Internet-Based Collaborative Geographic Information System

by

Nadine Sami Alameh

B.E., Computer and Communication Engineering
American University of Beirut (1994)

Submitted to the Department of Urban Studies and Planning
and
the Department of Civil and Environmental Engineering

in partial fulfillment of the requirements for the degrees of
Master in City Planning
and
Master of Science in Civil and Environmental Engineering

at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June 1997

© Nadine Alameh, 1997. All Rights Reserved.

The author hereby grants to MIT permission to reproduce and distribute publicly paper and electronic copies of this document in whole or in part, and to grants others the right to do so.

Author .....
Department of Urban Studies and Planning
May 9,1997

Certified by .....
Professor Joseph Ferreira
Department of Urban Studies and Planning
Thesis Supervisor

Certified by .....
Professor John Williams
Department of Civil And Environmental Engineering
Thesis Supervisor

Accepted by .....
Associate Professor Mark Schuster
Chair, MCP Committee
Department of Urban Studies and Planning

Accepted by .....
Professor Joseph Sussman
Chairman, Departmental Committee on Graduate Students
Department of Civil and Environmental Engineering

JUN 25 1997

Internet-Based Collaborative Geographic Information System

by

Nadine Alameh

Submitted to the Department of Urban Studies and Planning and the Department of Civil and Environmental Engineering on May 9, 1997, in partial fulfillment of the requirements for the degrees of Master of City Planning and Master of Science in Civil and Environmental Engineering

Abstract

The World-Wide Web (WWW) has increasingly been recognized as a cost-effective and reliable medium for information dissemination. Recently, with the development of tools such as Java and ActiveX, it has become possible to use the Web to develop collaborative applications for supporting geographically dispersed parties. For such collaborative types of applications, the Web is considered to be a suitable infrastructure because of properties such as platform independence, ease of accessibility, relative cost-effectiveness and easy and familiar browser interface.

In this thesis, we propose a system that applies and extends the concept of Internet-Based collaborative systems to the field of Geographic Information Systems (GIS).While other commercial white-board types of applications require the use of graphic/raster format maps, we want to preserve the vector-based nature of GIS layers. The proposed system uses a set of tools for porting the vector data into browser readable format to provide basic mapping and spatial functionalities. It also allows users to annotate their own maps , in real time, while sharing them with their working group.

Thesis Supervisor: Joseph Ferreira
Title: Professor of Urban Studies and Operations Research

Thesis Supervisor: John Williams
Title: Professor of Civil and Environmental Engineering

Acknowledgments

I would like to thank Professor J. Ferreira for his support and guidance throughout this study. I am particularly grateful to him for always expanding my research horizons and for constantly encouraging me to pursue my interests and to apply them in the context of planning. I would also like to thank Professor J. Williams for his faith in me and for his help throughout my graduate studies.

I am extremely thankful to the PSS professors, students and staff for a friendly and motivating research environment. In particular, I would like to thank professor Shiffer and professor Shen for their endless encouragement and support. I am also thankful to the MIT Intelligent Transportation Systems Lab group, especially Mr. Hotz and Mr. Welch for always being very close friends.

I would like to thank my friends at MIT for making my stay here a very enjoyable and fruitful experience. In particular, I would like to thank the Lebanese Club gang , especially Fadi, for creating such a home-like environment. I would also like to thank Mazen in Lebanon for his support and motivating remarks.

I can never thank enough my husband, Hisham, for being so patient with me, especially during the last semester. He has been my best friend all along. I thank him for his faith in me and for making me believe anything is possible. He taught me the meaning of love and optimism and for that I dearly thank him.

Most of all, I thank my family for giving me unconditional support beyond any measure. I especially thank my mother for sacrificing her whole life to make my dreams come true. I thank my sister Rola and brother Rani for believing in me and for being there for me all along.

Finally I would like to dedicate this thesis to my brother Rani. I sincerely hope I can make his dreams come true one day.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

The Internet has been increasingly recognized as a cost-effective, accessible and reliable medium for information communication and sharing. There has been a recent explosion in the types of data that are exchanged over the Internet. More specifically in the field of Geographic Information Systems (GIS), development tools, such as Java and ActiveX, are being used to address the growing need for publishing and interacting with dynamic maps. The availability and ease of access to such tools has allowed the Internet to resume its original role as a networking infrastructure for collaborative applications [28].

In this thesis, we experiment with the Internet as a collaboration medium and develop a set of tools in order to simulate an Internet-based collaborative Geographic Information System (GIS) laboratory. A Prototype Internet-based Collaborative Geographic Information System (PICGIS) is designed to provide basic spatial functionalities and act as a forum for geographic data sharing. In a typical setting, the operations performed by one of the users are automatically propagated to other logged-in users, providing a convenient alternative to having all users located in a single room, working over a single computer terminal.

The value of this system, as opposed to other simple graphical white-board applications, stems from its unique ability to deal with raw GIS data. In other words, with such a

system, there is no need for an extra layer of data conversion. Moreover, the system can be used to add and modify current layers, in real time, in addition to possibly printing or saving them for future reference.

1.2 Problem Definition

Reading, publishing and interacting with spatial data on the Internet has been a recent form of competition among several interested software parties. The goal is to build simple and familiar tools that allow the users to interact with available maps, by means of easy-to-use, friendly interfaces with the option of spatial data queries.

However, current web-based GIS applications lack the feature of collaboration. By collaboration, we envision a system that facilitates spatial information presentation and sharing. At today's GIS technology level, both novice and professional users lack a system where two or more parties are able to simultaneously manipulate a map without being in the same physical place.

Current attempts to solve the problem of spatial collaboration (such as PictureTel) often consist of expensive, proprietary and platform dependent software. By virtue of being Internet-based, our proposed prototype (PICGIS) solves this problem by using a system architecture that does not require new software installation. Instead, PICGIS can be downloaded through any commercial browser and used on any platform.

It might be argued that spatial collaboration is indeed currently achievable through the numerous web-based white-board applications. However, these applications are limited in several respects, including the data types they support, the functionality they offer, and the purposes they serve. For instance, maps are often required to be in a graphic/raster format (such as gif, tif, or jpg). Raw GIS data is however vector-based, requiring GIS users to go through an irreversible intermediate data conversion process, in order to use such applica-

tions. Not only does this approach undermine the use of the underlying map databases, it is also lacking in terms of some basic spatial analysis functionalities.

Our prototype, PICGIS, is designed to solve the presented problems by providing a "meeting place" on the Internet, where a dispersed group of people can communicate their ideas in both textual and raw spatial format.

1.3 System Audience and Potential Uses

The identification of the audience targeted by PICGIS, and its potential applications is critical for determining the prototype's architecture and supporting functionalities. This identification process also serves to highlight some aspects of GIS that can be enhanced by an Internet-based collaborative system.

The reader should also note that these applications are also valid in the context of the recently emerging field of Intranets, which is expected to compete with Internet usage before the end of the century [20]. Intranets are used for delivering internal applications over corporate Local Area Networks. Because Intranets run on the same open TCP/IP networks and inherit the Internet's standard protocols and technologies, they can use the same types of servers and browsers that are used for the World Wide Web.

1.3.1 Educational Setting

Increasingly, GIS is being introduced in many schools' planning and geography curricula. Unfortunately, current teaching methodology limits the students to one option, namely following the teacher's instructions, usually projected on a big screen. This type of setting restricts the student to doing one task at a time, usually that of watching the sequence of actions performed by the professor and maybe trying them later on his/her own.

The learning experience of the student in such a context can be enhanced by the introduction of a collaborative system, that is easily accessible and specifically designed to be used in the context of GIS. Such a system will act as a liaison between the student and the teacher, increasing the information flow between the two, and hence improving the student's productivity and learning experience.

Due to the visual nature of GIS, both students and professors need effective tools for illustration and analysis purposes. For example, a simple utility that allows the student to view the professor's actions on his/her terminal gives the student the ability to capture the professor's explanations, and replay them later. This is especially useful when the student uses the system to compare his/her results with those obtained by the professor.

There are also other ways in which a collaborative system might enhance the student's educational experience. Imagine a situation where a student, while doing his/her GIS homework, encounters a certain conceptual or technical difficulty. A collaborative system will allow this student to easily communicate with his/her course assistant. Furthermore, if this collaborative system is easily accessible from any location, the teaching assistant, in this case, can setup his/her office hours practically anywhere. A convenient way to make the collaborative GIS system easily accessible is by designing it to be Internet-based, since the Internet is a widely accessible and established infrastructure.

In addition, there is a need for a system through which students can collaborate on GIS projects. GIS projects may require team work and the expertise of several parties. In such cases, a collaborative system can increase total productivity, save time, and allow individual students to work remotely. In fact, if such a system is Internet-based, these students are not then limited to their own expertise as they can work and consult with any expert on earth.

1.3.2 Training Setting

The needs mentioned earlier also apply to a training setting. In addition, training the company's employees is a costly process. Experts have to be flown over to administer the training course, and often expensive software has to be installed. An Internet-based collaborative system will reduce travel cost and the need to install new software. Furthermore, by being Internet-based and hence most likely browser-based, such a system will project a sense of familiarity regarding the user interface.

In addition, in a collaborative system, it is easy to switch the roles of the trainee and the trainer. This setting allows the trainee to show the instructor his/her work or his/her sequence of actions. Such features are currently missing from both GIS packages and computerized training courses.

1.3.3 Professional/Technical Setting

Since GIS is a tool that is used in diverse fields, GIS projects may require the expertise of several professionals, for whom it may be very hard to gather in a single place. Currently, the telephone along with some video-conferencing tools enables such dispersed experts to create virtual meetings to discuss their projects. Such collaboration tools however are completely detached from the actual GIS system. There is a need for a collaborative GIS system that integrates such collaboration tools with GIS functionalities, to enable these experts to jointly work on their GIS maps and annotations in real time. For example, fundamentally collaborative planning activities such as negotiations and Environmental Impacts Reviews (EIR) will be enhanced by the sharing of a "powerfully intuitive and visual dimension of a live map" [1]. Such an on-line collaboration scheme is becoming increasingly plausible with the advances in Internet technology, and the availability of Internet telephony.

This concept of expertise sharing can also be applied in the area of customer support. A system is needed to enhance the customer support process by providing simple and accessible GIS tools for both the customers and the GIS software company. A collaborative GIS system allows both parties to go through sequences of GIS operations on the screen, while exchanging the details of both the problem and the associated solution through a voice channel.

1.4 Objectives

In light of the audience needs outlined above, the goal of this thesis is to build PICGIS, a working prototype for the Internet-based collaborative GIS. As a first step, a basic set of tools needs to be developed to establish the reading, publishing and interactivity functions of the spatial data over the Internet. These tools provide the GIS functionalities that are to be embedded in every client. On top of the basic spatial features, collaboration functions need to be designed and built to simulate the collaboration environment of the system.

The general objectives can be further subdivided into more specific goals, including:

1. Identifying the system's architecture in view of its goals and desired functionality.
1. Identifying the basic spatial and GIS functionalities to be included.
2. Identifying the data type(s) to be supported by the system.
3. Adding the tools for viewing the GIS layers on the web.
4. Implementing the client to client forwarding of different transactions.
5. Testing the system with basic maps.
6. Comparing the system with other available products.

