameters to the CGI program. Environment variables are used by the Web server to communicate with the CGI program about the environmental information such as the server name, request method, content types, content length, path and directory, and script name.
- 5. The HTTP server starts a CGI program by executing an instance of the CGI program specified in the URI.
- 6. The CGI program reads the environment variables and discovers, in this case, that it is responding to a POST method.
- 7. The CGI program receives the HTTP message body (i.e., those name-values and name-value strings) via the standard input pipe (stdin) and parses the string to retrieve the form data. It uses the *content\_length* environment variable to determine how many characters to read in from the standard input pipe.
- 8. The CGI program invokes the GIS program or map server and translates the request to a format or set of variables that the map server can understand. Each request is answered in a separate process by a sep-

arate instance of the CGI program, and a new process of the GIS program or map server is launched for each request. The map server creates as many processes as the number of user requests received. More simultaneous requests require the server to create more concurrent processes.

- 9. The map server then processes the request by geocoding the address and making a map centered at the requested address. It then sends the output back to the CGI script.
- 10. The CGI program wraps the output with HTML or some other MIMEtype format by writing the map server output to its standard output stream and sends the output as the HTTP response entity and the HTTP response header back to the Web server. The CGI program then returns the results to the HTTP server via its standard output (stdout).
- 11. The HTTP server receives the results on its standard input and concludes the CGI interaction. The HTTP server sends the results back to the Web browser. Either it can append some response header to the results it received from the CGI program or it sends it "as is" if the response header was added by the CGI program. It then breaks the connection with the Web client.
- 12. At the Web client, the map image is displayed at the Web browser.

The use of HTML forms and CGI to process user requests makes the Web mapping possible. The user can request its own map by specifying the layers and scales. It can also take advantage of the analysis functions of existing GIS programs. But it has four main drawbacks: low performance (a new process has to be created for every request), statelessness, platform dependence, and security concerns (Orfali et al., 1999).

First, every request has to create a new CGI process, which is time consuming and requires large amounts of server RAM. This can restrict the resources available for sharing with other server applications. Therefore, CGI applications do not scale well. When there are many simultaneous requests from Web clients, the system will perform poorly. CGI may become a bottleneck or even a failing point in the whole Web mapping system.

Second, the CGI and HTTP server is stateless. This means that every single request, even a simple zoom in or zoom out, needs to go through the whole process from the Web client to the Weber server to invoking the CGI program and map server and back to the Web client. This creates a lot of network traffic and slows down the whole process. Furthermore, the stateless nature limits the user interactivity at the Web client. In addition, the output is still a map image. Users cannot directly interact with the map.

The third drawback of this approach is that CGI is platform dependent; that is, different CGI programs have to be created for each computing platform.

Finally, CGI programs could pose a security risk to the Web server and other applications on the server because CGI is comprised of native codes that have access to other native programs. Hackers can send malicious codes through the CGI program to infect the server programs.

Two problems of the HTTP and CGI approach—slow performance and statelessness—can be mitigated. The performance of the CGI program can be improved by using the Dynamic Link Library (DLL). The slow performance of the CGI is due to two reasons: (1) the HTTP server has to create a separate process for each request received and (2) when the request is done, the process is then closed. The opening and closing of the process take time and slow down the server response. The DLL serves the function of CGI, but unlike CGI, it stays in memory, ready to service other requests until the server decides it is no longer needed. The ISAPI and the NSAPI are in the form of a DLL. ISAPI works on the Microsoft Internet Information Server (IIS), while NSAPI works on the Netscape Enterprise/FastTrack server. They are used to extend the capabilities of the Web server.

ISAPI and NSAPI DLLs reside in the same process as the HTTP server; therefore, all the resources that are made available by the HTTP server process are also available to the ISAPI or NSAPI DLLs, whereas the CGI applications run in different processes. Some benchmark programs show that loading DLLs in-process can perform considerably faster than loading them into a new process. Furthermore, in-process applications scale much better under heavy loads. Multiple ISAPI or NSAPI DLLs can coexist in the same process as the server. They are multithread-safe to handle multiple simultaneous requests.

The statelessness of the HTTP and CGI can be eliminated by using hidden fields in the HTML forms and/or cookies. A hidden field is invisible from the form but contains values that can be transmitted to the CGI program. The values in the hidden field from previous user input and kept by the Hidden field in the  $\langle INPUT \rangle$  tag so that the user does not need to reenter them each time. For example, if the user entered a street address to request for a geocoded map, when the user received the map from the server, he or she decided to zoom in to a larger scale. How does the CGI keep the state that this user already has a map that centered at a specific address? Well, the CGI uses the hidden field to store information from previous forms to the next. So, when the user later clicks on the "Zoom in" button in the form, the previous user input, such as street address and map extent, becomes a hidden field and is sent to the CGI program through the HTTP request. The CGI program would parse these hidden fields as well as the new INPUT fields and send them to the GIS server to produce a new map. The new map would be sent to the client via the CGI and HTTP server.

Another approach to keep state information is to use *cookies*. A cookie is a small piece of text file that is stored in the client machine. It records user information such as user IDs or other basic configuration information. The role of the cookie is similar to a hidden field; the data stored on the client are similar to the value of the hidden field in the forms. Cookies communicate with the server that creates them every time the user revisits the Web site. Cookies are commonly used in e-commerce Web sites, but they are not often used to keep the state of the Web mapping.

## 4.3.5 Xerox Map Viewer Example

The use of HTTP plus CGI for online mapping dates back to 1989, when the first such system was designed by McDonald Dettwiler and Associates (Reed, 2002). The early client was built using Mosaic (pre-Netscape) to display map images generated from the server. But the most popular example is Xerox PARC map viewer.

The software framework of Xerox Map Viewer is another one of the earliest examples of CGI-based Web mapping application (Figure 4.13). The HTTP clients can submit their requests by sending a URI to the HTTP server. The HTTP server will parse the URI strings and then launch the CGI extension to invoke two Perl programs (MAPWRITER and RASTOGIF) located on the same machine.

Map images in GIF format were generated by the two separate utility programs on the Sun UNIX server. The first program, MAPWRITER, produced raster map images from two public domain vector map databases. The second program, RASTOGIF, converted raster images to GIF format.

After the RASTOGIF generates a new GIF image, the CGI program on the server will create a new HTML with the new GIF image link and POST the new HTML file back to the client-side Web browser.

Here are the two examples of encoded URI requests for Xerox Map Viewer. First, the URI request includes the following commands:

• • •

• border=1 (turn on the country border theme)

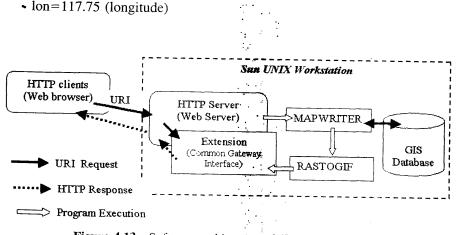


Figure 4.13 Software architecture of Xerox Map Viewer.

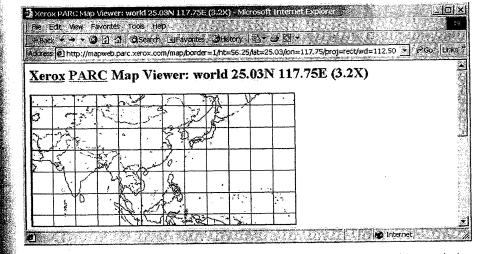
- lat=25.03 (latitude)
- proj=rect (projection uses rectangles)

The CGI programs located on the server-side process the request and generate a new image, such as Figure 4.14. Figure 4.15 is another example of URI requests sent by users generated by the CGI programs:

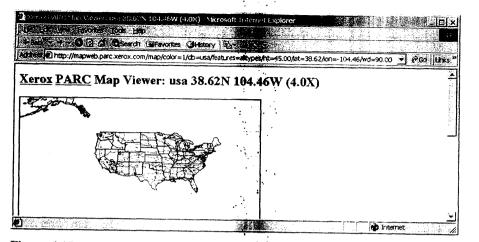
- color=1 (turn on the color display)
- db=usa (access U.S. database)
- feature=alltypes (turn on all types of features)

## 4.3.6 Map Servers and Other Server-Side Applications

The map server is the actual workhorse that processes the user requests. You could develop your own map server to serve a special request, such as a travel plan program. You could also use existing client/server GIS programs as map servers, such as ArcInfo, ArcView, GeoMedia, or MapInfo. The advantage of using existing GIS programs is that the GIS functions that were developed in the program can be used by the user on the Web browsers. But the existing desktop GIS programs were not designed for the Internet and thus do not scale well in the Internet and Web environment. Since each user request has to create a process, the map server has to be able to create multiple processes. This requires you to have a map server with sufficient user seats or user licenses.



