MAPEDITOR: AN EXTENSIBLE FRAMEWORK FOR THE DYNAMIC MANIPULATION OF SPATIAL DATA OBJECTS ON THE WWW

George Messaritakis[1], George Zaharioudakis[2], Spyros Lalis[3] and Catherine Houstis[4]

(3, 4): Institute of Computer Science, Foundation for Research and Technology, Hellas
Science and TEchnology Park of Crete
P.O.Box 1385, GR 711 10, Heraklion, Crete, Greece
and
(1, 2): Computer Science Department, University of Crete, Hellas

{messarit,gzaxar,lalis,houstis}@ics.forth.gr

Abstract: The Internet greatly influences geographic science and the traditional GIS paradigm. The WWW is being used to disseminate spatial data over a large and diverse audience, but desktop GISs are not designed for this use and cannot meet present needs. So, a WWW-based GIS appears as the ideal solution. In this paper, we present a framework for the exploitation of spatial data over the web. The framework is open, extensible and provides rich functionality. Data is managed through autonomous, intelligent objects that take over processing and visualization related to their data. Users decide at runtime which data to view and how to interact with them, dynamically creating active maps on which spatial information of all kinds can be overlaid and manipulated. Moreover, such map documents can be shared between a group of collaborating users in a straightforward way.

Keywords: WWW GIS, spatial objects, active maps, dynamic manipulation, open architecture.

1. Introduction

The Internet, the computer network spanning the globe, is having a profound effect on society. Every aspect of life is being influenced and information sharing is becoming increasingly easier and quicker worldwide, as more people gain access to the Internet. The rapid and continuing growth of the World Wide Web is equally affecting science and technology, especially in areas of interest to the public. One such area is geographic information science, where significant changes are taking place, because existing geographic information systems are ill suited for public use. Thus, the idea of a GIS over the Internet has emerged and gained the assent of the mapping community [1].

The concept of a web GIS dates back to the appearance of the first public browsers, at around 1994. Since then, researchers have been exploring the feasibility of delivering spatial data to clients through their browser [2] and using HTML to offer rudimentary GIS functionality [3]. Following the phenomenal web expansion, most GIS vendors introduced data publishing capabilities to their desktop GIS packages. Of course, these first attempts seem primitive by today's standards. Static raster images were the preferred solution and interaction with the user involved filling HTML forms. Soon afterwards, new products appeared both from vendors and from researchers, boasting dynamic mapping and intelligent data. Still, these efforts were limited by the drawbacks of the web protocols.

The web GIS endeavor took off with the advent of innovative technologies, notably Java [4], ActiveX [5] and CORBA [6]. As these mature and become stable and reliable, allowing for more flexibility and advanced functionality, the vision of a comprehensive web GIS comes closer to reality. Several browser plug-ins created with them appeared, acting as web-based front-ends to desktop GIS packages, offering some of their functionality to remote users.

2. Issues for WWW GIS Design

However, the WWW has drastically changed the way information is processed, often enough undermining the assumptions according to which systems were built in the past. For this reason, giving applications that have traditionally been used as standalone tools a 'naïve' web disguise is likely to have a poor result. To identify and cater for present and future needs one must first consider the changes occurring in both the scientific and business geoscience community by the revolution brought by the WWW. A few important issues are given in the following.

2.1. Data Access

In the early days of web publishing, data providers were scarce; owners in the government and corporate world expected no immediate benefits from publishing. Indeed, data availability was identified as a major barrier to web GIS expansion. Today, the opposite end has been reached, as providers are eager to share their data with the large audience of the web. Data over-abundance makes locating the right information for each problem a rigorous task.

On top of that, even after the proper data have been located and retrieved, access to them is still not straightforward: the multitude of available formats presents another problem. Datasets produced from different agencies will be in different storage formats, owing to different methodologies, purposes and background. Furthermore, organizations strive to avoid conversions if possible, for the sake of simplicity and the consistency of their software tools. Such practices are to be expected in a competitive market, but they can only complicate matters.