At a higher level, the system's objectives focus on providing the necessary tools for building an Internet-based GIS laboratory. In addition to the basic mapping and spatial features of the lab, it also facilitates collaboration between geographically dispersed groups. Such a system can be considered as the intersection of three research domains (Figure 1.1), namely

1. The Internet.
2. Geographic Information Systems.
3. Collaboration Software (GroupWare).

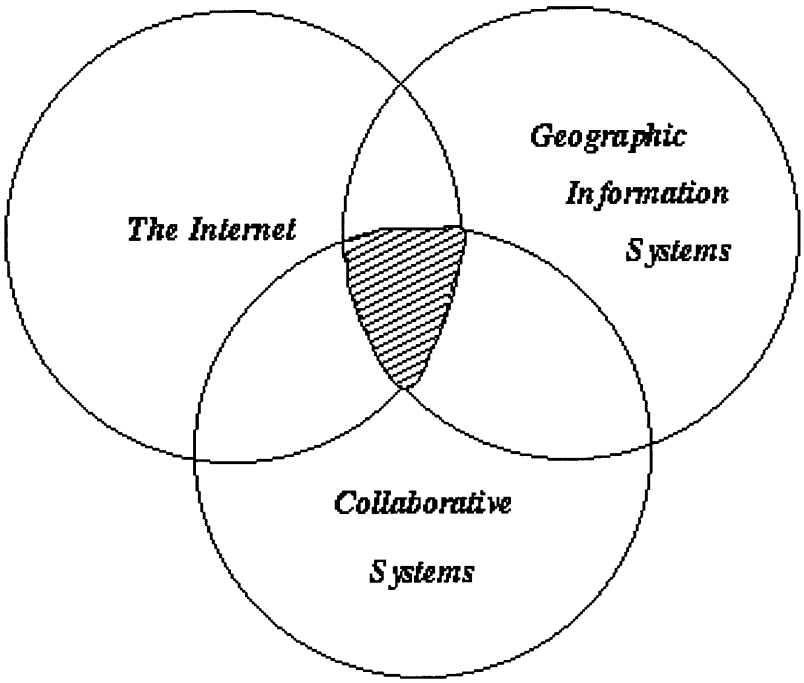

Figure 1.1: Viewing the System as the Intersection of Other Fields of Research.

1.5 Significance of the Research

The thesis essentially contributes to the three fields of research outlined in Figure 1.1. In fact, this work takes advantage of the recent developments in the area of Internet research in terms of using it for interaction and communication in PICGIS. The prototype is intended to present certain generic functionalities that can be further explored and incorporated in more advanced applications. This is made possible by following an Object Oriented design approach.

Moreover, the research takes Geographic Information Systems one step further by porting them to the Internet and adapting them to behave in a collaborative way. By devel-

oping web based tools, the Internet is expanded to support yet another type of data, namely the spatial (or GIS) data. For this purpose, the thesis will probably borrow some modeling concepts from recent products such as the Environmental Systems Research Institute (ESRI) MapObjects software and Spatial Database Engine (SDE) server.

The project suggests new on-line collaboration schemes that might change the way the targeted audience goes about solving and working on a GIS project. Interaction in the educational setting is expected to increase, as well as the productivity of both the student and the teacher. Negotiators, urban planners and professional groups can use the system with the help of another communication medium (e.g. Chat applications or Internet telephony) to discuss and negotiate certain projects, without physically being in the same place. The prototype, developed and described in this thesis, is the basis of more elaborate projects, that can lend to a significant decrease in the cost and time needed for training, consulting and system support.

The significance of the research can be further clarified by listing certain distinguishing characteristics of the system. In the following sections, we list the characteristics related to the collaboration feature of the project, its reliance on the Internet as an infrastructure and some aspects of its modular design.

1.5.1 Collaboration Feature

The collaborative nature of the system allows geographically distant workgroups to efficiently collaborate on GIS projects. The dispersed parties can use the system to contribute to the product in real time at a reduced cost. This feature can also be transferred to other applications of collaborative nature on organizations' Intranets.

1.5.2 The Internet as the Infrastructure

The Internet was selected as the medium for the deployment of this project because the resulting software would exploit several Internet features, including

- Platform independence/interoperability.
- Sparing of new and expensive software installation.
- Avoidance of the use of proprietary software.
- Wide availability of and user familiarity with commercial and inexpensive browsers.

1.5.3 Modular Design

The modular design of the project is reflected through

- A client/server architecture.
- An Object Oriented Design methodology.
- An attempt to use existing GIS data without conversion.
- A resolution to make the system easy-to-use and flexible enough to expand.

1.6 Methodology

The thesis process will start by a rigorous literature review, which is essential for a better understanding of the underlying problem, possible solutions and related issues. The literature research will focus on areas regarding collaboration, the Internet and web-based mapping. This stage also involves the process of exploring some existing spatially enabled sites, which will provide insights on features, limitations and drawbacks of current implementations.

The next step involves the identification of the final prototype components, following the general guidelines outlined by the project objectives (see Section 1.4). More specifically, the system's generic functionalities as well as its general behavior need to be determined. After identifying possible architectures and system setups, it becomes easier to select the approach that most efficiently achieves the prototype's goals.

The client/server relationship also has to be investigated by pointing out the functionalities and roles of each. The generic design and the selected system setup will automatically influence the Graphical User Interface (GUI), as well as the system's interaction with the underlying GIS databases.

The language of implementation of the prototype is Java, chosen because of its platform independence, interoperability, faster application loading times and lower client hardware requirements. In addition, Java is characterized by relatively small program sizes, wide support from major development and authoring tools vendors as well as increasing number of features released by SunSoft. Developing PICGIS in Java implies less dependence on the server, less network traffic and less server processing power [35].

In order to ensure that the final product meets the original specifications as well as the users' needs, it has to be tested and compared to other available products. Current limitations and future extensions have to be identified. The methodology is summarized in Figure 1.2.

1.7 Thesis Overview

A background study of literature is presented in Chapter 2. Chapter 3 presents the issues and alternatives pertaining to the design of PICGIS. Chapter 4 describes the system architecture of the proposed system while Chapter 5 illustrates its implementation details. Finally, Chapter 6 provides some concluding remarks and contributions as well as guidelines for future work.

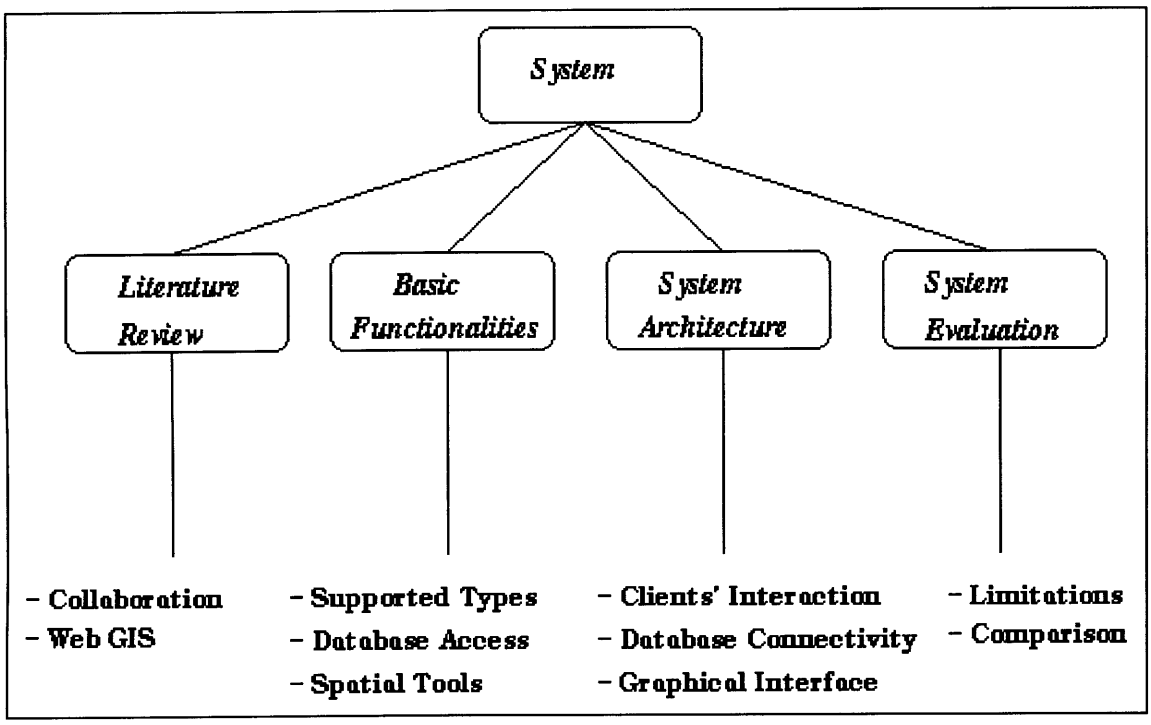

Figure 1.2: Basic Methodology Elements and Steps.

Chapter 2

Background

An overview of recent research trends, regarding the fundamental elements involved in this work, is essential in order to establish an overall framework for the thesis. In particular, this chapter will briefly describe current collaborative systems as well as current Geographic Information Systems. This section will also shed some light on the recent move of these applications to the Internet.

2.1 Collaborative Systems

The dispersion of both people and information creates a business challenge in organizations, to work efficiently and collaboratively under such circumstances. A prevalent solution to this problem is in the form of collaborative systems that are designed to overcome the spatial, temporal and organizational constraints. Collaborative systems consist of a class of computer technologies that enable sharing and exchange of information, as well as the coordination and collaboration between two or more distant people [14].

Nowadays, there exists a large number of such systems in the market, targeting collaboration in areas such as scientific research, software development, conference planning, political activism and creative writing [7]. These systems include functionalities such as shared databases, workflows, document management, electronic white-boards, computer

conferencing among many others. These features have been accentuated by the recent technological boom in the fields of multimedia, databases, networking and video conferencing [14]. Current players in this field include Lotus (with Lotus Notes), Microsoft (with Microsoft Exchange) and Oracle (with Oracle Workgroup Server).

2.1.1 Characteristics of Collaborative Systems

Collaborative systems were invented to fulfill the need for a system that serves dispersed individuals or team members. It is argued that this need is continuously growing as organizations tend to expand their traditional marketplace boundaries. In addition to achieving these goals, collaborative systems lower the costs of collaboration and team work.

The main driving attributes of such systems [14] are:

- Improving groups' performance and productivity.
- Creating global work teams.
- Facilitating the collaborative process.
- Reducing operational costs.

2.1.2 The Move to the Internet

As the World Wide Web widely expands and gains more popularity at all business levels, this geographically distributed network will increasingly provide an ideal platform for collaborative applications [6]. This trend is particularly driven by the wide availability of browsers on all platforms, the established standards for documents' interchange (such as html or java) as well as the relative ease and inexpensiveness of connecting to the Internet. This trend has also been accentuated by the rapid development of real time database access and update, as well as the support of animation and interactivity through the web.

Collaborative systems are taking advantage of these web features to create applications that include database connectivity and graphic interactiviy. Moreover, with the

advances in security, such collaboration initiatives can be guaranteed to be confidential for intra-organizational sharing and collaboration, even across continents.

In the next section, we give an overview of Geographic Information Systems and present some of their primary characteristics. This is essential for understanding the challenges faced in the process of porting this type of system to the web.