**Figure 4.14** Xerox PARC Map Viewer request 1. (Figure reprinted with permission from Palo Alto Research Center.)



**Figure 4.15** Xerox PARC Map Viewer request 2. (Figure reprinted with permission from Palo Alto Research Center.)

Besides a map server, other server applications such as DBMS could also be run to fulfill the user request. In fact, in a more complex user request, the map server has to work together with other server applications, such as a DBMS server and TP Monitor to produce the output that the user requested. TP Monitor could be used to manage and balance the load of user requests. It is essential to handle a lot of simultaneous requests.

In summary, Web mapping based on HTTP/CGI provides somewhat limited interactivity and functionality between the Web client and the map server. It can do a reasonable job in creating customized maps on-line under a low load of user requests. There is no restriction on the client-side computer platform. Anyone with a Web browser is able to make maps on demand.

However, Web mapping using HTTP/CGI as middleware does not scale well. It does not offer users the desktop GIS feel and functions. Users cannot directly interact with the maps, as they can in desktop GIS. Every interaction has to go through the forms, which is an indirect interaction. Users cannot draw a box or square or query a spatial element directly from the map. CGI has serious shortcomings. First, it is stateless and does not maintain the state between connections. This poses major obstacles in stateful GIS operations. Second, the CGI has to load the GIS program into memory for each request. This hogs a lot of server resources and creates considerable performance limitations.

#### 4.4 INTERACTIVE WEB MAPPING

A simple HTML viewer with forms is very limited in terms of user interactivity, especially when dealing with maps and spatial objects. To create more interactive Web mapping, we need alternative viewers that can facilitate the user to interact with the spatial object and maps directly. Therefore, dynamic HTML and client-side applications such as plug-ins or help programs, Java applets, and ActiveX controls were developed to handle maps and spatial objects. These are dynamic or interactive viewers for users to directly interact with spatial objects that are interactive Web mapping applications.

One major characteristic of the interactive Web mapping applications is that they offer more interactions between the user and the client interface and more client-side processing and functionalities than the static Web mapping applications. Many current Internet GIS programs such as Arc IMS, Geo-Media Web Map, MapXtreme, and MapGuide belong in this category.

Another characteristic of interactive Web mapping is that CGI extensions are used as the middleware to mitigate the shortcomings of the CGI. These CGI extensions include Netscape's NSAPI, Microsoft's ISAPI and ASP, Apple's WebObjects, Javasoft's servlets, Allaire's ColdFusion, and many others. These CGI extensions generally perform better than the CGI scripts. The common feature of these CGI workarounds is that they all run some sort of server-side scripts (or plug-in codes) in the same address space as the Web server.

Most current interactive Web GIS programs are based on this model, that is, a dynamic viewer coupled with CGI or CGI extensions. This section will describe the characteristics of different client viewers and server-side CGI extensions.

#### 4.4.1 Interactive Viewers

Client viewers are places for users to interact with maps and spatial objects. Different interactive viewers with various functions have been developed using different programs and technologies, from very simple HTML interfaces with forms as we discussed in the previous section to dynamic HTML and to more advanced client-side applications such as plug-ins, ActiveX controls, and Java applets. We will introduce four types of client viewers in the interactive Web mapping programs: DHTML viewer, GIS plug-ins, Java applets, and ActiveX controls.

#### Key Concepts

The HTML viewer is static or noninteractive for Web mapping. DHTML makes plain HTML dynamic by using client-side scripting, DOM, and CSS. Plug-in viewers are software executables that run on the browser to extend the capabilities of Web browsers. A Java applet is an executable Java code that is downloadable from the server and executed on the client at runtime. A Java applet viewer displays geospatial information and handles requests. ActiveX viewers use ActiveX controls to program the viewer. An ActiveX control is a modular piece

#### 178 TECHNOLOGY EVOLUTIONS OF WEB MAPPING

of software that performs tasks and communicates information to other programs and modules over the Internet via OLE.

**4.4.1.1 DHTML Viewer** The HTML viewer is static or noninteractive for Web mapping. After a page and map are loaded in the browser, they become static. The only things a user can do in a static web page are:

• If there is a link, click on it.

• If there is a form or image form, fill it out and click on the submit button.

The response to either of the above is not all that quick because, in either case, the page appears after a complete round trip to the server and back, even for a simple response such as zooming into a map feature. This is where DHTML comes in.

DHTML is just plain HTML that can change even after a page has been loaded into a browser. An area of a map can change color when the mouse moves over it or a menu can drop down or a new popup window appear. Most HTML elements can be made to react to user actions after the page loads. The DHTML viewer has three major advantages over the static HTML viewer:

- The Web page and the map will respond to user actions.
- That response is immediate (without making a round trip to the server).
- No special plug-in is needed to install at the browser.

DHTML makes static HTML page dynamic by using the following:

- 1. *Client-Side Scripting* Client-side scripting uses JavaScript and VBScript to change HTML. VBScript works only in Microsoft Internet Explorer, while JavaScript works in other browsers as well.
- 2. Document Object Model The DOM is the hierarchy of elements that are present in the browser. This includes browser properties such as the browser's version number, window properties such as window location (the page's URI), and HTML elements such as (p) tags, or tables. For example, you can point to the specific check box in a specific form in a page and make it checked. Thus, by exposing the elements and their properties to scripting languages, browsers enable the user to manipulate them. The DOM also specifies the events that get triggered as a result of a user action. For example, the DOM defines an event "on-MouseOver" for a link. This enables you to write a script for something to happen when a user passes the mouse over that link.
- 3. Cascading Style Sheets (CSS) CSS not only let the user specify style information in one place for an entire Web but also allow the user to set style values in such a way that they can be easily manipulated by a

.

scripting language. By changing the CSS properties of a page element (such as its color, position, or size), it is possible to change almost anything about the way a page looks.

**4.4.1.2 Plug-In Viewer** Plug-in viewers are software executables that run on the browser to extend the capabilities of Web browsers. Plug-in viewers can support both vector data and raster images. The role of plug-in viewers is to provide user interactivity with geospatial data and map images so that the user can view the maps and select features and make queries directly on the map. While plug-in viewers are small applications installed in the Web browser, GIS helper programs can be large GIS applications or existing GIS software that is located in the user's local machine. GIS software such as ArcView, MapInfo, and GeoMedia can all be GIS helper programs. When the Web browser detects a GIS data type in an HTML page, it can automatically launch the respective GIS helper program (though this function is not yet available for many existing desktop GIS programs).

## Internet GIS Showcase: Brownfield Location Information System

The Brownfield Location Information System (BLIS) is Wisconsin's effort to promote the redevelopment of underused properties throughout the state. Potential redevelopers can use the site's map and query system to locate the tax-delinquent, abandoned, blighted, or hazardous site that best fits his or her needs and selection criteria. BLIS is designed to help commercial, industrial, and retail businesses locate reusable land while simultaneously assisting landowners market their sites.

To provide easy access, BLIS was developed so that users need only a Web browser that can handle HTML and JavaScript. There are no applets or extra plug-ins required. Buttons and functions on the site are intended to be intuitive, even to users who are not familiar with mapping or GIS software. Users are able to perform drag-box zooms (without a Java applet) and view the 10 best sites for their query criteria, not just sites that meet all their criteria. Also, the map size is sizable variable and will always fit the maximum available space in a user's browser.

Map clutter on BLIS was reduced by limiting which layers users are able to toggle on and off and by setting zoom thresholds where certain layers appear and disappear automatically. The incorporation of an "identify" function lessens the need for labeling all features.

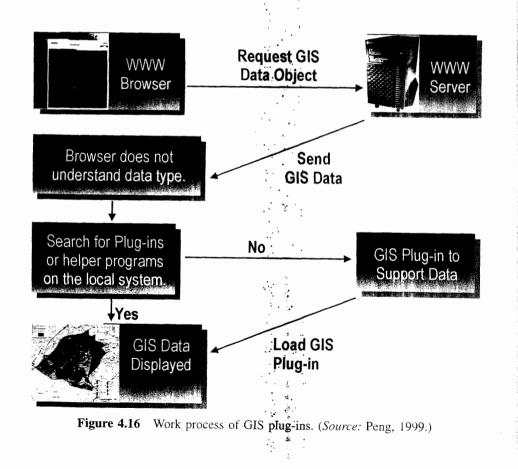
BLIS uses ArcIMS technology developed by ESRI for displaying and interacting with a map.