To help remedy this situation, standards in data formatting are being discussed by the geoscience community. The foremost authority in standardization is the Open GIS Consortium (OGC), which co-ordinates the design of an Open Geodata Interoperability Specification (OGIS) [7]. All major desktop GIS vendors, along with research institutions, government agencies and computer companies are OGC members, in an attempt to reach universal consensus and facilitate geodata interoperability. However, OGIS-compliant implementations are still rare, therefore, a web GIS should recognize the different data formats and database protocols.

2.2. Interface and Functionality

When the public demand for spatial information is considered, it is evident that publishing data on the web shows promise. Weather forecast sites, for instance, are one of the popular categories. When deciding on the interface and functionality of a web GIS it is important to keep in mind that casual and inexperienced users dominate the target audience [8]. Their needs should be thoroughly analyzed and catered for and their familiarity with the web paradigm of a unique interface for multiple services should be exploited.

However, the requirement of simple and uniform interfaces need not be a setback for available functionality. Indeed, it *must not* –or the system will degrade into a fancy visualization tool. The analysis operations and manipulation capabilities of the desktop GISs should be retained, automated where necessary or possible, but still supporting advanced processing. A high availability in supported operations will help users in achieving their goals, which are information comprehension and problem solving.

Of course, emphasis on casual users should not preclude support for the professionals' needs. Precision, quality and efficiency are the fundamental characteristics in this case, for both the interface and the functionality. These users, too, expect features similar to those they are familiar with, namely desktop GISs' –so ease of use ranks second to functionality. Ideally, the web GIS will expose different interfaces to users, depending on their background, while the underlying implementation remains unchanged.

2.3. Advanced Features

Catering for the expert users is also important for another reason: *they* can expect the most significant benefits from using a web GIS. Casual users will normally use one to analyze just a few datasets and perform simple queries. On the other hand, for the geoscience professionals the prospect of sharing data and collaborating with their peers is too important to overlook. More so, when concurrent map editing is a possibility. The GIS can allow remote users to simultaneously connect to the server and then coordinate real-time sharing of maps created by them at run-time. While these features are obviously sought after by the more advanced users, they are currently unavailable.

A more mundane, but still highly desirable feature is the persistence of user-defined maps. Spatial data usually take long to transfer and computations on them can be lengthy and complicated. Neither the data, nor the computation results can be reliably stored on the client machine; this could even be impossible, due to size or legal concerns. Besides, a web GIS should encourage users to share, in turn, *their* information; this is achieved, if the client can store and retrieve user maps on demand. The GIS would then be an Internet editor for custom maps. Therefore, data sharing and collaboration would be promoted and reference/history material could be created.

A related issue, which too is of paramount importance to professional desktop GIS users, is authentication and security support. It is not uncommon for technicians on the field, foremen and privileged users to want to access their organization's spatial database from a laptop or other mobile hardware. In addition, the organization would really like employees to utilize the same web GIS as casual users visiting their site, for easier support and maintenance –but without compromising consistency or safety. If user identity can be verified by the GIS, access to data can be controlled and permitted for authorized individuals. Extending this idea a little further, the providers can offer high quality and up-to-date spatial data to users willing to pay for them.

2.4. Openness and Extensibility

Traditional desktop GISs are closed and proprietary software applications. In the competitive market this is a reasonable practice: clearly, vendors don't want others to be able to upgrade or improve their products. By charging for updates and new versions, they can further finance research and development. However, the majority of a web GIS's audience is people that are unlikely to pay for it –because they will use it sporadically–, or don't have to keep up with every upgrade –because they aren't professionals. In addition, researchers will want to experiment with it, without the limitations of the desktop GISs' scripting languages. Similarly, data interoperability is a fundamental requirement for the web GIS, but because of the number of diverse data formats, its developers won't be able to create and maintain a comprehensive collection of tools for them. Besides, a format's creators are the best qualified to write software that can use it fully and correctly. Consequently, the GIS developers should provide enough specifications to allow third parties to build custom tools for their needs.