2.2 Geographic Information Systems

Defined in the early seventies, a Geographic Information System (GIS) represents a "computer based information system that enables capture, modeling, manipulation, retrieval, analysis, and presentation of geographically referenced data" [37]. In simpler terms, GIS can be described as a "computer system capable of holding and using data describing places on the earth's surface" [10]. The research field of GIS combines special interest computer science fields, encompassing the areas of databases, graphics, systems engineering and computational geometry [37].

Geographic Information Systems are used in a variety of applications, including high quality cartography, land use planning, natural resource management, environmental assessment, tax mapping, ecological research, emergency vehicle dispatching as well as business applications. In addition, GIS systems are used by demographers for market analysis, address matching and geocoding, as well as forecasting and planning [10].

The next few paragraphs introduce important aspects of a Geographic Information System, and describe typical problems addressed by such a system, its general data model, and its reliance on topological structures.

2.2.1 Problems Addressed by a GIS

A Geographic Information System is specifically designed to answer questions about spatial data. Sample issues that a typical GIS can be used to solve include questions such as [37],

- What is at a particular location?
- Where is a certain object?
- What locations satisfy certain requirements?
- What is the spatial relationship between selected objects?
- What spatial pattern does the data support?
- Spatial What ifs.

2.2.2 The GIS Model

The heart of a Geographic Information System is a database. Besides the general functionality inherited by a general purpose database, a GIS database also implements additional functional requirements for handling spatial data. The database feature also differentiates a GIS from a drafting computer system, whose only function is to produce high quality output graphics. Unlike such systems, a GIS relies on the process of associating pieces of information with features on the map. This is achieved by linking the two elements of GIS [10], namely

1. The actual spatial information representing the location and shape of geographic features,
2. The associated descriptive information about the map features, stored in conventional databases.

2.2.3 Topology

A GIS keeps track of the underlying spatial relationships of its features through topology. Topology is defined as a "mathematical procedure for explicitly defining spatial relationships", by expressing "different types of spatial relationships as lists of features" [10].

Topology is used because of its advantages of higher data storage efficiency and faster processing of large spatial data sets.

Geographic Information Systems deal with spatial data in vector-based and raster-based formats. Raster data are structured as arrays of pixels, each addressed by its position in the array. Vector data on the other hand, represents data in "straight line segments defined by the end points". The location of these end points is relative to a specific coordinate or projection system. A frequently used example is that of defining an arc as being a sequence of vectors. Similarly, an area would be represented as a collection of vectors defining the area's boundaries [37]. There are back-and-forth translation methods between the raster and vector classes, namely rasterization and vectorization algorithms. Such methods, however, suffer from information loss drawbacks. A vector-to-raster transformation, for example, sacrifices information that cannot be recovered by applying vectorization.

With the recent high demand for spatial data on the Internet, many efforts have been targeted towards porting spatial data and the GIS model to the Internet, as described in the next section.

2.3 Web-Based GIS

It has been recently recognized that the Internet can add new dimensions to the Geographic Information Systems field. The emergence of the Information Superhighway is making it possible to easily disseminate geographic information to mass markets. This can be clearly seen on the web through the various demonstrations attempted by interactive directories, entertainment and home shopping homepages. In particular, telecommunications and Yellow Pages companies are competing to implement this variety of on-line interactive services [12]. Those companies argue that integrating maps with such on-line

services provides the users with a more "intuitive and natural" way of browsing through a large amount of hypertext information.

Current applications allow tourists to view maps of their destinations from their homes. Similarly, real estate agents can query a region's databases and obtain visual as well as textual results. Moreover, the potential this technology brings into the public sector is endless. In this category, we might consider an application where the city planning office makes property maps available over the Internet, allowing residents to directly inquire about the closest public services, commercial types or even dwelling types and prices.

There are many reasons behind the motivation for having GIS ported to the Internet. For developers, the internet provides a platform independent development environment, inheriting all the client/server advantages, including having centrally controlled and maintained data, providing a better processing performance due to the distributed nature of the system, and facilitating system scalability [31]. Essentially, by developing web-based mapping tools, the basic functionality of a Geographic Information System will be globally available via the Internet[15].

Furthermore, making such functionalities globally available through the internet will indirectly provide non-professional users with the access to simple GIS tools. This allows non-professional users to have access to the specific spatial functions that they need, as opposed to paying, installing and learning a full GIS package. The users' experience will be much smoother, given the ease of the browser's use and the users' familiarity with its interface. Consequently, it is expected that training costs will become less expensive, and that adapting the system to support such functionalities will require minimal effort. Note however that there is a controversial argument stating that organizations relying on web

based GIS will not have to bear the responsibility of updating the data on a regular basis [15].

2.3.1 Supporting Spatial Data on the Internet

Supporting spatial data on the Internet implies supporting both the raster-based and vector-based formats of a GIS. As current browsers already possess the capability of dealing with raster-based formats, the problem reduces to finding ways for displaying the vector-based formatted files within the browser. There are several reasons why application developers and Internet users might require their browsers to support vector-based formats, including [32]:

- Eliminating the need for preprocessing and pre-analyzing spatial and non spatial queries.

- Taking advantage of scaling the vector data to fit different size areas without any loss in resolution.

- Making data transfer less voluminous across the internet (compared to the same data in raster format).

- Taking advantages of other basic vector data properties, such as its structure "as individual components for selective viewing without affecting the remainder of the actual image".

Despite the reasons mentioned above, most of the current implementations of web GIS are raster-based. Since current browsers lack the intelligence to display maps based on vector data, it is necessary to rasterize a map before sending it across the network. By maintaining a link back to the map's vector-based data source, basic GIS operations, such as map overlaying, object selection and manipulation, can still be performed on the raster data within the browser.

The next section describes some of the ways that are currently used to implement and/or simulate web based GIS. Advantages and limitations of each method are also briefly discussed.

2.3.2 Delivering Spatial Data on the Web

With the high demand for dynamic maps on the Internet, many web page designers have rushed to include web-based GIS in their sites, creating different methods for the display of spatial information on the WWW. Examples of such sites include producers of geographic data demonstrating their services, businesses using maps for advertising purposes and many other small participants promoting their ideas using dynamic maps. A sample of those sites, along with some of the vendors that are selling tools for creating such applications are listed in Appendix A. The methods used in such sites fall into one of the following categories [25],

- **Graphic Snapshots**

  This method relies on the display of pre-generated static graphic format maps (such as gif, tif or jpg) on the web. Users' requests are anticipated and relevant maps are prepared and stored ahead of time, at a fixed resolution and scale. The widespread use of this approach stems from its underlying simplicity, as well as its ease of use and implementation. It is best applied in simple applications that do not require a certain level of interactivity. However, due to its static nature, the method of graphic snapshots has a fixed resolution, and consequently a limited use as it lacks the important element of interactivity.

- **Spatial Database Catalogs**

  These consist of web based services providing a list of maps available for either full download or simple display. Several download formats are usually available, including raw export formats (such as Arc/Info e00 or ArcView ShapeFile), or graphical ones. The easy access to the data metadata for queries, the option for image preview as well the availability of the data for off-line analysis make this method an attractive one for many applications. Its only drawback consists of the time consuming

and rather difficult setup it requires. Similar to the graphic snapshots method, this approach still involves a pre-generation/preparation phase as well as a way to anticipate generated users' interests.

• Map Generators

In order to overcome the lack of flexibility and functionality of the above methods, this approach relies on a real time link between a running Geographic Information System and a web server. The interface usually consists of entry forms for boundaries, layers of interest, resolution, symbols, etc. The user's request is forwarded to the mapping engine on the server, which in turn processes the data and exports the results to a graphic format that is sent to the user to be viewed through the browser. The major advantage of this method is the relative flexibility and ease of providing custom maps for the user. However, it is important to note that the results still do not consist of the raw data. In addition, this method is relatively slower than the other methods because of the processing involved at the server side, the heavy client/server traffic and the more involved setup.

• Real Time Browsers

More sophisticated and demanding than the previously listed methods, this one focuses on having the browser support raw GIS data in the same fashion it supports other data types, such as text, images, sound, etc. Available demonstrations of this method (see Appendix A) have either a Java applet front end or an application developed using recent spatial development environments such as MapObjects [18]. It appears that this approach is not very different from the previous one, except for the use of existing packages that do the "spatial translation part". More on this will be covered through the rest of the thesis as we attempt to define where our approach fits among the presented categories.

Chapter 3

Design Issues and Alternatives

The proposed Prototype Internet-based Collaborative Geographic Information System (or PICGIS) consists of two separate components:

- The GIS Component: responsible for delivering and manipulating the spatial data,
- The Collaboration Component: responsible for data communication between users.

Before describing the final model of PICGIS, it is necessary to present candidate approaches for each component's design and implementation methods.

As an Internet-based system, it is assumed that the design of the product will be based on a client/server architecture, where the client and server roles might differ according to the approach that is followed. Accordingly, each candidate approach presented assumes the existence of clients and a server in the system and tries to assign different tasks to each of them depending on its configuration.

In this chapter, we evaluate these approaches in an attempt to identify a design strategy for PICGIS, that capitalizes on recent technological trends and planning needs. For this purpose, several design, development and usage criteria are identified. All candidate approaches involve basic trade-offs between several factors. Each candidate is weighed for its level of implementation effort and difficulty of use versus its general functionality and design flexibility. In addition, execution speed, system set-up (involving the separation and

identification of the client and server tasks of the system), system database connectivity and scalability as well as degree of reusability are also crucial factors that distinguish one approach from another. Table 3.1 lists the evaluation criteria classified under the categories of Design, Development or Use.

Design	Development	Use
Functionality	Ease of Implementation	Ease of Use
Flexibility/Scalability	System Set-up	Training/User Interface
Database Connectivity	Components Reusability	Execution Speed
Client/Server Tasks	Client/Server	Formats Supported

Table 3.1: Evaluation Criteria for the Listed Design Alternatives.

The chapter starts with the description of a simple solution to the problem of GIS collaboration that involves a traditional screen dumping approach. Later sections focus on the two primary components of the system (GIS and Collaboration), and discuss some of the available strategies that can be followed, along with examples whenever available.

3.1 The Screen Dumping Approach

If the problem is defined as being able to use a system for manipulating raw vector based data in a collaborative fashion, some might argue that the simplest solution to our collaborative GIS problem is a screen dumping program. A screen dumping program is a tool that automatically takes continuous snapshots of a window running on machine and displays it on the screen of another machine. If the window that is being "copied" is running a GIS program, we will have established a rudimentary collaborative GIS system.

In order assess this approach, we used a program called *xwatchwin*, that we downloaded from comp.source.x (see Appendix B). *xwatchwin* is a program that allows the user

to "peek" at a window on another X server. This utility was originally developed as a teaching aid for students learning about the use of X. In a similar fashion, this program can be used in a GIS teaching context. A typical setting involves situations where the professor is illustrating the use of a command/tool and propagating the results of his actions to the screens of the students. Other applications follow a similar pattern.

3.1.1 Advantages

The attractive features of this method revolve around its simplicity and ease of use. Moreover, it is ideal for one-way communication situations, where there is only one leader/driver that has the power to work on a map and update it, especially in the cases where such communication is reinforced by a simultaneous telephone conversation.

xwatchwin provides a good and flexible example for such an implementation with its adjustable window sampling rate and its simplicity in terms of specifying the windows by name. This method allows the leader to use any GIS system that is installed on his/her machine, taking advantage of all the capabilities of a full blown Geographic Information System. In the context of the GIS and collaboration components, screen dumping uses conventional GIS packages for the GIS component, and *xwatchwin* to emulate the collaboration component.