Source: http://comgis1.commerce.state.wi.us/wiscomp/blis\_start.htm.

In the case of direct feature data support, plug-in viewers can communicate with vector geospatial data. Similar to other plug-ins, GIS plug-in viewers handle GIS data from a URI that is provided as a stream as it arrives from the network. This allows a GIS plug-in to implement a progressive viewer without seeing an entire stream. Individual plug-ins can request multiple data streams simultaneously.

Figure 4.16 illustrates the working process of a GIS plug-in viewer. When the Web browser encounters a geospatial data type (most GIS data types are unknown to Web browsers) in a Web page from a server, it will look for a plug-in that is associated with that data type and then load it. If a GIS plugin or helper program is not available in the client computer, it has to be downloaded from the Web server over the network. Once the GIS plug-in or helper program is installed, it then communicates directly with the GIS data stream from the server.

**4.4.1.3 Java Applet Viewers** Viewers in the form of Java applets are executable Java code that is downloadable from the server and executed on the



## Internet GIS Showcase: City of Nanaimo

The City of Nanaimo, Canada, has used MapGuide to create an on-line mapping tool for public use. Nanaimo's city map provides users with the ability to

find a street address on the map,

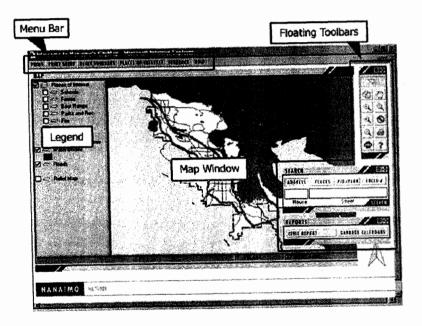
print a replacement garbage calendar,

review school catchment areas,

locate city parking lots,

locate parks and trails, and

search for common points of interest such as shopping centers, schools, and hospitals.



The Civic Report is a Web-based document that provides information about a selected property. The Civic Report provides

- · legal description information,
- address information,
- zoning information,
- · garbage calendar route information, and
- · sewer service information.

Source: http://www.city.nanaimo.bc.ca/citymap.asp.

client at runtime. Java applets initially reside on the Web server. They are referenced inside an HTML document and executed by a Web browser at the client side. These applet files are downloaded and executed when a user connects to the Web site and invokes the HTML document containing a reference (the (applet) tag in the older version of HTML or the (Object) tag in HTML 4.0 or later) to the Java applet. The Java applet is then seamlessly integrated inside the Web browser. Figure 4.17 illustrates the process of loading a Java applet viewer.

Java viewers use a Java applet for displaying geospatial information and handling requests. It allows the user to interact directly with the spatial features on the map. The Java viewer usually incorporates map-rendering and data processing functions in the Java applet so that the user can render maps, make queries, and do other processing inside the viewer without going back to the map server.

Java viewers can support both feature data and map images. In the case of supporting map images, the Java applet is simply a fancy display of map images similar to the DHTML viewer. The user interacts with map images. But all logic processing such as map rendering and query processing is conducted at the server.

If the Java viewer supports streamed feature data, the data server streams the vector data to the Java viewer, and the user then interacts with the feature data. Some or all processing can be performed at the client viewer. Java viewers have the potential to support unlimited client-side processing. Feature data that are streamed to the Java viewers are temporarily cached on the client machine. The Java viewer establishes a connection channel between the Java viewer and the database server via a JDBC drive. When the user's request requires data that are not currently in the cache, the request is sent to the

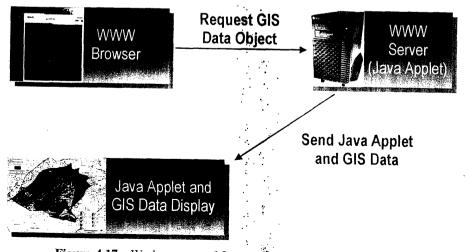


Figure 4.17 Work process of Java applets. (Source: Peng, 1999.)

server to either retrieve more data or process data residing on the server. The temporary cache is removed when the Java viewer is closed. So the Java viewer does not take any permanent disk space in the client's computer.

In addition to Java applets that work with a Web browser, a whole Javabased viewer, Java Beans, can also be created to work as an independent viewer or a mobile client agent.

Many GIS vendors are developing Java-based viewers, such as ESRI's ArcIMS Java viewer (Figure 4.18).

**4.4.1.4 ActiveX Viewer** ActiveX viewers use ActiveX controls to program the viewer. ActiveX was developed by Microsoft to "activate the Internet." It builds on the OLE standard to provide a common framework for extending the capability of Microsoft's Web browser, Internet Explorer (Chappell, 1996).

An ActiveX control is a modular piece of software that performs tasks and communicates information to other programs and modules over the Internet via OLE. It can also be used and reused by any programming language or

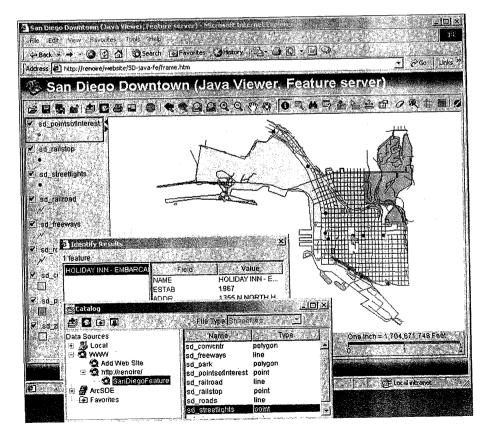


Figure 4.18 ESRI ArcIMS Java viewer.

application that supports the OLE standard (Chappell, 1996). ActiveX controls are general componentware that can plug into any application. There are many different types of ActiveX controls, each with different capabilities and functionalities. Client viewers implemented in ActiveX controls are created to handle GIS data and as a graphic interface between the user and the feature data and/or maps over the Web browser. They can be used just like plug-ins and Java applets within Web pages.

ActiveX viewers are referenced inside an HTML document and executed by a Web browser on the browser's machine. The GIS control is downloaded from a Web server when it is needed or it might already be present on the client machine if it was previously downloaded.

The ActiveX viewer can access a URI and retrieve GIS data just as a standard Web browser client does. GIS data are streamed asynchronously to the GIS control as the information arrives from the network, making it possible to implement viewers and other interfaces that can progressively display information. If the GIS control needs more data than can be supplied through a single data stream, multiple and simultaneous data streams can be requested by the ActiveX viewer. Furthermore, ActiveX viewers can be set up to allow the user to combine local data and streamed feature data from the server to process request in the same ActiveX viewer. Figure 4.19 illustrates the simple architecture of the ActiveX viewer.

While Java applets are in the form of byte codes and have to run inside a Java VM, Activex is native binary code and runs directly inside the computer's native operating system. Being a native code, ActiveX controls can take full advantage of the local computer's computing power and have direct access to all local platform functionality such as local files, local memory, and other system resources that are unavailable to a Java applet. Therefore,

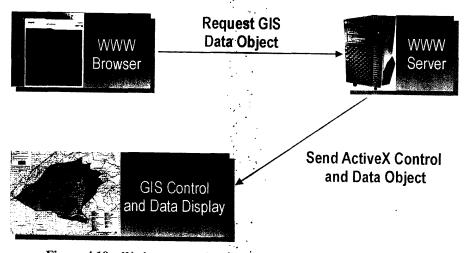


Figure 4.19 Work process of ActiveX controls. (Source: Peng, 1999.)

compared with a Java applet, ActiveX components have better performance (Shan and Earle, 1998).

However, this performance advantage also carries a price—portability and safety concerns. First, because ActiveX controls are compiled to the native executable format, different versions of ActiveX controls have to be made available for all platforms. That means ActiveX controls are platform dependent. This is in contrast with the Java applets, which are platform independent. One Java applet code (theoretically) can be run on all platforms. Second, because ActiveX controls are able to access to local files and other local resources, they could pose greater danger to the users' local computers. Someone could write a vicious ActiveX code that could erase all local files. (We will discuss the safety issues in more detail in Chapter 10.)

Figure 4.20 illustrates an example of the ActiveX viewer developed by INTEGRAPH's GeoMedia Web.