Still, an open architecture is not, by itself, significant, if the overall design does not consider extensibility. For example, the specifications may be available, but the system may require a complete redesign to work with new components. In this case, even such trivial extensions as the incorporation of different formats will be an awkward and frustrating task. In addition, the system will be a maintenance nightmare and become obsolete as new technologies emerge. Of course, extensibility is not a property that can be added to the system after it has been built. Instead, it is an essential characteristic that has to be present in every design phase –only then can custom components be seamlessly integrated afterwards. Therefore, a clear separation must exist between fixed and swappable modules and the former must be restricted to the absolutely essential. The system will be more extensible and adaptable, the more functions can be changed or added a posteriori.

3. Related Work

The existing web mapping systems can be roughly divided into two categories. First, those that work only on a certain format and present a predetermined collection of datasets. Such systems are, for example, the GeoConcept Internet Server [9] and the GEO-Data Explorer [10]. Systems in this category are too limited to be considered real GISs; they are not extensible and cannot use different datasets without modification. Therefore, we will not explore them further.

The second category contains the more advanced systems. Some of these are based on desktop GISs, while others are original software programs. Internet GeoStar [11] is an OGIS-compliant web GIS, which provides rich functionality and dynamic data selection, by connecting to spatial databases and retrieving data. It also allows users to print the maps on their printers. CommonGIS [12], an EU initiative, is an ambitious project for the creation of a web GIS that is open, object-oriented and distributed. It concentrates heavily on the casual user and aims to offer very straightforward data retrieval facilities. The goal is to automatically generate thematic maps, using a knowledge-based approach. The user simply selects the information they want to see and the appropriate visualization is created. Finally, DOOGIS [13] is a sophisticated web GIS, that allows the user to dynamically download intelligent objects, which perform some behavior or supply data. The objects reside on the GeoViewer [14] spatial data server.

While these systems are adequate for some needs, they do not address the issues discussed earlier. In the following, we present our approach. The next section describes the fundamental ideas behind the project and how they pertain to the above. A brief overview of the system's architecture is given in section 5. Finally, section 6 concludes and sets future possibilities.

4. The MapEditor Project

MapEditor is a framework for the dynamic manipulation of intelligent spatial objects over the Internet. This framework is open and designed to be highly extensible and modular. It is separated into two parts: the essential framework structure and the object modules. The framework acts as the operating environment for the objects, which are loaded by clients on demand. The objects can be built around and exploit a particular data format, or provide format-independent functionality. The system can be extended by simply plugging in custom objects.

4.1. Dynamic mapping

MapEditor differs from other similar systems in the way maps are created. The whole process is completely dynamic and decisions are taken by the users, when they actually use the program. No restrictions are imposed on them, neither predetermined map configurations exist. Instead, each user individually decides what data they want to operate on, how they prefer to visualize them and what calculations they will perform on them. This is all done by adding the appropriate objects to the map; these provide the required data and functionality. Because they are intelligent and not simply data wrappers, they make the map interactive and "live". Therefore, dynamic mapping in this context means both that users can create maps at runtime and that they can interact with them.

4.2. Customization and Extensibility

Separating the system into two parts allows it to be extremely configurable and extensible. The framework structure is largely fixed, but the objects have complete freedom of action, as long as they follow certain fundamental guidelines. These are not restrictive per se; they define how communication between the two parts is achieved. Writing spatial objects that work in MapEditor is neither difficult nor time consuming. Therefore, third parties can develop and provide their own, custom objects. Normally, these will recognize and exploit proprietary formats, support additional functionality, or operate on other objects (see 4.3). What they do is up to developers to decide –as is how they do it. Customization and extensibility are built around object independence and MapEditor's modularity. Unsupported data formats can be integrated; existing web systems can be accessed and their facilities used; optimized algorithms can be used in operations. All that is required is for someone to create the respective object. Even existing features can be improved, by either modifying the original objects, or writing new ones that do the same things in a different manner. All these objects will cooperate unhindered, as long as the framework guidelines are observed.