3.1.2 Limitations

Although it is simple, this primitive method lacks several features that a real collaborative GIS system should possess. Taking the *xwatchwin* implementation as an illustration, several such limitations can be identified. Platform dependence, in this case, depicted by the necessity of having an X server running on both the client and the server machines, is the first violation of our constraints of universal access and platform independence. This approach is also limiting due to its missing support of duplex communication situations.

In addition, this method does not provide the user with the option of saving geographic layers or even automatically maintaining a record of the sequence of events. On a more technical level, *xwatchwin* does not capture mouse or menu events, is rather slow due to its frequent fetching of the latest version of the main window, and usually crashes if the window peeked at is resized exactly at the sample time.

At a higher level, the screen dumping approach imposes several requirements on the system, including having each client user download and compile the program locally. It also requires the server to have a reliable GIS running on his/her machine, which in turn limits the coverages available for collaboration to those accessible by the server only.

The next three sections will focus on the candidate approaches for the vector-based GIS component of the system design.

3.2 Client-Based Approach for GIS Component

This section presents two client-based approaches. By client-based, we specifically target systems that focus on providing the client with all the functionalities needed in order to display and manipulate spatial data in a collaborative fashion. The two approaches here described are the pure client-based one and the plug-in solution. Both rely on the Internet as the communication infrastructure, and hence operate within the context of a web browser, bringing us closer to our defined goals.

3.2.1 Pure Client Based Approach

With the recent availability of Internet programming languages such as Java, it has become possible to translate existing GIS code to such languages, making both the programs and the data they access universally available on the web. This approach is labeled "pure client-based" (Figure 3.1) because it essentially packs all the GIS functionalities in

the client. Once accessed, a site using this approach automatically initiates the download of the code onto the client's machine, which can then run the associated functions within the context of the web browser.

With the recent explosion in the availability of web programming development environments such Symmetrix's Java SuperCede and Symantec's Java Visual Cafe, it has become relatively easy to migrate existing GIS to Java. This is sensed on the web through several illustrations posted by competing companies, such as Internet GIS.com's ActiveMaps (http://internetgis.com) and EuroCom's ArcJ (http://www.affari.com/eurocom/arcj/).

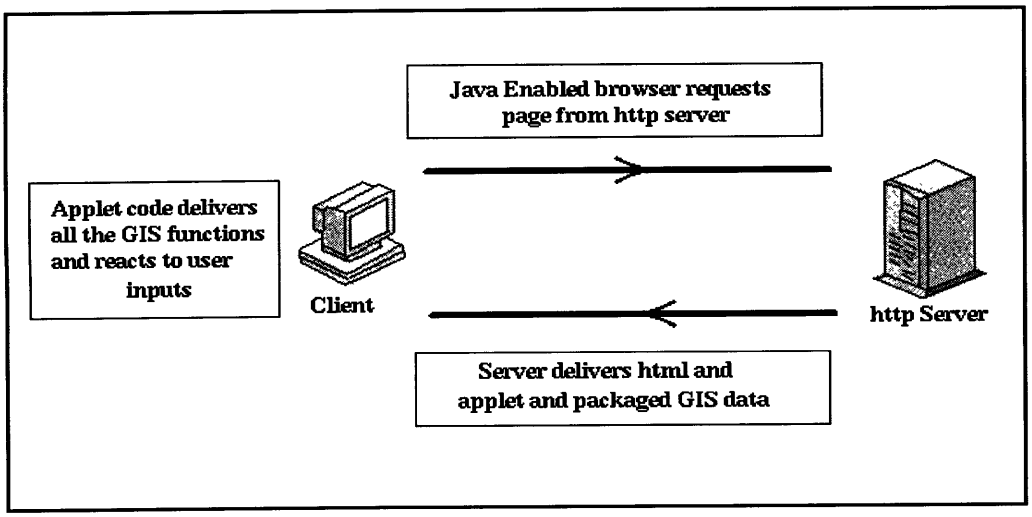

Figure 3.1: Java-Based Client Approach.

• Advantages

Due to its reliance on the download of all the GIS package to the client, this approach is characterized by its rather fast client execution and response time. This is justified by the fact that all the functionalities are encapsulated in the client and therefore no communication with a server is necessary (hence the name "pure client

based"). Moreover, this approach inherits the portability of a platform independent language such as Java, thus requiring minimal amount of installation or setup to run on the client machine. In fact, the only prerequisite for using this method is the existence of a Java enabled browser on the client's machine. Finally, due to the fact that the GIS functions and methods are completely regenerated using Java, this approach allows the reading and manipulation of vector-based data. The ActiveMaps program, for example, supports the popular ArcView and MapInfo maps formats.

- **Limitations**

  Although this method sounds both attractive and efficient, it exhibits several drawbacks starting by the costly development effort required to re-code a whole GIS system in a new language. It is also important to note that due to the client-centric nature of this approach, it requires a high client horsepower and involves a high download time (due to the huge amount of code being downloaded). Although the emergence of vendor dependent proprietary systems shows that the market is moving towards incorporating GIS functionalities into the web browser, it is hard for users to add custom based functions to such systems. In addition, these efforts have not yet explored the possibility of integrating collaboration tools with the GIS functionalities.

3.2.2 Client via Plug-In

The client via plug-in solution (Figure 3.2) is, in a way, a variation of the pure client based method. The main idea of holding the client responsible for locally performing all the processing needed for the system's operations still applies. However, the approach in this case is different. According to the Netscape browser definition [20], plug-ins are "programs that extend the capabilities of the browser in specific ways", by enhancing the power of the browser to support various additional data formats. Examples of currently

available plug-ins include those supporting 3D, animation, virtual reality, audio and video among many other documents formats.

According to the same reference, software companies are currently developing plug-ins for their products at a "phenomenal rate". They are also developing what is known as plug-ins extra, which further enhance the functionalities of existing plug-ins. Tools such as the Internet Developer Toolkit of Power Builder 5.0 are used for the development of basic and extra types of plug-ins.

Although, to the best of our current knowledge, there are no available plug-ins for any format of spatial vector data, we believe that such an otherwise tested and proven technology might constitute a suitable solution for the problem of display and manipulation of spatial data over the web. Such initiatives are expected to originate from leading GIS software vendors such as ESRI and Intergraph.

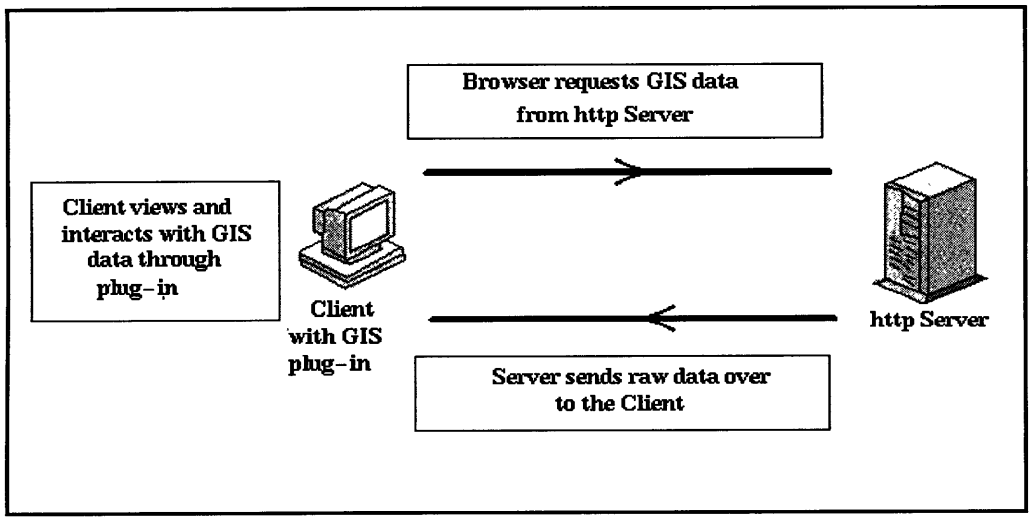

Figure 3.2: Client via plug-in.

3.2.3 Advantages and Limitations

As a subclass of the general client-based method, this method exhibits the advantages of speed and minimal network traffic. It also provides a convenient method for the users to work with raw spatial vector data within their browsers.

On the other hand, this method involves a considerable amount of plug-in development for each data type, for all platforms, operating systems and browser systems. In addition, there might be some controversy regarding the specific spatial formats available, as there currently is no global standard for spatial data format. Hence, it might be years before this market converges to a standard. Today, the lack of such a standard is fueling a competition that does not address collaboration issues in an interoperable way.

This section concludes the discussion of the client-based approach of accessing Geographic Information Systems data on the web. The next section presents the server-side solution to the same problem.

3.3 Centralized Approach for GIS Component

The server-based approach (Figure 3.3) is the exact opposite of the client-based one. In this case, the design is such that all the processing needed for the task is performed at the server. Common Gateway Interface (CGI scripting) has been one of the first illustrations of allowing such a centralized approach. In general, the basic steps of this approach can be summarized as follows:

1. The client sends a request to the server to perform a certain operation. The client can send those commands in the form of URL (as is the case of CGI), dynamic html or java/ javascript commands.

2. The server is pre-programmed to handle such requests and process the client's request.

3. The server returns the result of the operation, if successful, in a data format that is

readable by the client's browser, such as text or recognized graphical formats.

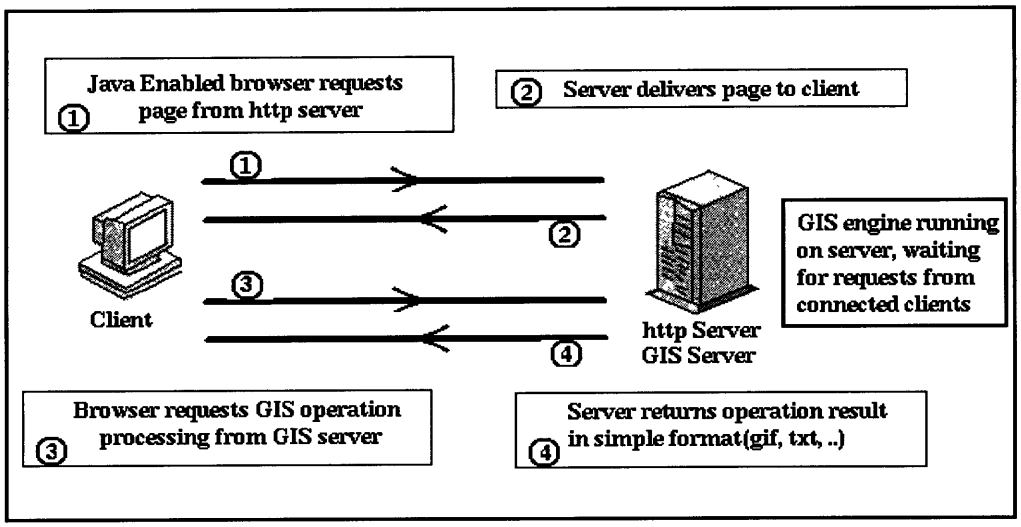

Figure 3.3: Centralized Approach.