Another interesting implementation of the ActiveX controls is Citrix's WinFrame Web Client ActiveX control (http://www.citrix.com/). The Win-Frame Web Client ActiveX control can allow users to access GIS data and applications remotely. WinFrame Web Client uses the techniques of Application Launching and Embedding (ALE) to run applications remotely from a Web page. Users can launch an application from a Web page by clicking on

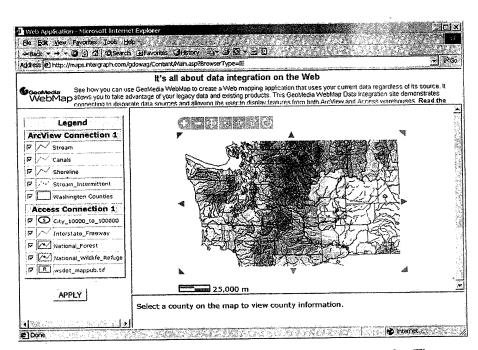


Figure 4.20 GeoMedia Web map viewer, an ActiveX viewer example. (Figure reprinted with permission from Integraph.)

a hyperlink. The server will launch a GIS application in a window on the user's local desktop. The launched application can run on its own window or within the Web page (embedding). The users can then use this application as if it were installed and running locally on his or her own machine. They can also use the application to edit and manipulate the information, save it to a local file, or even save it back to the remote site (if the file permissions allow).

In summary, the viewers for interactive Web mapping could be as simple as a dynamic HTML page or as complex as a Java applet or ActiveX control. But regardless of the form of the viewer, a client viewer should be able to display a map, make a query, identify a data source, extract a subset of data from a data server, and have the result rendered on the screen in the form of a map image or a vector map. More advanced clients would have added functions and capabilities, such as editing, data integration, and spatial analysis.

## 4.4.2 Server-Side CGI Extensions

Those client viewers are linked with a map server and data server via CGI or CGI extensions. To overcome the previously discussed shortcomings of the CGI, several extensions are used to extend the capabilities of CGI in interactive Web mapping. Rather than create a separate process and close the process for each request received, as the CGI does, the server-side CGI extensions stay in the memory and are always ready to service other requests. Furthermore, unlike CGI programs that run in different processes, the CGI extensions run server-side scripts in the same address as the Web server. Therefore, all the resources that are made available by the HTTP server process are also available to CGI extensions. Loading CGI extension programs in-process can improve performance considerably compared to loading them into a new process. We have covered NSAPI and ISAPI in the last section; we will discuss servlets, ASP, and ColdFusion here.

4.4.2.1	Servlets	
Key (	Concepts	

Servlets are modules of Java code that run in the Web server to extend the capabilities of the HTTP server. ASP is a Web server extension to receive and process user requests on the Microsoft Internet Information Server (IIS). ColdFusion is a Web application server that runs on a Web server that works with Linux, Solaris, and Windows servers.

Similar to "applets" on the client side that are used to extend the capabilities of the Web client, servlets are modules of Java code that run in the Web server to extend the capabilities of the HTTP server (e.g., to answer client requests). Servlets are commonly used with HTTP servers, so they are often

•

referred to as "HTTP servlets," even though they can be used with any client/ server protocols. Since servlets are written in the highly portable Java language and follow a standard framework, they can be used independently of server types and operating systems.

HTTP servlets are similar to CGI scripts and usually have the following major functions (Orfali et al., 1999):

- processing and/or storing data submitted by an HTML form;
- providing dynamic contents (e.g., returning the results of a database query to the client);
- managing state information on top of the stateless HTTP (e.g., managing many concurrent requests for the same map services);
- initiating a connection to a database and maintaining its connection across requests;
- passing a client request to another servlet, a feature called servlet chaining; and
- providing an interface between Web users and a legacy (mainframe) application and its database.

Compared to CGI, servlets have several advantages (Orfali et al., 1999):

- While CGI runs in a separate process, a servlet does not run in a separate process. This removes the overhead of creating a new process for each request every time, thus improving the performance.
- Similar to ISAPI and NSAPI DLL, a servlet stays active in memory between requests, while a CGI program needs to be loaded and started for each request. This is another way to increase responsiveness and performance.
- There is no need to create multiple instances to respond to multiple requests. Only a single instance is needed to answer all requests concurrently. This saves memory and allows a servlet to easily manage persistent data. Servlets are multithread safe.
- A servlet can be run by a servlet engine or servlet container in a restrictive sandbox just like an applet runs in a Web browser's Java VM, which increases the server security.

Because of these features, servlets become a pretty good alternative to the CGI programs, especially for simple applications. Servlets are better than CGI at accepting form input, interacting with a single database, and dynamically generating an HTML response page. They provide functions to easily extract the HTTP name-value pairs and compose a dynamic HTTP response.

Servlets do a good job for a simple CGI-like request-response system. However, the servlet is just a little better than CGI as Web middleware. It is still very primitive. Since servlets use a generic API, they have a set of predefined methods. Therefore, you have to do your own marshaling and unmarshaling of parameters. Also, servlets do not support typed interfaces; you have to create your own command formats. Servlets do not fit well with the distributed-object system. They cannot take advantage of object interfaces and do not have the features that many scalable server-side component technologies provide, such as transaction (Orfali et al., 1999).

**4.4.2.2** Active Server Page An ASP is another server-side feature or Web server extension to receive and process user requests on the Microsoft IIS. It is used to replace the CGI scripts on the Web server. An ASP is essentially an HTML page that includes one or more scripts (small embedded programs). These scripts are processed on a Microsoft Web server before the page is sent to the user. The user accesses the ASP Web page on the Web browser; the user requests are then sent to the ASP scripts on the server. The script in the Web page at the server then accesses data from a database and builds or customizes a page on the fly before sending it back to the user at the browser. In other words, the server-side ASP script simply creates a regular HTML page or ASP file by processing the user requests and/or extracting data from the database on the server.

ASP scripts can be written in either VBScript or Jscript. User requests can be fulfilled using ADO program statements. Scripts can reside in either the server side or the client side. Client-side scripts create more interactivity while scripts on the server side are more versatile and have no limitation on browsers.

**4.4.2.3 ColdFusion** ColdFusion was developed by the Allaire Corporation, which has merged with Macromedia, to be an alternative to Perl and other CGI technologies. It is a Web application server that runs on a Web server that works with Linux, Solaris, and Windows servers. Similar to ASP, the ColdFusion Web application server works with the HTTP server to process user requests for Web pages. When a user sends a request from a Web browser for a ColdFusion page, the ColdFusion application server executes the scripts or programs the page contains.

ColdFusion can create and modify variables just like other languages. It has some built-in functions for performing some complicated tasks. Cold-Fusion applications can access databases using Microsoft's OLE DB, ODBC, or drivers that access Oracle and Sybase databases. Just like ASP, ColdFusion uses standard SQL to link Web pages and Web applications with the backend data servers to retrieve, store, format; and present information dynamically.

ColdFusion uses its proprietary markup language CFML (ColdFusion Markup Language) to make web programming. CFML encompasses HTML and XML and is tag based. A JIT compiler turns the CFML into Web pages to be served to the Web client. CFML has 70+ CFML tags and over 200 custom functions. It also offers tools similar to those at the server side CGI

· :

extensions to extend the server-side functions. ColdFusion can be coordinated with distributed applications that use CORBA or Microsoft's DCOM to interact with other network applications. ColdFusion also has tags to embed COM, CORBA objects, and Java applets/servlets.

In addition to the middleware or application server between the Web server and the map server, there are other services on the server, including catalog services, load balance services, and state services. Catalog services keep track of where the data are in a distributed environment; load balance services balance the load of different server functions; while the state services keep the state of user requests.

These CGI-like middleware and other services are connected with mapping servers and database servers. It is the mapping server that fulfils the user request and makes maps. The middleware is simply a translator that receives requests from the Web browser and forwards the requests in a proper format to the map server for process.

Notice that in the interactive Web mapping applications, although the client side could use the Java applets and ActiveX controls, the middleware is still CGI or CGI extensions. This Java-to-CGI client/server approach is still the traditional Web client/server architecture. It is different from the distributed GIS, the next phase of distributed GIS, as we will discuss in the next chapter.

Although interactive mapping programs, whether static or interactive, are very popular and well recognized in the GIS community, there are common problems: The performance is slow, the functions are limited, and they are proprietary and not interoperable. Different Web mapping programs were developed in different database frameworks and using different technologies. The heterogeneous techniques and software programs prevented the integration and sharing of information among these Web mapping programs. Furthermore, it is difficult to migrate technology from one platform and one stage to another, as demonstrated in the ADL project.