4.3. Object Hierarchies

The framework provides facilities that objects can use to discover the existence and properties of other objects on the client. Consequently, objects can be developed, that operate on other objects. These are meta-objects and are equally important as plain data objects. Normally, they will be used to offer advanced functionality and combine data from different objects in computations. Typical examples are the already available query and animation objects, which we present in section 5. Moreover, the meta-objects are spatial objects in every aspect and can be manipulated, in turn, by other meta-objects. Thus, complex hierarchies of objects can be built and dissimilar data merged and integrated. In addition, the users are presented with a single interface from which to manipulate the data, regardless of their internal structure, formatting or physical location on the Internet. The link between objects and meta-objects is achieved through a programming interface, to which objects can adhere if they wish to be visible to meta-objects. Therefore, existing meta-objects will work correctly on objects created after them.

4.4. Persistence

MapEditor enables users to store and retrieve their customized maps on demand. In this context, these maps are called projects; a project is the composition of objects loaded on the client in their current state, together with the client's state information. Users can store the active project and later resume their work on it. Of course, the framework does not store the actual objects, but instead a description of them. Each object, in turn, must provide information about its state; when the project is restored, the objects are initialized using this information. This unique MapEditor feature obviously has endless possibilities: users can keep their preferred map configurations, interesting scenarios, or archives of their work. They can create maps, store them and allow other users to edit them; viewpoints and insights can thus be shared.

A similar feature is the creation of raster images from the contents of the map. Users can save GIF images on the server and then retrieve them using their browsers. Therefore, it is possible to obtain hard copies of projects.

5. MapEditor Architecture

MapEditor features a client-server architecture: a Java applet that downloads to the client machine and the supporting server. The applet provides data visualization and analysis in a flexible and configurable workbench. The server co-operates with the applet and provides persistence and communication with remote computers.

5.1. The server

The server computer runs a Java implementation of a simple HTTP server, which also supports servlets [15], the Java equivalent to CGI scripts. Its author is Jef Poskanzer [16]. We

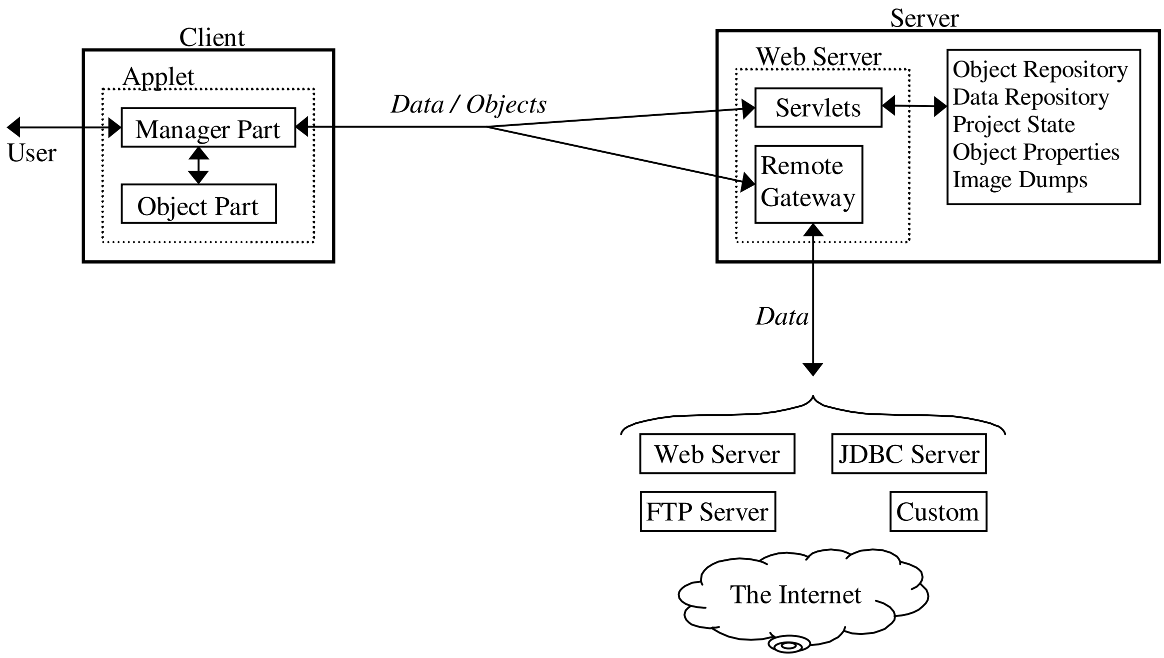

Figure 1.