This approach is currently in use by many GIS software vendors as those vendors realize the escalating demand for dynamic maps on the web and the large audience they can reach through the Internet. Examples of such applications include the Environmental Systems Research Institute (ESRI) MapObjects and ArcView Internet Map Servers. Those products allow users to "browse, explore, and query active maps" [18] using their browsers.

These products (Figure 3.4) propose to extend the power of the well-established vendor products (such as ArcView and Arc/Info by ESRI) to serve maps over the web using proprietary Internet Map Servers. These products include a web-server extension, programmable objects, and ready-to-use applets or scripts, used to access the server functions. The communication occurs through URL format packaged requests.

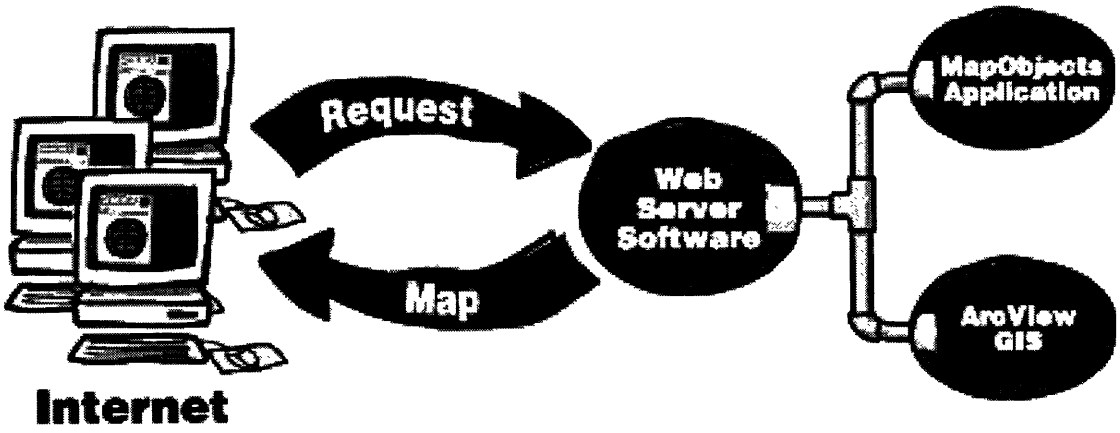

Figure 3.4: ESRI Internet Map Servers Model. (Source: http://www.esri.com [8])

3.3.1 Advantages

The centralized server based-approach is considered as a candidate solution to our problem of web-based GIS because of its various advantages topped by the minimal requirements imposed on the client under such a configuration. The only client requirement is the access to a simple browser. Due to the centralization of all processing at the server, the client browser does not even require the addition of any plug-in. This in turn implies a greater flexibility in the construction of both the communication layer and the client graphical user interface.

From development perspectives, this method is attractive because its centralized feature allows for flexible development environments on the server. Consequently, developers can reuse their code modules for other server-based applications. Finally, for our purposes, this method takes advantage of all the power embedded in currently available GIS products, without having to redevelop a GIS product from scratch, in a new language.

3.3.2 Limitations

Although the server-based architecture lends itself to a thin client implementation, it is not that efficient in terms of network traffic. The continuous and excessive communication between the server and every connected client can cause some serious problems regarding the network performance and decrease the processing speed of the server itself. The higher the number of connected users, the higher the server's load, and the busier the network traffic becomes, causing this configuration to become less efficient in many aspects.

Table 3.2 lists all the alternatives described thus far, along with a brief description, an example and a summarized list of their advantages and limitations in the context of our design objectives for the PICGIS.

Approach	Description/ Example	Advantages	Limitations
Screen Dumping	frequent screen update and refreshing **Example:** *xwatchwin*	- simple to use - no implementation involved - ideal for one way communication (along with telephone) - ability of using existing GIS on the server's side	- platform dependent - no duplex communication - no option for saving/ recording events - constrained by server's system - requires download and compilation of the program on every client

Table 3.2: Advantages and Limitations of Candidate Design Approaches.

Approach	Description/ Example	Advantages	Limitations
Pure Client-Based	porting existing GIS code to a platform independent language such as Java **Example:** ActiveMaps by InternetGIS.com	- very fast - platform independent/ portable - no installation or setup - use of commercial browsers - minimal network traffic - support of vector based data	- high client horsepower - long download time - vendor dependent - high development effort and cost - unnecessary redevelopment of a whole GIS
Client via Plug-In	develop or use a plug in for vector based spatial data **Example:** Power Builder 5.0 Internet Developer Toolkit	- fast - use of commercial browsers - minimal network traffic - support of vector based data	- involves setup - involves development of plug-in for every platform - identification of standards supported by the plug-in - no available implementations yet
Centralized/ Server-Based	setup a program on the server that accepts clients' requests and returns the results in a form readable by the browser **Example:** ESRI MapObjects and ArcView Internet Map Servers	- thin clients - flexible client interface - makes use of existing power of GIS - flexible development environment - code reusability - no need for plug-in - system scalability	- high server load - heavy network traffic - considerable server and communication scheme setup - inefficient/slow with high number of connected users

Table 3.2: Advantages and Limitations of Candidate Design Approaches.

3.4 Other Approaches for GIS Component

It is important to realize that the previously described design alternatives represent extreme approaches that can be followed to create an Internet-based collaborative system. Each of these extreme cases possesses positive characteristics as well as some drawbacks. With these trade-offs in mind, a designer is free to create a combination of these methods that best suits the project's objectives. We will refer to this kind of approach as the hybrid one because it combines features from both client and server-based methods.

The basic idea involves subdividing the project essential tasks between the client and the server. While the server for example is still held responsible for all the database queries and updates of the project, the client might be designed to handle some of the graphics operations pertaining to spatial data such as the zoom in/out, pan and even the identification and visual-select types of operations. Essentially, the objective is to put reasonable functionality in the client and attempt to minimize the communication with the server by sending operations that are best performed by the latter.

Now that we have covered the discussion pertaining to the porting of GIS data to the Internet through the various alternatives, we need to address the approaches followed to implement the collaboration component of the project.

3.5 Collaboration Component: Issues and Alternatives

The Internet-based collaborative GIS prototype involves two major design decisions, the first of which consists of bringing the raw spatial data to the web, the second, which is the focus of this section, is that of connecting the clients together through the system. The issue in this case is the determination of whether the client or the server is responsible for forwarding relevant information to other connected users. This decision largely depends

on the application at hand and on the level of development and scalability of the final product. Generally speaking the approaches fall under two methodologies, peer-to-peer (or client- based) and centralized forwarding (or server-based).

3.5.1 Peer-to-Peer Communication

In peer to peer communication (Figure 3.5), each client controls the forwarding of operations, requests or their results to the other clients. The client can be responsible for determining which of the other logged-in clients need the new information update or can blindly broadcast the data to everybody. In the latter case, the clients have an embedded mechanism that determines whether new information is needed for processing or can be simply ignored.

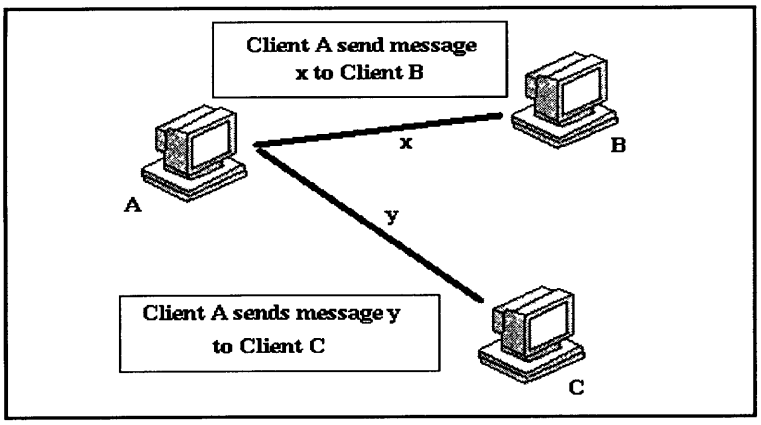

Figure 3.5: Peer-to-Peer Communication.

According to Table 3.3, the advantages of peer-to-peer communication include having client independence of a centralized server, hence making the overall system more stable. In this case, a system failure on the server's end does not imply drastic performance deterioration. Moreover, the fact that each client can be independently upgraded with more functionalities makes the system a more scalable one. This point is also reinforced from a

usage perspective whereby adding more users to the system does not highly affect the speed or load of any particular client.

As for the drawbacks of this model, they parallel any client-based modeling approach. Basically, the drawbacks are sensed through the high processing load on the clients as well as the additional client components needed to monitor other clients in the system. A client embedding all these capabilities is more likely to be a "fatter" one, and consequently affects the client application downloading time as well as local storage space/memory.

3.5.2 Server-Based Communication

Contrary to the previous model, the server-based one (Figure 3.6) is based upon the exclusive dependence of the system on one centralized server component. In this case, all clients have to report to the server, which they use to communicate among each other. Depending on optional broadcasting flags attached to messages, a smart server can effectively choose the users to whom it forwards the data or, in the worst case, can broadcast all messages to all clients, who in turn will know how to filter the arriving messages.

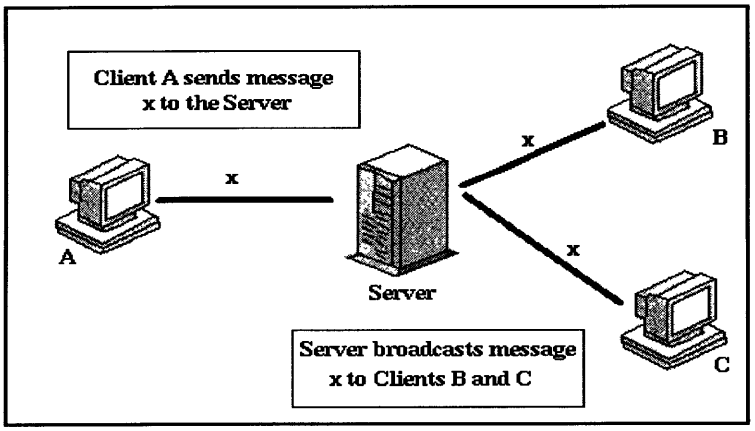

Figure 3.6: Server-Based Communication.

By putting more functionality in the server, the clients shift to a thinner and simpler design. This approach suffers from a high server load and a heavy network communication

traffic. An important issue in this case is the actual location of the clients relative to the server' s location. A very inefficient configuration might occur if two clients are physically situated close to each other, but nonetheless have to communicate through a server physically miles away. A peer-to-peer connection might work better in that case. Determining the most efficient configuration depends highly on the particular application at hand. Knowing the advantages and limitations of each method (Table 3.3) is an important stage of this process.

Approach	Description	Advantages	Limitations
Peer to Peer	relevant information is directly forwarded from a client to another peer client without the intervention of a server or a forwarding service	- independent of centralized processing - no unnecessary communication or data forwarding - easily scalable system - robust implementation	- more processing load for clients - difficulty in maintaining list of relevant connected clients in each client
Server Based	any information transmitted from a client goes through the server, which in turn determines the list of connected clients that need this new data forwarded	- thin clients implementation - server keeps track of clients' status - easier implementation	- high server load - more server responsibilities due to centralized approach - server to clients physical distance issues and networking - excessive network communication

Table 3.3: Characteristics of Implementation Methods for Collaborative Aspect.