The problem will get worse as more vendors start developing Web mapping programs. OGC has been making efforts to develop a set of standards to guide the development of Web mapping programs so that they can be interoperable. Therefore, OGC developed Web Map Server (WMS) implementation information specifications based on some Web mapping testbeds or pilot programs, which represents the first effort to standardize the implementation of the Web mapping programs.

# 4.5 OPENGIS WMS IMPLEMENTATION INTERFACE SPECIFICATIONS

#### Key Concepts

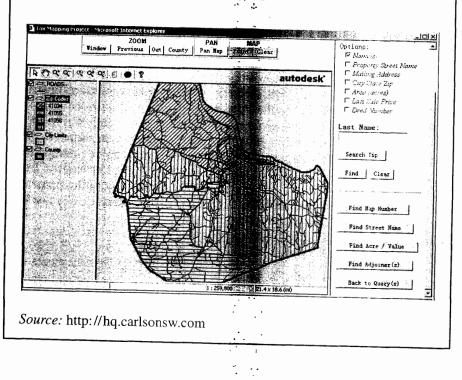
The OpenGIS **WMS implementation interface specifications** provide guidelines for current Web map servers with the specifications of HTTP contents and URI communication syntax. Its specifications also lay out

# Internet GIS Showcase: Mason County, Kentucky, PVA Project

This demonstration model was created to showcase the benefits of an Internet/intranet-based mapping system. The local Property Valuation Administration (PVA) office, with cooperation from the State of Kentucky, provided the digital, graphical, tabular, and geographic data for this project

The application combines the PVA database information with digital maps and images. The process is fulfilled by incorporating a dynamic HTML (Web page) editor. The Web page design program used in this application was ColdFusion, from Allaire Corporation. ColdFusion allowed the developer to easily integrate the database information into an HTML (Web page) format. Also, ColdFusion was used to build dynamic queries from the database and relate them to the MapGuide viewer, a Web browser client.

This site combines the ColdFusion pages with the viewer to have a complete page showing the PVA tracts and database entries from simple queries. In the next stage, actual lot survey drawings, descriptions, photos, and so on, will be attached to each parcel.



the major tasks of Internet map servers, which can be applied in the architecture of distributed GIServices.

#### 4.5.1 Background and Overview

One of the main goals of OGC is to come up with a set of specifications to be used as a guide to Internet GIS design for different software vendors so that their designed systems can be communicated or interoperated with each other. OGC's interoperability programs cover a range of areas, from geospatial data to geospatial processing. Web mapping is one of them. OpenGIS specifications result from common understandings and the consensus of the GIS vendor industry as well as from experiences learned from different testbeds or pilot programs.

OGC's Web mapping activities started with a WWW mapping framework by Doyle (1997). A task force of OGC was formed to come up with a consensus position on the WWW mapping Special Interest Groups (SIG) that is described in "User Interaction with Geospatial Data" by Cuthbert (1997). This document presents an abstraction for the display of geospatial data. It provides a common set of terms that can be used to describe a variety of software implementations. Based on the basic ideas from Doyle (1997) and Cuthbert (1997) as well as from "A Web Mapping Scenario" by Gardels (1998), OGC sponsored the Web Mapping Testbed (WMT) initiative. The WMT initiative invited GIS software vendors as well as governmental agencies to design pilot Web mapping systems to test implement the ideas in the WWW mapping framework and OGC consensus position papers. The WMT demonstration was made in September 1999. The OpenGIS WMS interface implementation specification was subsequently published in April 2000.

A request for a Quotation and a call for participation in the OGC Web Mapping Testbed Phase II (WMT-2) were made in April 2000 to further test and expand the Web mapping specifications. The goal of WMT-2 was to rapidly develop interface specifications that lead to Standards-based Commercial-Off-The-Shelf (SCOTS) implementations of software that support use and exploitation of geospatial data and images over the WWW. WMT-2 builds upon the framework of specifications that have already been adopted or will soon be adopted by OGC. WMT-2 efforts will help refine existing OGC specifications and may create new specifications. Ultimately, this initiative will lead to standardized geospatial tools from multiple vendors that satisfy requirements for Web mapping (OGC Project Document 00-028). The OpenGIS WMS implementation interface specifications provide guidelines for current WMSs with the specifications of HTTP contents and URI communication syntax. The WMS specifications also lay out the major tasks of Internet map servers that can be applied in the architecture of distributed GIServices.

The major content of the OpenGIS WMS specifications focuses on how to describe a Web map server and map services with standardized URI syntaxes and semantics. A URI is a short string that identifies resources in the Web.

The format of URI strings indicates the syntax and semantics of formalized information for location and access to resources via the Internet.

The OpenGIS WMS specifications standardize the syntax and semantic contents of the URIs for WMSs and focus on the three major tasks. In general, "a standard web browser can ask a Map Server to do these things just by submitting requests in the form of Uniform Resource Locators. The content of such URIs depends on which of the three tasks is requested" (OGC, 2000, p. 9). The WMS implementation interface specification indicates that a WMS should be able to (OGC, 2000, p. 9).

- 1. produce a map (as a picture, as a series of graphical elements, or as a packaged set of geographic feature data),
- 2. answer basic queries about the content of the map, and
- 3. tell other programs what maps it can produce and which of those can be queried further.

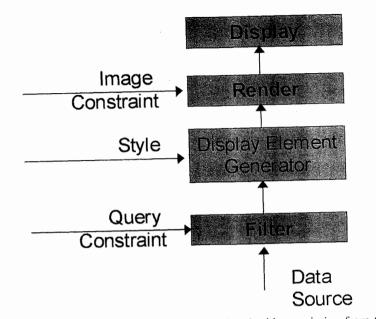
#### 4.5.2 WMS Architecture

Besides the specification of three major WMS tasks, the WMS specifications also identify four main processing stages in a WMS: filter service, display element generator, render service, and display service. The concept of four processing stages is derived from Cuthbert (1997), who describes geospatial data visualization from data to a map as a flow line with four processes, as shown in Figure 4.21:

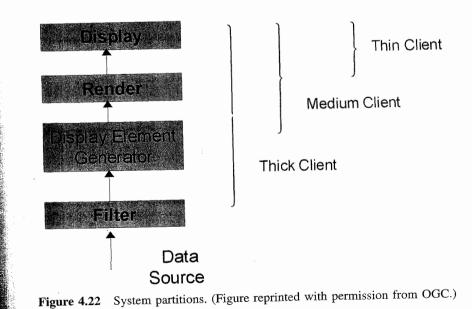
- 1. the *selection* of geospatial data to be displayed,
- 2. the generation of display elements from the selected geospatial data,
- 3. the rendering of display elements into a rendered map, and
- 4. the display of the rendered map to the user.

These four processes can be considered as service components. Each service component becomes a client of the other component, and each has interfaces that can be invoked by clients of that service. Depending on the location of these service components on the network, the whole process can be described as systems with "thin," "medium," and "thick" clients (Figure 4.22).

- If only the *rendered map* is carried over the Internet to the client, it would be a thin-client system with virtually no client-side capabilities. A typical example of this is displaying rendered maps as GIF files.
- If the *display elements* are carried **over** the Internet to the Web browser, it would be a medium-client system, which allows for a limited client-side processing, such as panning and zooming and selection.







• If the *geospatial data and display element generator service* is carried over the Internet to the Web client; it would be a thick-client system with unlimited client-side capabilities.

OGC later abandoned the use of "thin," "medium," and "thick" client in its WMS interface implementation specification because of some very imprecise definitions of "thin" and "thick" client used in the marketing literature. Instead, OGC uses the kind of information presented at the Web client to categorize the Web mapping services. This led to the three "cases": namely the "picture case," the "graphic element case," and the "data or feature case" (OGC, 2000).

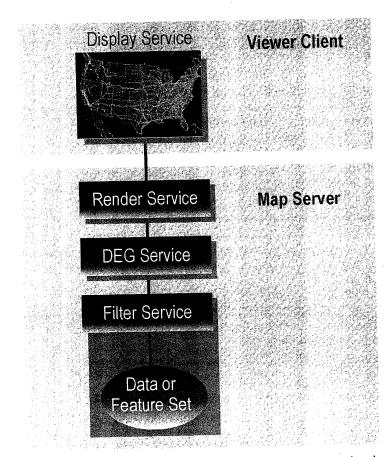
In the picture case, what travels across the Internet to the Web browser in response to a client's request is essentially a picture of a map in such format as a GIF, JPEG, or PNG. The map image was constructed by a map server and was transported to the Web client.