extended it by adding eight servlets and a gateway to remote computers (Fig. 1). The servlets are invoked by the applet and carry out specific tasks, mainly object and data transfers and saving and loading of projects. The gateway is a link between the applet and remote network entities, such as databases and web or application servers. We must use the gateway to access remote servers, because the applet is not signed. The server machine is also used to provide storage. On it reside the object classes that are loaded on the applet and possible data associated with them. What these data are and how they are accessed is up to the object. In addition, the server holds the states of projects saved by users; state information consists of applet data and object data. User actions from the applet can change project information. Finally, when a raster image of a map is created, it is stored on the server.

5.2. The client

The client is a Java applet that downloads from the server and executes in the browser's Java Virtual Machine (JVM). It is divided into two parts, the manager and the object part. Initially, when the applet is first loaded, only the manager is present. The user can then fetch objects at runtime and create custom projects. The role of the manager and individual objects is briefly described below.

Object Manager.
This part is the applet's core and consists of three sections:

- Essentials: This section initializes the environment, controls the cooperation of objects and provides the fundamental operations. These are project management (save and load), object management (add and remove) and map management (pan and zoom). It is the link between the rest of the applet and the server. It also manages the list of objects for the active project and delegates requests to them.

- Visualization: This section contains the classes that implement the graphical user interface. They are responsible for keeping the drawing surface consistent and for translating geographic coordinates to screen pixels and vice-versa.
- Utilities: Several classes comprise this section. They are used in the manager part, but are also available to objects, for their convenience. They include a GIF encoder [17], a polygon clipping algorithm [18] and a few useful UI elements.

Implemented Wrapper Objects.
These objects are wrappers around the different spatial data formats. Some simply display the data, while others can also perform queries on them. Currently, the only industry standard format we support is ESRI's shapefile [19] –the other supported formats are publisher-specific.

- DatFile: This object reads latitude-longitude pairs in ASCII format from a file. These pairs define coastlines and country boundaries. The object simply draws them as lines on the map. The datasets were created using Rich Signell's Coastline Extractor [20].
- Shapefile: This is a wrapper for ESRI's shapefile data format and behaves like the vendor's ArcExplorer [21] program. It reads and draws vector maps (coastlines, city maps, etc). The attribute information that comes with the maps –in DBF [22] formatted files– can also be displayed and queried.
- SSH: Data for the SSH object are binary files that contain values for the height of the sea's surface, on a global scale. They are remotely sensed satellite data and are provided by the Physical Oceanography Distributed Active Archive Center (PO.DAAC), at the Jet Propulsion Laboratory [23]. The object plots the values, using a simple color-coding scheme.
- NCDC Animation: The National Climatic Data Center [24] provides global data that measure several climatic attributes, on a global scale. The published files contain monthly data for a full calendar year and are, therefore, ideal for creating animations. The object reads these files and animates over the monthly data. It can load files either from the server, or directly from NCDC's FTP site [25] –to achieve this it uses the server's remote gateway facility. It also allows the user to query the data by giving a pair of values and returns the places for which the attributes are within range.
- Dienst: This object connects to a Dienst [26] server that can perform spatial queries [27] and retrieve documents that relate to geographic areas. It is still under development; the final version will show the documents on the map and allow users to view them in their browsers.
- MySQL: This is a front-end to a JDBC[28]-compliant database that contains spatial data. The user can query the database and retrieve data. We currently use a MySQL [29] database, loaded with spatial data extracted from NCDC's files. The object acts similarly to those that handle simple data files, but also takes advantage of the structured organization of the data, achieving better performance. In an updated version, we will add several privilege levels to the database and the object will ask users for their passwords. This will demonstrate authentication support.
- Topologies: This simple object shows georeferenced GIF images. The images contain the topology of some American States.

Meta-Objects.

Meta-objects provide services based on other objects. Normally, because each object knows how to access its data and perform operations on them, meta-objects will delegate data-specific tasks to them and manipulate the results.