The next two chapters present the methodology followed in the design of PICGIS along with the system architecture, the software implementation and a test scenario.

Chapter 4

System Architecture

In light of the issues and alternatives highlighted in the previous chapter, we are now in a position to decide on the best approach to be used for PICGIS. In this chapter, the elements of the system and the interfaces between those elements are identified using a standard software engineering approach [13]. We start by listing the software and functional requirements. We then map them into an architecture, whose constituents are well-defined modules, each targeting a specific aspect of the overall system. After the identification of the major components, the tasks associated with each component are allocated to either the client or the server elements of PICGIS.

4.1 Analysis of System Requirements

The objective of the thesis is the design and implementation of a prototype Internet-based collaborative Geographic Information System (PICGIS) as a meeting place for GIS users, where they can cooperatively work on GIS maps and databases in the context of a given problem at hand. Recall that in Section 1.4, we stated that PICGIS can be considered as the intersection of three research domains, namely the Internet, GIS and GroupWare. This study focuses on combining these application fields and their properties in a system targeted towards people familiar with GIS, and can benefit from an on-line mapping collaboration forum. We are particularly interested in demonstrating the viability of combining

these fields, without necessarily developing a complete set of functionality for each component of the prototype. Knowing the system's potential users and their application domains is the starting point for defining that functional subset for PICGIS and the requirements related to each of its components.

4.1.1 Requirements Specification for GIS Component

Since porting GIS spatial and database functionalities onto the web is not the sole goal of the study, it is necessary to limit the scope of functionalities we choose to include in the prototype, to those we identify as essential to any GIS operation. Functionalities to be incorporated need to be simple and essential, especially for the case of "mapping collaboration". The idea is to have users use PICGIS for discussion, collaboration and elementary spatial analysis on a set of maps and overlays, prepared in advance, using more sophisticated GIS packages. However, this does not preclude the option of enhancing PICGIS in the future, to include sophisticated GIS functionalities.

This section lists a small subset of GIS features that form the least common denominator to all GIS systems. In other words, any prototype should incorporate at least the following elementary GIS operations:

- Graphically overlaying maps and switching layers on and off.
- Zooming in and out of the current layers (as well as zooming to the extent of the current layers).
- Getting information about a layer's underlying structure and databases.
- Retrieving tables associated with spatial objects in the layers.
- Constructing simple one dimensional queries and displaying the results in both spatial and tabular formats.

We believe that the listed functions, although limited in number and functionality, constitute a basis for any GIS system. Other functions can be built on top of those five main

elements, that cover both the presentation and database aspects of GIS. Finally, from a data perspective, it is also important to emphasize that PICGIS' s design should be generic, allowing the display, manipulation and querying of any map conforming with the system's supported GIS formats (that are to be specified later).

4.1.2 Requirements Specification for Collaboration Component

At this stage of the design, the collaboration scheme used by the system has to be determined in order to specify the users' interaction process. At the prototype level, it was decided that the system follow a **free setup** [16], where all participants may take control at any time, even simultaneously. This setup is particularly efficient in the case of a small number of geographically dispersed professional participants who are using the system for well-defined visual discussion of mapping issues. The free setup collaboration approach can be easily modified at a more advanced implementation stage to add more control to the negotiation process.

As stated in Chapter 1, the system's goals of GIS collaboration are highly enhanced by the inclusion of another communication medium such as the telephone. However, in case this option is not available for the users, PICGIS should also support some form of a chatting mechanism for on-line real-time discussion, in parallel with the other activities it supports.

Finally, along the same lines, the system should also include a drawing panel or a board that displays the result of the latest layers' update or queries. This panel can be extended to include some drawing capabilities for users, enabling them to add their own cosmetic layers. The panel should at least include the basic drawing functionalities, such as drawing lines, rectangles and ovals with different colors and line widths, as well as selecting, grouping and deleting the created objects. Addition of textual information on top of maps should also be made available for mapping annotation purposes.

4.1.3 Specification of Other Requirements

The system is designed using an object-oriented approach, indicating that the software is organized as a collection of objects that contain both structure and behavior. This approach has several benefits including design flexibility, easier information exchange, easier debugging, and code maintainability and reusability.

For the purpose of system evaluation, the design strategy is finally reviewed to ensure a scalable design in terms of number of connected users and functionalities. In addition, we evaluate the extent to which PICGIS meets the system requirements in terms of performance and usability.

4.2 System's Functional Components

In order to satisfy the aforementioned requirements, the top level components of the system need to be identified. According to those requirements, PICGIS consists of three major elements: the GIS element, the collaboration element and the user interface element. In the next few sections, we describe each element in detail, with emphasis on each element's method of processing information generated and used by PICGIS.

4.2.1 The GIS Component

The GIS component functionality is two-fold. It performs conventional GIS functions, and handles access to the geographic data and its underlying databases through the web. Following the discussion in Section 3.3, the centralized server model for the GIS component was found appropriate for our study because

- It avoids the need to rebuild a brand new client based GIS,
- The approach takes advantage of a powerful and dedicated GIS engine running on the server, and
- The advantages of a thin client are desirable in our case given the targeted system

audience.

Regarding the particular selection of the GIS engine/server, ArcView 3.0 was a favorable candidate because of its availability on all platforms, hence providing server portability and greater implementation flexibility. In addition, Avenue, its scripting language is object-oriented, making it perfectly compatible with the overall object-oriented system design. The data types supported by PICGIS, as a result of choosing ArcView 3.0, include a wide variety of formats such as Arc/Info coverages, ArcView shape files, SDE formatted data, DWG, DXF as well as raster graphic file formats.

The GIS information flow diagram is presented in Figure 4.1. In this flow diagram, information flow is represented by arrows, transformations or processes are represented by circles and information stores are depicted by double lines [13]. This diagram shows that when a browser request for GIS maps or databases is initiated by a user, the request is encapsulated in a message, that is sent to the server for processing. In turn, the server decodes the message and transforms it to an executable GIS script. The GIS server executes the script, and the results (maps or databases) are saved into a GIS information store and then transformed into a browser readable format files such as gif or txt. Once the files are created, the server instructs the clients to access those files and update their interfaces, to reflect the latest changes caused by the initial request.

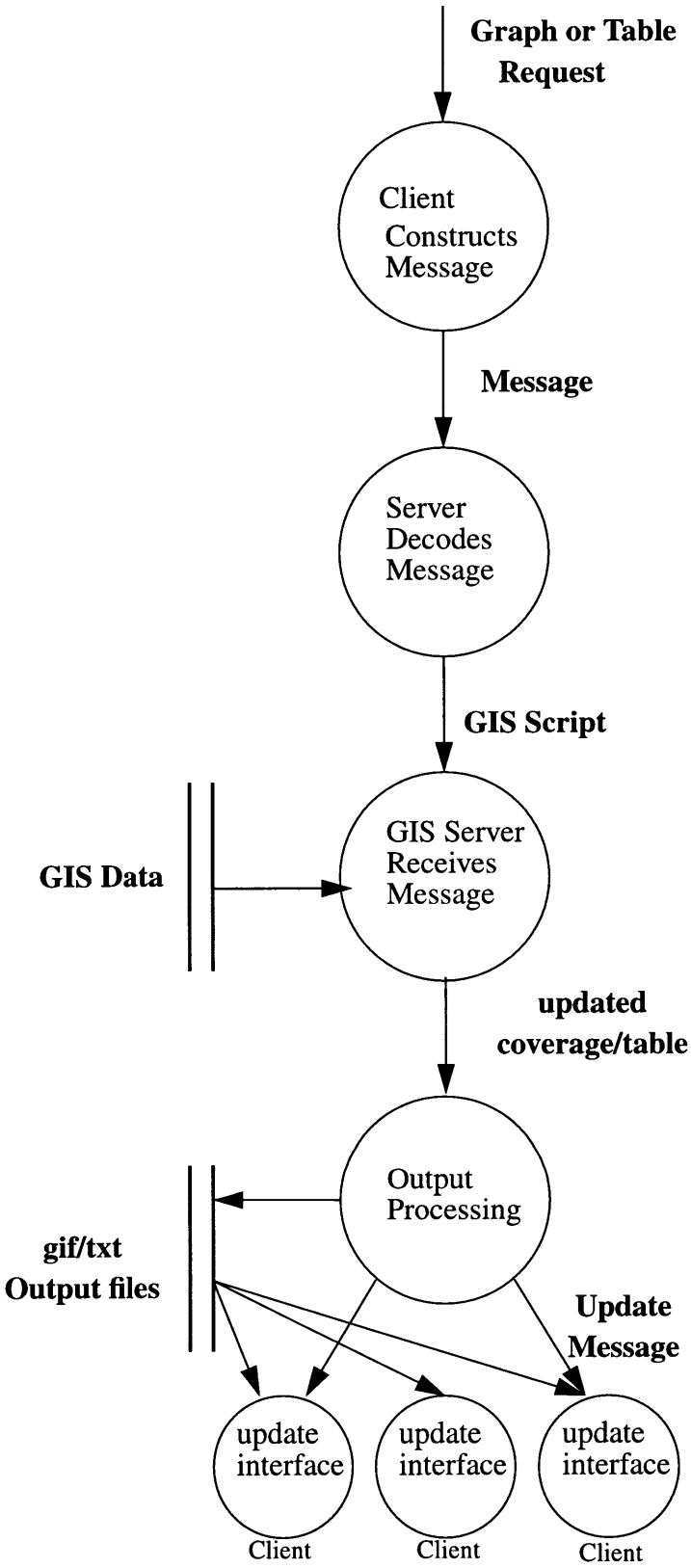

Figure 4.1: GIS Component Data Flow Diagram.

4.2.2 The Collaboration Component

The collaboration component is responsible for creating and maintaining the communication links between the users of PICGIS. As a result of the discussion in Chapter 3 about the alternative approaches available, it was decided to use the centralized approach (refer to Section 3.5), i.e. the approach of forwarding all messages and requests to all clients and letting the clients decide whether they need the information received and if so, how to process it. This basically has the effect of replicating the user's action on the screens of all the other logged-in users. Therefore, all the screens are identical throughout the session's lifetime.

The information flow diagram of the collaboration element is depicted in Figure 4.2. The simple information flow in this case is a result of our earlier choice of the free setup collaboration approach. Once the system receives any input from a user, the appropriate message is constructed and forwarded to all the users, including the sender. In the case of an interaction event with the user interface (e.g. mouse button click or textual input), the sender disregards the incoming message because the interaction has already taken place at the sender's screen. In case of layout or table update requests, all clients, including the sender, react to the incoming message by fetching the latest version of the updated object from the server.