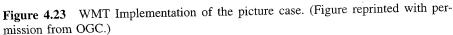
In the graphic element case, what travels to the Web client is a packaged set of individual elements, typically already in a projected reference system and with already defined symbolization for geographic features. Some graphic element formats include Scalable Vector Graphics (SVG) and Web Computer Graphics Metafile (CGM). For example, a freeway might be a thick red polyline, a lake could be a blue polygon, and so on. Some of the graphic elements could themselves be pictures like a bitmap or predrawn fragment of a map, so the graphic element case may also include the picture case as a subset.

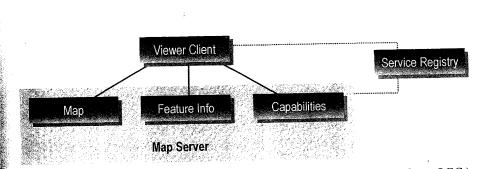
Finally, the data or feature case provides the ability to send geographic feature data from the server to the client. These feature data can be processed and manipulated directly on the Web client using display element generators and map-rendering tools. XML was tested in WMT Phase I to encode OpenGIS simple features, which resulted in an OGC specification of GML, which can be used to transport data from the server to the client.

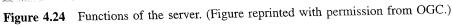
**4.5.2.1 WMS Specifications for the Picture Case** The WMS implementation specification covers primarily the picture case. As mentioned before, in the picture case, only pictures of maps are presented at the Web client; all other map rendering and data selections are conducted in the server, as shown in Figure 4.23. To simplify things, we can consider the server as one unit and focus on the functions of the server, as shown in Figure 4.24.

There are three functions that a map server could or should have in order to answer the user's request. First, the map server should be able to provide users with maps at the Web browser. Second, it needs to (may not have to) provide users with information about the maps, including information about the specific areas of the map and specific layers of the map. Lastly, the map server should be able to provide information about what interfaces a map server supports and what map layers it can serve. These functions are supported by three WMS interfaces: map interface, feature information interface,









and capabilities interface. Individually, they are sometimes informally referred to as GetMap, GetFeatureInfo, and GetCapabilities.

Map Request (GetMap) Interfaces The design of map request interfaces focuses on the display and production of Web-based map services: "To produce a map, the URI parameters indicate which portion of the Earth is to be mapped, the coordinate system to be used, the type(s) of information to be shown, the desired output format, and perhaps the output size, rendering style, or other parameters" (OGC, 2000, p. 9). The parameters of map request interfaces include the map layers, picture format, picture size, background color, and so on. Table 4.1 illustrates the parameters used in the map request interfaces.

Feature Request (GetFeature) Interfaces The feature request interfaces identify the request mechanisms for map contents and feature attributes.

#### TABLE 4.1 Map Request Interfaces

URL Component	Description
http://server_address/path/script?	URL prefix of server
	(Section 6.2.5.1.1)
WMTVER=1.0.0	Request version, required
	(Section 6.2.5.1.2)
REQUEST=map	Request name, required
LAYERS=layer_list	Comma-separated list of one or more map
	layers, required
STYLES=style_list	Comma-separated list of one rendering
	. style per requested layer, required
SRS=srs_identifier	Spatial reference system (SRS), required
BBOX=xmin,ymin,xmax,ymax	Bounding box corners (lower left, upper
	right) in SRS units, required
WIDTH=output_width	Width in pixels of map picture, required
HEIGHT=output_height	Height in pixels of map picture, required
FORMAT=output_format	Output format of map, required
<b>FRANSPARENT</b> =true_or_false	" <b>TRUE</b> " "FALSE": If TRUE, then the
	<b>background</b> color of the picture is to be
	made transparent if the image format
	supports transparency; optional;
	default=FALSE
BGCOLOR=color_value	A hexadecimal red-green-blue color value
	(Oxirggbb) for the background color;
	• optional; default=0xFFFFFF
EXCEPTIONS=exception_format	The format in which exceptins are to be
PPP	reported by the map server; optional;
	default=0xFFFFFF
Vendor-specific parameters	Section 6.2.5.1.5
-r parameters	Scrupi 0.2.3.1.3

To query the content of the map features, the URI parameters indicate what map (layer) is being queried and which location on the map is of interest (X, Y coordinates). Table 4.2 indicates the elements of feature request interfaces.

Capabilities Request (GetCapabilities) Interfaces The capabilities request interfaces are used to provide extensive map services, such as catalog services or metadata queries, in addition to the basic map display and attribute query (Table 4.3). For example, to ask a map server about its holdings, the URI parameters can be included in the capabilities requests, such as "Database=Colorado+California." However, current OpenGIS WMS specifications do not specify the exact contents of the GetCapabilities interfaces. The WMS specifications only suggest the possible use of GetCapabilities interfaces and leave the detailed design of the interfaces and contents to software vendors with their vendor-specific parameters.

Figure 4.24 introduces another element, the service registry. It is a component that delivers information about available services of different map servers to any client. The service registry identifies services through a search of metadata across map servers by invoking the capabilities request. The service registry also provides interface for publishing service descriptions to a publisher client. In WMT Phase I, a service registry was constructed using the OpenGIS catalog services specification (OGC, 1999).

#### Description URL Component URL prefix of server http://server\_address/path/script? (Section 6.2.5.1.1) Request version; required WMTVER=1.0.0 (Section 6.2.5.1.2) Request name; copy of map request **REOUEST**=feature\_info (map parameters that generated the map for request copywhich information is desired (Section 6.2.8.2) Comma-separated list of one or more **OUERY\_LAYERS**=layer\_list layers to be queried Return format of feature information; INFO\_FORMAT=output\_format optional; default=MIME How many features to return FEATURE\_COUNT=number information about; optional; default=1 X coordinate in pixels of feature X=pixel\_column (measured from upper left corner=0) Y coordinate in pixels of feature Y=pixel\_row (measured from upper left corner=0) Vendor-specific parameters (section 6.2.5.1.5)

Source: OGC, 2000, p. 30, with permission from OGC.

 TABLE 4.2
 Feature Request Interfaces

#### TABLE 4.3 Capabilities Request Interfaces

URL Component		Description
http://server_address/path/script?		URL prefix of server
	:	(Section 6.2.5.1.1)
WMTVER=1.0.0		Request version;
	•	required
<b>DECOURCE</b>		(Section 6.2.5.1.2)
<b>REQUEST=capabilities</b>		Request name;
		required
Vendor-specific parameters		Section 6.2.5.1.5

Source: OGC, 2000, p. 22, with permission from OGC.

In a distributed Internet environment there are many map servers across the Internet. Therefore, a "cascading map server" should be used to aggregate the capabilities of the individual map servers into one logical "place." The cascading map server can function as both a client and a server. It is a client to access many other map servers, while it is a map server to other Web browser clients. Furthermore, a cascading map server can also perform additional services. For example, a cascading map server can convert many different graphics formats (e.g., PNG, JPEG) into GIF format. This would allow any viewer clients to display any output from different map servers. Similarly, a cascading map server might perform coordinate transformations on behalf of other servers.

4.5.2.2 WMS Specifications for the Graphic Element Case The graphic element case is the medium-client model that the client-side machines can provide both display and render services (Figure 4.25). The servers will process the geodata from the GIS databases and generate well-defined geodata objects with associated symbols and colors. AutoDesk's MapGuide is one example of the graphic element case. The advantage of the graphic element case is that the combination of render and display services can allow more interactive user manipulation of map features, such as the vector-based highlights/selections and dynamic graphic display elements. In the graphic element case, map users can create a new graphic element on the client side and send it back to the server for updating (such as the map notes function in MapGuide). The response time and display performance is faster and better than the picture case, especially in the zoom-in, zoom-out types of display functions. However, map users have to download specialized Web plug-ins, ActiveX controls, or Java applets besides the regular Web browsers in order to see the graphic elements. The implementation of the WMS is more difficult than the picture cases because the graphic element case needs to modify the functions of the HTTP servers and add a middleware on the server, such as a Java servlet engine or CGI, for communication between Web servers, GIS databases, and client-side viewers.