- Query: This is an object that coordinates the execution of queries on objects that implement the relevant interface. Querying is a two-step process. First, the user performs queries in a series of objects; these queries are on object data and are handled by each individual object separately. Then, this object collects the results, combines them and displays the final result, as specified by the user. Currently, the queries can be combined with logical operations –the polygon clipping algorithm from the manager is used. When a Query object is saved, it stores links to the queried objects. When the project is restored, the user can see how the final result was produced and query these objects to get different results. Objects are expected to implement the query interface and support queries on their data.
- Animation: This one manages frames extracted from objects that support that option. The user creates the animation dynamically, by selecting which object will provide each frame, as well as the frame's content. Then, he sets the animation's parameters and the frames are displayed in a sequence. When the project is saved, this object can store the animation sequence –not the actual frames, but information about them– and recreate it when loaded.

5.3. Typical Usage.

To better illustrate how all the above are put into place, we present a typical scenario of the program's use.

At the beginning, the user visits MapEditor's webpage and the applet loads. He is prompted to input a user name and a project name and an initially empty map is shown. Then, the user adds a DatFile object, to display the world map and an NCDC object, selecting the Snow Coverage dataset. Following, he adds another NCDC object, for which he selects the Mean Rain Fraction dataset, directly from NCDC's FTP site (Fig. 2). Then, he adds a Query object, and specifies that the two NCDC objects will participate in this query. After querying them and merging the individual results, the final result is shown (Fig. 3).

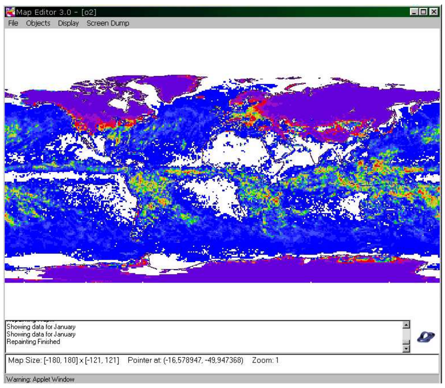

Figure 2.

At this point, the user decides to store the current project; MapEditor asks for a project name. A Shapefile object is then added, showing the US State boundaries. Finally, a Query object is added, linking the first Query and the Shapefile. The user queries the Shapefile object and combines this result with that of the first Query object (Fig. 4). Before exiting the applet, he saves the project, under a different name.

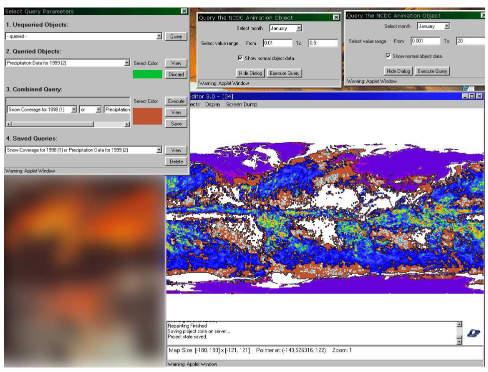

Figure 3.

This scenario showcases all of MapEditor's fundamental features: data and functionality specified dynamically through objects, direct user interaction with the map, creation of complex object hierarchies at runtime, persistence of projects and communication with remote network entities.

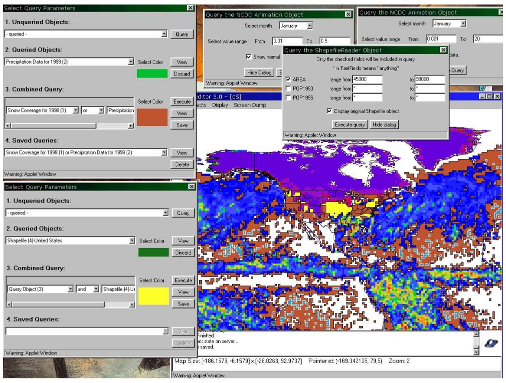

Figure 4.