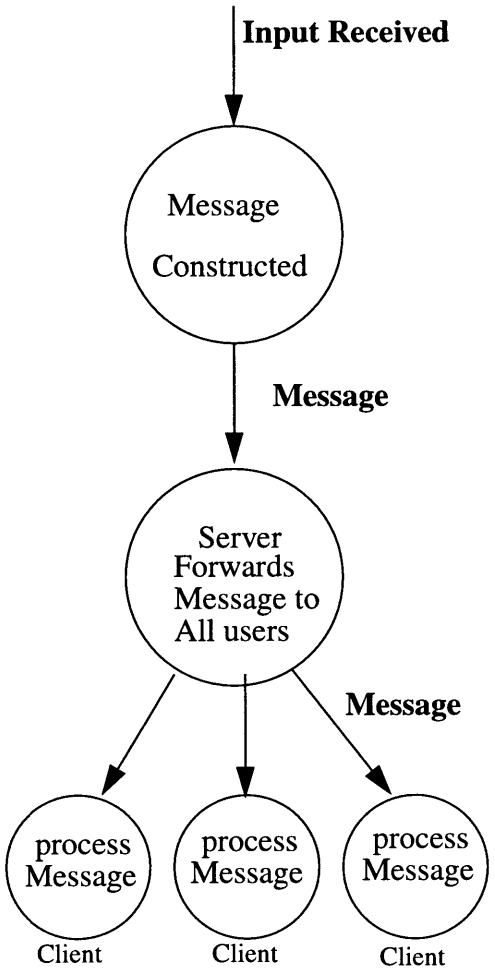

Figure 4.2: Collaboration Component Data Flow Diagram.

4.2.3 The User Interface

The user interface involves the design of an appropriate control panel for accessing and displaying the multiple layers of a GIS map. The interface must also provide a chat/ discussion panel as well as a drawing panel for adding annotations to existing maps. Therefore, we see that the user interface includes an element for each of the components discussed in Section 4.1. Figure 4.3 shows a picture of the user interface used in PICGIS.

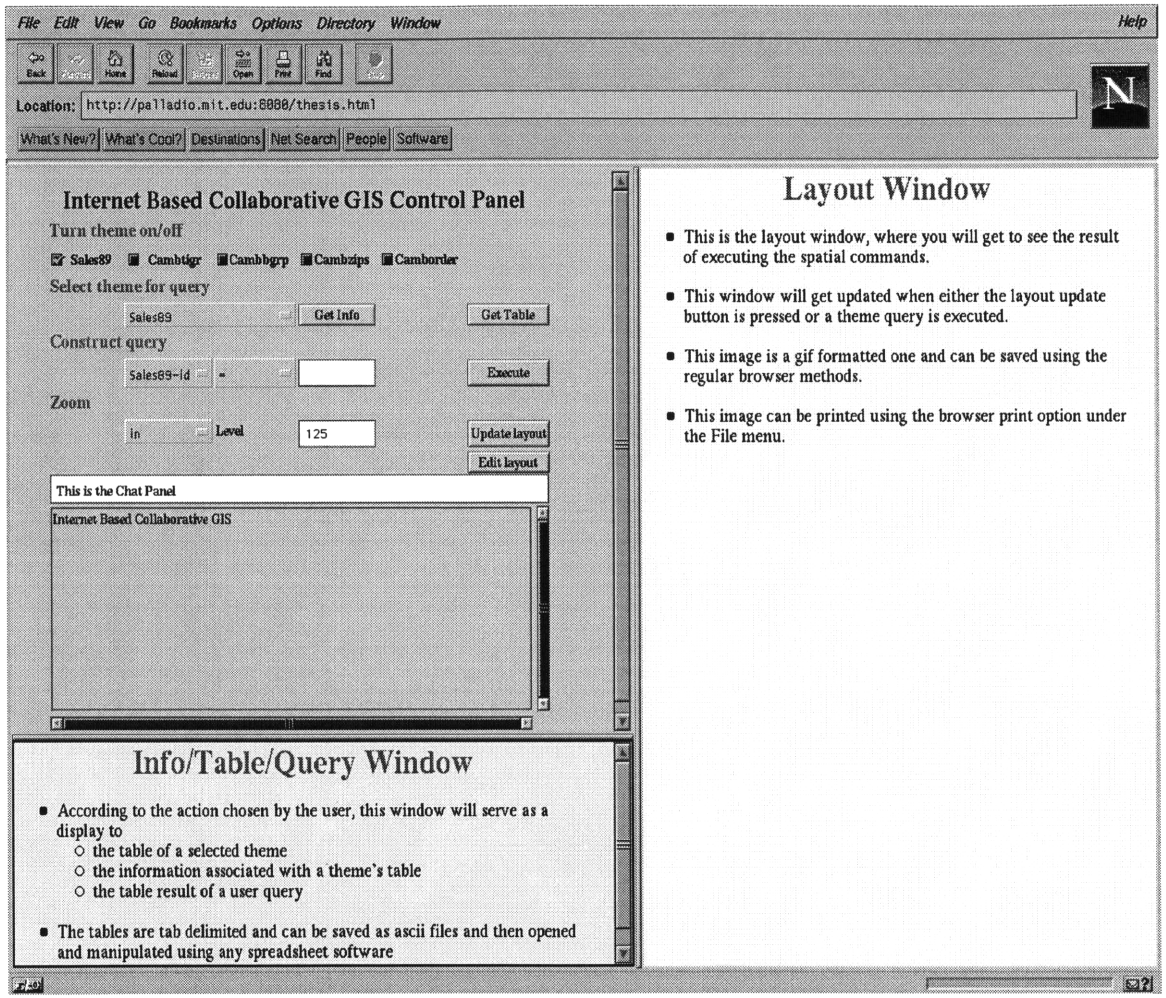

Figure 4.3: PICGIS User Interface

At this stage, the system requirements and the corresponding components of GIS data access and collaboration have been determined. In the next sections, we break up each component into individual functionalities that are assigned to the client and/or server elements of the distributed system. The overall software structure of the client/server-based PICGIS is depicted in Figure 4.4.

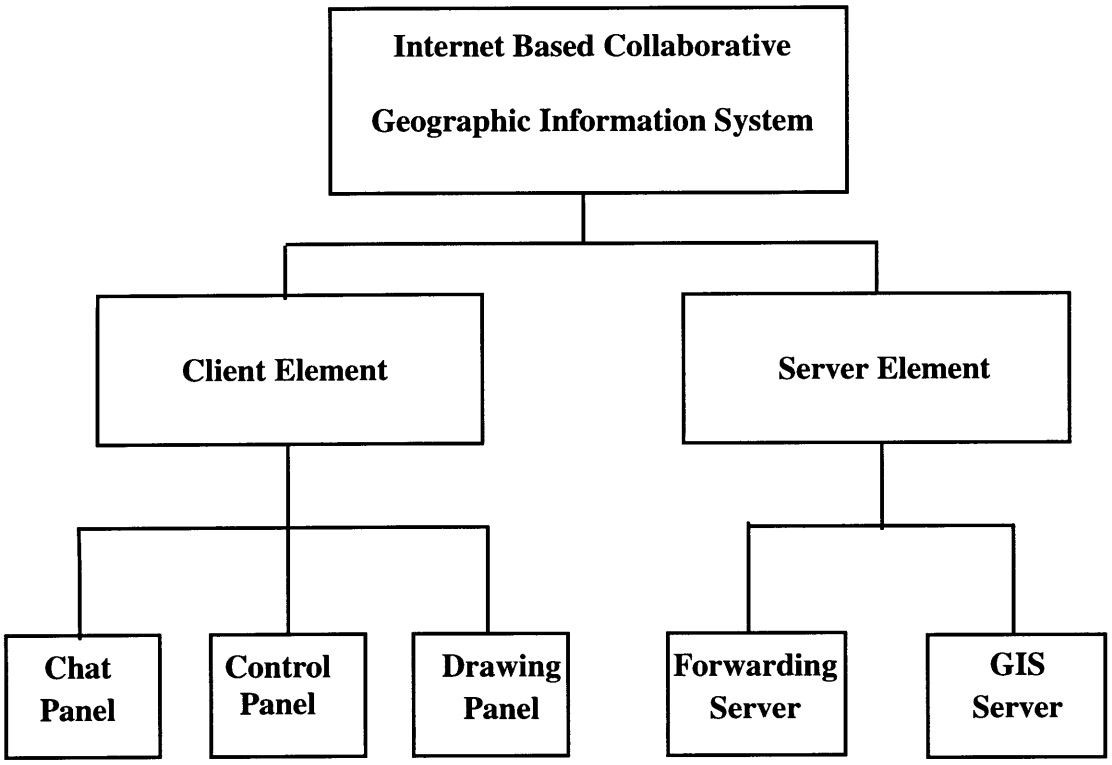

Figure 4.4: Software General Structure.

4.3 The Server

According to the earlier description of the GIS and collaboration components, the server mainly performs two functions:

1. Connecting the clients to the GIS engine in order to perform the GIS requests.

2. Connecting the clients to each other by forwarding each user's actions to all other users.

In order to meet the above objectives, the server is split into two sub-elements, namely the GIS server and the messaging server.

4.3.1 GIS Server

As previously pointed out, the GIS package used in PICGIS is ESRI ArcView 3.0. ArcView 3.0 allows other clients to connect to it via Remote Procedure Calling (RPC). RPC provides a framework for implementing remote access to a system by creating a distributed environment that is established and controlled at the procedure level. In other terms, RPC allows the users to transparently send remote procedure calls to ArcView.

In our case, ArcView is setup to run an RPC server that can respond to clients' requests. ArcView RPC relies on the Open Network Computing (ONC) standard [2] and supports RPC through the *RPCClient* and *RPCServer* classes. In this setup, the role of the GIS server reduces to receiving requests through RPC, executing them, and then exporting the results into browser readable formats. Those outputs are conveniently saved on the server to be accessed by the clients through the regular http protocol.

The setup of the GIS server also includes the implementation of several Avenue scripts for handling the incoming client commands. The scripts have to span the GIS requirement specifications listed in Section 4.1.1, and also have the capability of exporting the maps and databases into other formats, and saving them onto the server.

4.3.2 Messaging Server

The messaging server is the server element that receives the requests from the connected clients and either forwards them to other clients or passes them on to the ArcView server for execution. The server is designed in a multithreaded way, creating a thread for each client, hence allowing the receipt of messages from different clients to occur in parallel, and without interference. Figure 4.5 shows the server structure flowchart. As part of