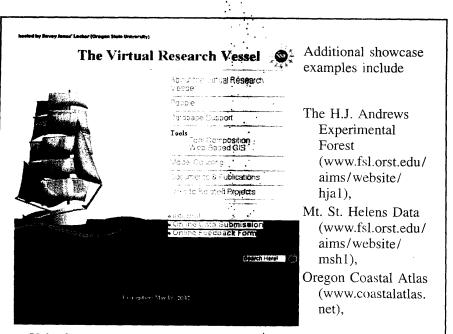
## Internet GIS Showcase: Internet GIS at Oregon State University

An exciting development with regard to Internet GIS is its use as a portal for GIS functionality as well as data distribution (e.g., Xue et al., 2002). Oregon State University (OrSt) researchers involved in developing Internet GIS applications along these lines include faculty and graduate students from the Departments of Geosciences, Entymology, Forest Science, Soil and Crop Science, Computer Science, and Bioengineering, the Marine Resource Management Program, and the Northwest Alliance for Computational Science and Engineering. Designs and applications range from the simple presentation of data via Web mapping to more complex signal analysis, real-time scientific collaboration and the incorporation of environmental models and decision support. For example, entymologists at the OrSt Integrated Plant Protection Center are developing a "public access GIS" using GRASSLinks (ippc2.orst.edu/glinks/) in order to integrate weather and climate data with soils, topography, insect distribution, and other environmental layers for the purposes of phenological (biology related to climate), population, and disease risk modeling. Their applications also include a real-time, multiresolution climate mapping expert system as well as the serving of national ecoregion GIS layers and maps in collaboration with EPA.

CRASSLARE as Word War We service to a prophysic latituding open (CR); official problem come to support of the service problem come to support to support of the service problem come to support of the service problem come to support of the service problem come to support to support of the service problem c

While the infrastructure for ready access to data and the resulting maps via Internet GIS (i.e., linking data to data) is desired and needed, OrSt researchers, in collaboration with geologists and computer scientists at the University of Oregon and computer scientists at

the Evergreen State College (Olympia, Washington), are building the Virtual Research Vessel (oregonstate.edu/dept/vrv), an experimental linkage of Internet GIS to additional database support, tool composition, and numerical models. Major objectives include the refinement of numerical simulations, better exploration of relationships between observations of the seafloor made with various instruments and vehicles, and the quantitative evaluation of scientific hypotheses. In this regard, Internet GIS is viewed as a preliminary step toward widespread data access rather than as a final solution. Better support for analysis, modeling, and decision support within or connected to Internet GIS should move users beyond the "data-to-data" mode toward "data-to-models" and "data-



Yolo County Sediment and Soil Analyses (yolo.een.orst.edu/ yolo.net), and

Virtual Oregon (virtual-oregon.nacse.org).

In addition, the following Internet GIS research questions are being considered:

- How should data models and data structures for Internet GIS differ from conventional GIS data structures?
- Are there standard metrics for GIS functionality that should be developed for specific application domains?
- What are the appropriate measures of performance for Internet GIS?
- What are the primary barriers to the usability of most Internet GIS sites? Usability engineering techniques are being investigated and deployed and multilevel Web-to-database interfaces are being developed for Internet GIS to enable customized access for meeting the needs of very different user groups.
- To what end should Internet GIS be developed? It is normally best used with broadband access in order to get satisfactory results. Yet, according to McGovern (2001), who cited statistics from NetValue, only 11% of American, 5% of German, 4% of French, and 3% of British households had such access in 2001.
- Given the predominance of Windows-based systems, what is the future of UNIX/Linux open-source GIS for the Web, and how can these toolkits be exploited and proliferated?
- Should Internet GIS be an "add-on" to standard Web browsers or cross-platform languages such as Java?

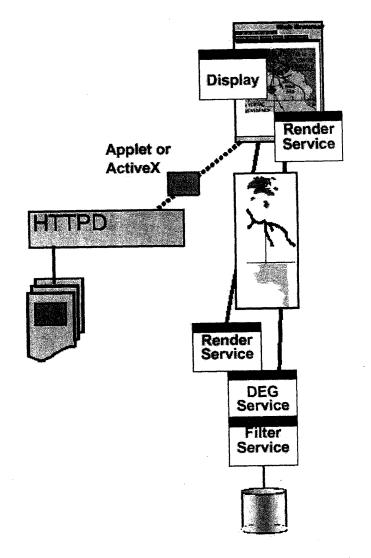


Figure 4.25 Graphic element case. (Figure reprinted with permission from OGC, 2000, p. 15.)

**4.5.2.3 WMS Specifications for the Data Case** The data (feature) case is the thick-client architecture where the client-side machines can perform display, render, and Display Element Generation (DEG) services (Figure 4.26). The servers will only be responsible for communicating GIS databases and the client-side map viewers. The communication between the client-side map viewers and servers may use XML or GML to specify the geodata elements and map display properties. All the map tasks, such as projections and

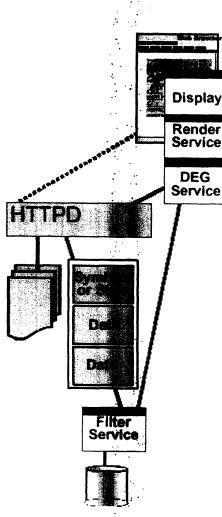


Figure 4.26 Data case. (Figure reprinted with permission from OGC, 2000, p. 15.)

symbol selections, will be performed locally in a viewer. ESRI's ArcIMS feature service is one of the examples of the data or feature case. The advantage of the data case is that it allows users to have the most freedom in manipulating geographic data items. Users can change the symbols and colors of map features locally without sending requests to the servers. Also, users can display both the Web-based map features with the data layers from local machines in local hard drives. Since the client viewer already has all the display capabilities, map users may use the client viewer to perform basic GIS operations, such as buffering and overlay operations. However, the map users may need to pay client-side software license fees in the data case arrangement because such powerful client-side map browsers can be used as regular GIS software packages. In general, the three case examples have their own advantages and disadvantages. Currently, the picture case is the most popular framework adopted by the GIS industry. However, the picture case only provides limited map display functions and less user interactions. Along with the progress of Web mapping and information technologies, the data case and the graphic element case may become more popular than the picture case in the future. The WMS implementation interface specifications (version 1.0) only focus on the picture case (thin clients) with the standardization of URI syntax and semantic contents. The next version of WMS may focus on the graphic element cases and the data cases with the adoption of GML applications.

The three cases in the WMS specifications demonstrate that different types of GIServices may need to adopt different types of software architecture. However, the software models proposed in the OpenGIS WMS specifications do not provide an approach for dynamically changing the architecture of Web map services. For example, the software framework in the picture case will not be able to upgrade to the graphic element case or data case if client-side map users ask for a higher level of map services or want to change their map applications. The ad hoc WMS specifications do not provide a flexible mechanism for migrating a software framework from one case to another.

One possible solution for providing an upgradable software framework for WMSs is to adopt the dynamic GIServices architecture proposed in this book. By adopting the dynamic framework proposed in Chapter 5, the WMS software framework can be easily upgraded from the picture case to the graphic element case or the data case by relocating the map service elements. Figure 4.27 illustrates such a dynamic architecture for Web map services, where each service element can be freely moved or relocated among client or server-side machines. This dynamic architecture will be able to provide a flexible software architecture for Web map services.

Figure 4.27 illustrates that different map users can access the same server that provides map services in either the picture case (scenario A) or the graphic element case (scenario B). For example, scenario A could be that a map user wants to display road maps in Boulder, Colorado, and the client machine only requires display services (the picture case is the best choice). Scenario B could be that a map user wants to find out the top 10 cities in the United States with the highest population growth rate. This scenario may require advanced map query capabilities and more flexible map display functions. Thus, the client machine could dynamically download a render service element from a server to the client machine (the graphic element case). By introducing the GIS component container and the dynamic GIServices architecture, map users can download different types of map service components based on their needs from servers to clients or vise versa. The dynamic change of the architecture will provide more flexible, upgradable, and user-oriented Web map services for users.

The WMS interface specifications deal primarily with the interfaces on the Web server. It is part of a more general Web mapping architecture that involves many distributed map servers on the Internet.

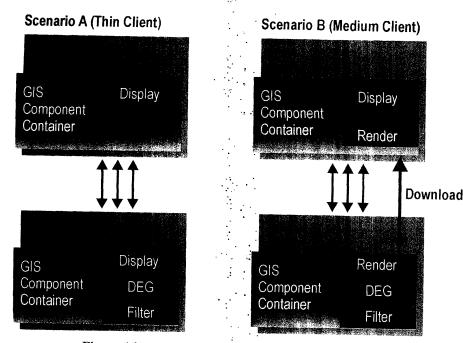


Figure 4.27 Dynamic architecture for Web map services.