6. Future Work

We can identify two possible directions for the future. First, the supported data types and functionality can be extended by providing additional objects. Some will recognize standard industry formats for spatial data, but others can be more interesting. For example, they can be front-ends to the various information retrieval and data mining systems currently available on the web. In addition, GIS operations can be provided through them, such as buffer zoning, layering, line-of-sight calculation, etc.

Alternatively, we can explore more innovative concepts, by integrating new features directly in the framework. We mentioned earlier that security and authentication are useful features and, indeed, objects can incorporate them individually. However, the framework could support them, too, and the objects just use the facilities. This would allow for additional services, such as controlling access to the objects themselves, or to saved projects and images. Similarly, the framework could have facilities for real-time sharing of maps between users – authentication would be useful here, as well. Finally, we could allow MapEditor servers to be distributed on the Web and cooperate with each other. Users could then retrieve objects not only from their applet's local server, but also from remote ones. So, data publishers could provide the objects to go with their data; after installing a MapEditor server, these objects would be available to every other server, instantly and transparently.

7. References

1) "GIS: Toy or Tool" discussion: http://www.gisnet.com/gis/notebook/ttsum.html
2) XEROX PARC pioneer web map: http://mapweb.parc.xerox.com/map
3) Susan Marie Huse. "GRASSLinks: A New Model for Spatial Information Access in Environmental Planning": http://www.regis.berkeley.edu/sue/phd/. University of California at Berkeley, 1995.
4) The Java programming language: http://java.sun.com/nav/whatis/
5) Microsoft's ActiveX controls: http://www.microsoft.com/com/tech/activex.asp?RLD=18
6) The Common Object Request Broker Architecture (CORBA): http://www.corba.org/
7) The Open Geodata Interoperability Specification: http://www.opengis.org/indexfr.htm
8) A 1997 GIS user survey: http://commongis.jrc.it/publications/perdigao2/img001.htm
9) GeoConcept Internet Server: http://194.183.214.212/gcis/java/
10) GEO-Data Explorer: http://geode.usgs.gov/
11) Internet GeoStar: http://www.rcgis.wtusm.edu.cn/wwwgis/
12) CommonGIS: http://commongis.jrc.it/

13) Distributed Object-Oriented GIS (DOOGIS): http://doogis.dis.anl.gov/
14) GeoViewer: http://www.dis.anl.gov/geoviewer
15) Java Servlets: http://java.sun.com/products/servlet/whitepaper.html
16) Jef Poskanzer's Acme web server: http://www.acme.com/java/software/Acme.Serve.Serve.html
17) Jef Poskanzer's Acme GIF Encoder: http://www.acme.com/java/software/Acme.JPM.Encoders.GifEncoder.html
18) Alan Murta's generic polygon clipper: http://www.cs.man.ac.uk/aig/staff/alan/software/
19) ESRI's shapefile format: http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf
20) Rich Signell's Coastline Extractor: http://crusty.er.usgs.gov/coast/getcoast.html
21) ESRI's ArcExplorer: http://www.esri.com/software/arcexplorer/index.html
22) DBF format: http://community.borland.com/article/0,1410,15838,00.html
23) JPL-PO.DAAC: http://podaac.jpl.nasa.gov/, The World Ocean Circulation Experiment FTP site: ftp://podaac.jpl.nasa.gov/pub/data_collections/woce_cdrom/topex/data/
24) The National Climatic Data Center: http://www.ncdc.noaa.gov/
25) Special Sensor Microwave/Imager (SSM/I) – Global Gridded Products: http://www.ncdc.noaa.gov/ol/satellite/ssmi/ssmiproducts.html, NCDC FTP site: ftp://orbit35i.nesdis.noaa.gov/pub/arad/ht/rferraro/ncdc
26) James R. Davis and Carl Lagoze. A protocol and server for a distributed digital technical report library: http://cs-tr.cs.cornell.edu/Dienst/UI/2.0/Describe/ncstrl.cornell/TR94-1418?abstract=lagoze. April 25, 1994.
27) A map interface for Dienst: http://members.xoom.com/messarit/projects/dienst.html
28) Java DataBase Connectivity (JDBC): http://java.sun.com/products/jdbc/index.html
29) The MySQL database: http://www.mysql.com/