Web mapping is still a preliminary GIS program that has very limited functions. To turn the Web mapping program into a truly distributed GIS, we need to rely on the more advanced distributed-component technology and its standards. In the next two chapters, we will introduce these more advanced distributed-component technologies and their emerging standards.

WEB RESOURCES	
Descriptions	URL
Xerox Map Viewer GRASSLinks ADL DLI Phase I DLI Phase II OGC ISO/TC 211 ESRI ArcIMS Java viewer GeoMedia ActiveX viewer	http://mapweb.parc.xerox.com/map (it is offline now) http://www.regis.berkeley.edu/grasslinks http://www.alexandria.ucsb.edu http://www.dli2.nsf.gov/dlione http://www.dli2.nsf.gov http://www.opengis.org http://www.isotc211.org http://www.esri.com/software/internetmaps/index.html http://www.intergraph.com/gis/gmwm

REFERENCES	205
------------	-----

Visa ATM locator MapQuest MapBlast! http://www.visa.com http://www.mapquest.com http://www.mapblast.com

#### REFERENCES

- Buehler, K., and McKee, L. (Eds.) (1996). The OpenGIS® Guide: Introduction to Interoperable Geoprocessing. Wayland, Massachusetts: Open GIS Consortium.
- Buehler, K., and McKee, L. (Eds.). (1998). The OpenGIS<sup>®</sup> Guide: Introduction to Interoperable Geoprocessing and the OpenGIS Specification, 3rd ed. Wayland, Massachusetts: Open GIS Consortium. URL: http://www.opengis.org/techno/ guide.htm, May 11, 2000.
- Buttenfield, B. P. (1998). Looking Forward: Geographic Information Services and Libraries in the Future. *Cartography and Geographic Information Systems*, 25(3), pp. 161–171.
- Buttenfield, B. P., and Goodchild, M. F. (1996). The Alexandria Digital Library Project: Distributed Library Services for Spatially Referenced Data. In *Proceedings of GIS/LIS'96*, Denver, Colorado. Bethesda, Maryland: American Society for Photogrammetry and Remote Sensing, pp. 76–84.
- Chappell, D. (1996). Understanding ActiveX and OLE. Redmond, Washington: Microsoft Press.
- Cook, S., and Daniels, J. (1994). Designing Object Systems: Object-Oriented Modeling with Syntropy. Englewood Cliffs, New Jersey: Prentice-Hall.
- Cuthbert, A. (1997). User Interaction with Geospatial Data. OpenGIS Project Document 98-060. Wayland, Massachusetts: Open GIS Consortium.
- Doyle, A. (1977). WWW Mapping Framework. OpenGIS Project Document 97-007. Wayland, Massachusetts: Open GIS Consortium.
- Gardels, K. (1996). The Open GIS Approach to Distributed Geodata and Geoprocessing. In *Proceedings of the Third International Conference on Integrating GIS and Environmental Modeling*, Santa Fe, New Mexico, National Center for Geographic Information and Analysis (NCGIA), CD-ROM.
- Gardels, K. (1998). A Web Mapping Scenario. OpenGIS Project Document 98-068. Wayland, Massachusetts: Open GIS Consortium.
- Goodchild, M. F. (1995). Alexandria Digital Library: Report on a Workshop on Metadata, Santa Barbara, California. URL: http://alexandria.sdc.ucsb.edu/publicdocuments/metadata/metadata\_ws.html, May 11, 2000.
- Huse, S. M. (1995). GRASSLinks: A New Model for Spatial Information Access in Environmental Planning. Unpublished Ph.D. dissertation, University of California at Berkeley, Department of Landscape Architecture, Berkeley, California.
- ISO/TC 211 Chairman. (1998). Draft Agreement between Open GIS Consortium, Inc. and ISO/TC 211. ISO/TC 211-N563.
- McGovern, G., 2001. The technology productivity paradox, *New Thinking*, 6(42), http: //www.gerrymcgovern.com/nt/2001/nt\_2001\_10\_29\_productivity.htm. Accessed 22 July 2002.

- 206 TECHNOLOGY EVOLUTIONS OF WEB MAPPING
- National Science Foundation. (1994). NSF Announces Awards for Digital Libraries Research. NSF PR 94-52. NSF: Washington, DC.
- Open GIS Consortium (OGC) (1998). *The OpenGIS Abstract Specification*, Version 3. Wayland, Massachusetts: Open GIS Consortium. URL: http://www.opengis.org/techno/specs.htm, May 11, 2000.
- Open GIS Consortium (OGC) (1999). The OpenGIS Abstract Specification, Version 4.0. Wayland, Massachusetts: Open GIS Consortium, URL: http://www.opengis.org/ public/abstract/99-113.pdf.
- Open GIS Consortium (OGC) (2000). OpenGIS Web Map Server Interface Implementation Specification, Revision 1.0.0. Wayland, Massachusetts: Open GIS Consortium.
- Open GIS Consortium (OGC) (2001). *OpenGIS Web Map Server Interface Implementation Specification*, Revision 1.1.0. Wayland, Massachusetts: Open GIS Consortium. URL: http://www.opengis.org/techno/specs.htm, September 11, 2001.
- Orfali, R., Harkey, D., and Edwards, J. (1999). Client/Server Survival Guide, 3rd ed. New York: Wiley.
- Peng, Z.-R. (1999). An Assessment Framework of the Development Strategies of Internet GIS. *Environment and Planning B: Planning and Design*, Vol. 26(1), pp. 117–132.
- Plewe, B. (1997). GIS Online: Information Retrieval, Mapping, and the Internet. Santa Fe, New Mexico: OnWord Press.
- Putz, S. (1994). Interactive Information Services Using World Wide Web Hypertext. In *Proceedings of the First International Conference on the World-Wide Web*, Geneva, Switzerland: CERN (European Organization for Nuclear Research) URL: http://www94.web.cern.ch/WWW94/PrelimProcs.html, May 11, 2000.
- Reed, C. (2002). Prior Art and Invention Related to Web Mapping, Version 1. Unpublished manuscript.
- Rowley, J. (1998). Draft Business Case for the Harmonisation between ISO/TC 211 and Open GIS Consortium, Inc. Resolution 47. ISO/TC 211-N472.
- Shan, Y.-P., and Earle, R. H. (1998). Enterprise Computing with Objects: From Client/ Server Environments to the Internet. Reading, Massachusetts: Addison-Wesley.
- Tang, Q. (1997). Component Software and Internet GIS. In Proceedings of GIS/ LIS'97, Cincinnati, Ohio, pp. 131-135.
- Wright, D. J., O'Dea, E., Cushing, J. B., Cuny, J. E., and Toomey, D. R. (2003). Why Web GIS May Not Be Enough: A Case Study with the Virtual Research Vessel, *Marine Geodesy*, 26(1-2).
- Xue, Y., Cracknell, A. P., Guo, H. D., 2002. Telegeoprocessing: The Integration of Remote Sensing, Geographic Information System (GIS), Global Positioning System (GPS) and Telecommunication, Int. J. Remote Sensing, 23(9): 1851–1893.

## **CHAPTER 5**

# FRAMEWORK OF DISTRIBUTED GEOGRAPHIC INFORMATION SERVICES

Geographic information systems are evolving to support a new, network-based architecture. This architecture is multiparticipant, collaborative, and will allow organizations to openly share and directly use GIS information from many distributed sources at the same time. —Jack Dangermond (2001)

#### 5.1 INTRODUCTION

The previous chapter introduced the technology of interactive Web mapping, which provide only half of truly distributed GIS. Client-side applications such as Java applets and ActiveX controls and dynamic HTML are designed mainly for graphic display of maps rather than truly providing GIS operations and analysis. There is very limited functionality in existing Web mapping programs, which do not offer much interactivity and flexibility for complicated GIS modeling and processing.

In addition, the architecture models for static Web mapping and interactive Web mapping require a CGI or CGI-like middleware between the Web client and the GIS and application server. The middleware approach adds the overhead of interactions between the Web client and a GIS server. What if we have a direct communication between the GIS server and the client? This is the idea of a distributed GIS.

So what is distributed GIS? Distributed GIS refers to a distributed platform of accessing and processing geospatial data using distributed GIService components on the Internet. It relies on mobile client components or downloadable clients that communicate directly with objects and data on the server across