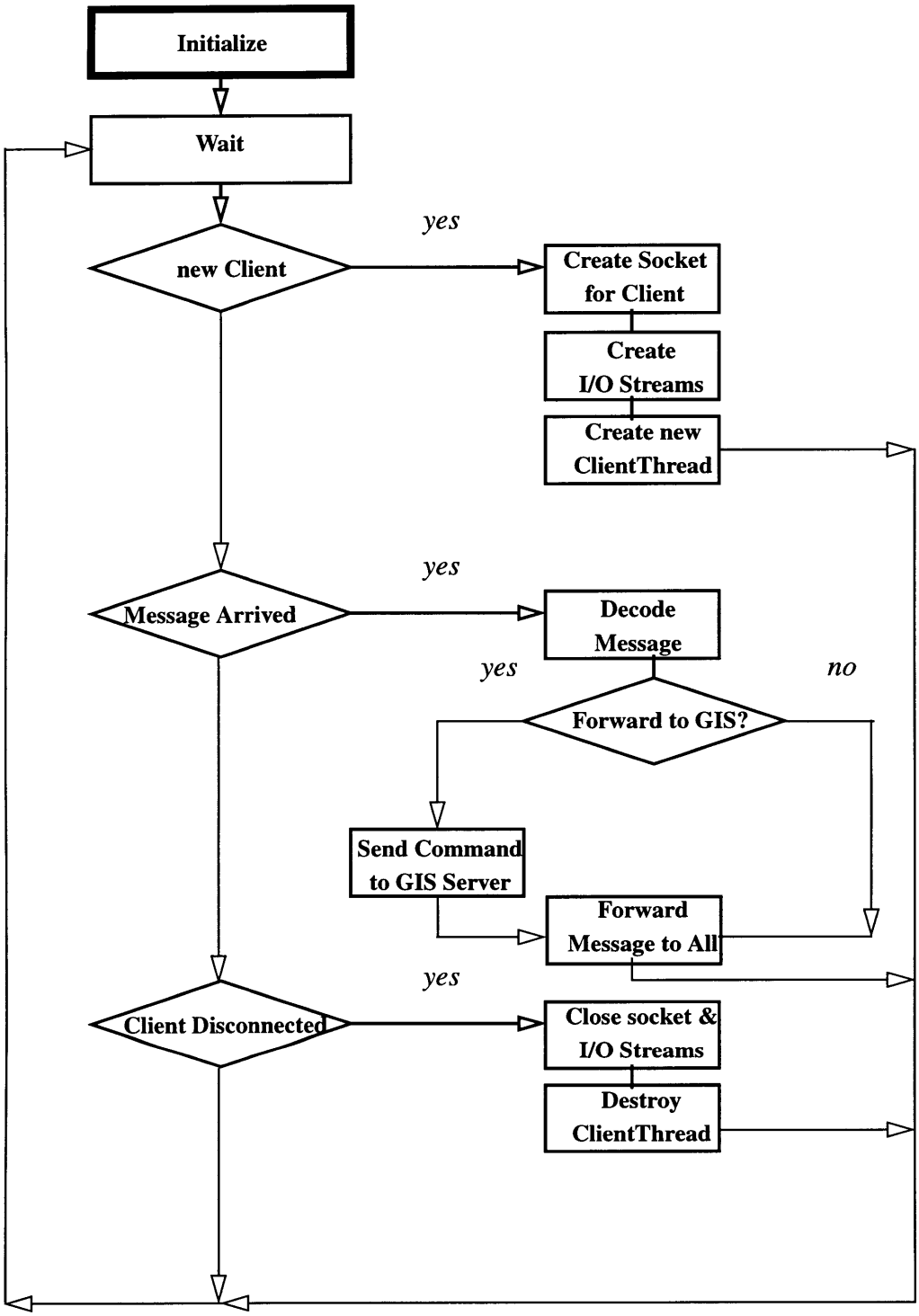

Figure 4.5: Messaging Server Structure Flowchart.

the initialization, the messaging server instructs the GIS server to dump the data describing the current layers and their attributes to a file that can be accessed by the clients upon connection. After initialization, the messaging server is always waiting for an event to happen. This event might be either a client trying to connect/disconnect or a message sent by one client. If a new client is attempting to connect, the server sets up a new socket and its corresponding input/output streams, and then creates a new thread for that client. Similarly, if a client is trying to disconnect, those elements are closed or destroyed.

If however a new message arrives, the messaging server determines, according to the message type, whether to forward it to the GIS server. Regardless of the message type, the messaging server forwards the message to all the clients, to inform them of the sender's action and prepare them for an interface update if necessary.

4.4 The Client

The client structure is best understood using the flowchart in Figure 4.6. Briefly, after the initialization phase, the client connects to the server and downloads the layers and attributes information file in order to know which layers' names and attributes to display. Once connected, the client can either send or receive messages. A message is constructed and sent when the user provides an input through the interface (or control panel).

If the client receives a message form the server, it first determines the sender of the message, then the message is directed according to its type to the appropriate panel, i.e. the control panel, the chat panel or the drawing panel.

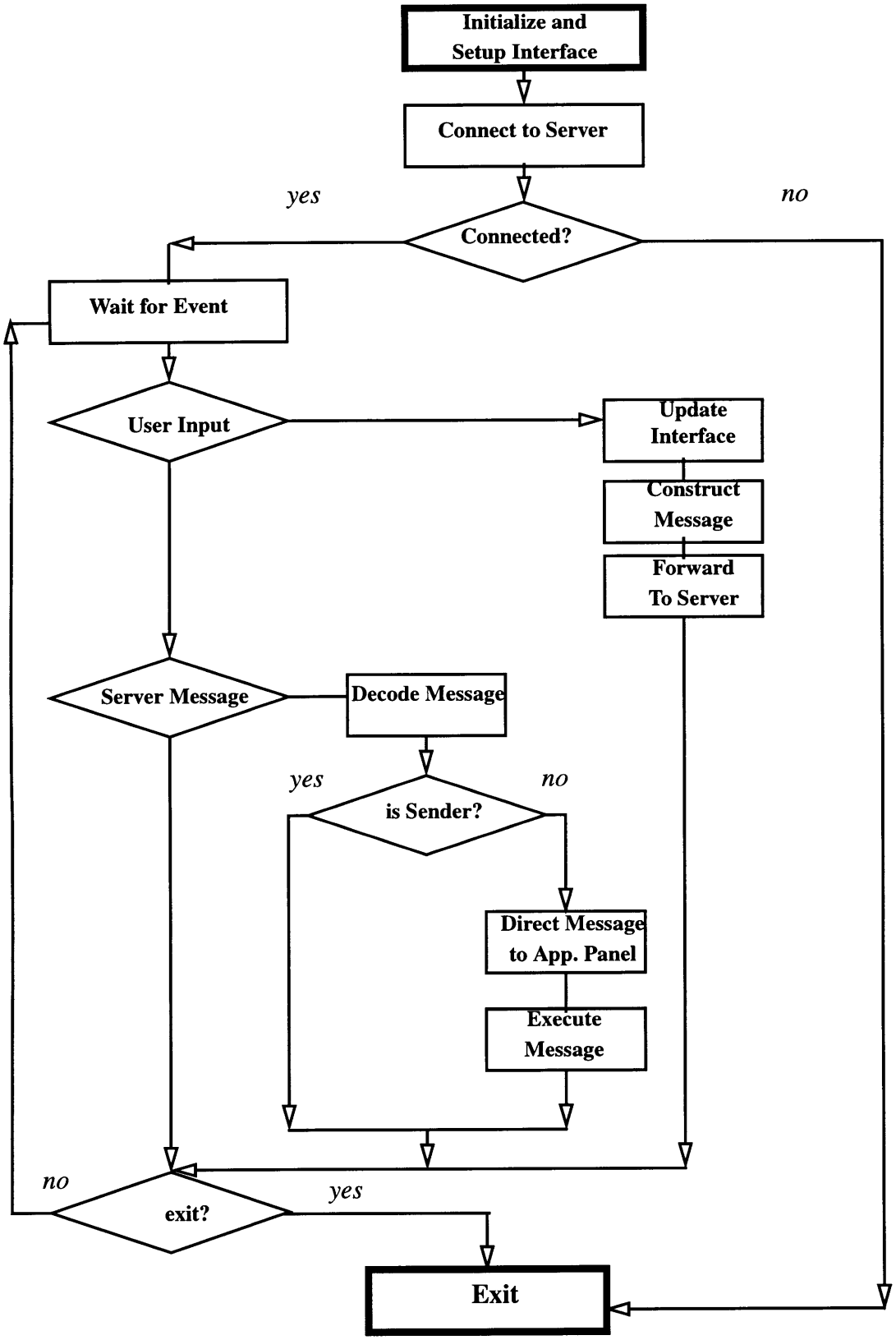

Figure 4.6: Client Structure Flowchart.

4.4.1 The Chat Panel

The chat panel is the part of the client responsible for receiving and sending text messages to the other logged-in clients. It acts as the textual communication medium between the participants in the session. Each message typed at the prompt of this panel is sent to the others preceded by the sender's username.

In addition, the chat panel also displays all the GIS commands that are sent to the server, for the benefit of the users who are interested in monitoring the sequence of those commands.

4.4.2 The Control Panel

The control panel (refer to Figure 4.3) provides the primary interface to the GIS functionalities mentioned in Section 4.2.1. Any event occurring in this panel is automatically forwarded to the server in order to replicate the event on the other client machines. Similarly, when this panel receives a message from the server, it updates its information and its interface according to the message.

The control panel is linked to a vital element of the system, referred to as the ArcView Interface (*AVInterface*). Each client is equipped with an *AVInterface*, whose role is to make the interface and behavior of all clients consistent and to provide a temporary storage space for the ArcView commands. It is also responsible for storing the session's state parameters, such as the current selected theme and the current selected attribute, as well as the ArcView scripts, which are sent to the server upon user request. The *AVInterface* continually updates the session's state parameters with every user input, and forwards an ArcView command to the server, after updating the corresponding state parameters. The concept becomes clearer with an example, which will be presented in Section 5.3. AVInterface is also explained in more technical detail in Appendix C.

4.4.3 The Drawing Panel

The drawing panel is only active when the users wish to add symbols and annotations to the latest generated layout, creating a visual discussion environment similar to that of a white-board's. In PICGIS, this portion of the system is not linked to the GIS server, implying that the complete set of GIS functionalities of PICGIS are not available when using the drawing panel. Instead, a set of basic tools are included for drawing on a cosmetic layer above the map originally generated by the server. The drawing panel is independent of the other panels, and therefore all panels can be active simultaneously, allowing users to perform GIS functions within the context of the remaining system panels.

The next section describes in more detail the interaction process between the clients and the server.

4.5 Linking the Clients and the Server

Making the clients and the server communicate using a consistent set of pre-defined possible messages is the basis of the *Message* object. Both the client and the server refer to this object in order to construct or decode any message. There are basically four message types:

- Messages to be forwarded to the ArcView server.
- Messages to be forwarded to the Control panel.
- Messages originating from a drawing panel.
- Messages originating from a chat panel.

This object also includes convenient methods for decoding messages, finding out the message's sender, and determining the type of the message. All the messages along with the methods associated with this object can be found in Appendix E.0.8. The details of the implementation along with an example make the previous descriptions of the system elements clearer in Chapter 5.

Chapter 5

Implementation

The next step in the software engineering process of PICGIS is the implementation phase, which we describe in this chapter. The required computer modules, data, objects and interfaces are derived [13]. As previously indicated, our prototype follows an object-oriented design methodology and a top-down implementation approach. Such a structure allows for integration testing where the final product is assembled and tested in a stepwise fashion.

All development was done on the SGI Irix platform. Unless stated otherwise, the main modules were developed using Java 1.0.2 (refer to Appendix D for the system's class hierarchy). Java is an object oriented programming language developed in 1991 by Sun Microsystems as part of a research project to develop software for consumer electronics devices. The language quickly moved to be the leading Internet programming language. Java is unique because it is "simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high performance, multithreaded and dynamic" [24].

The next sections present the implementation details regarding the server and the client portions of the system, along with an example showing the setup and the use of the system in a typical setting.

5.1 The Server

The server implementation simply follows the system architecture specification. As noted in the design Section 4.3, the server consists of two main elements: the GIS (ArcView) server and the messaging server. The implementation of these servers is detailed in the next sections.

5.1.1 The ArcView Server

The ArcView server is setup to receive requests from clients through RPC (see Section 4.3.1). The ArcView RPC server is started by issuing the *RPCServer.start* request. The arguments of the start command consist of the ArcView server machine name, identification number and version number. Any client can then connect to ArcView by specifying those parameters.

By establishing such a connection, we allow our application to connect and make requests to ArcView via Avenue statements. Avenue is the customization and development environment for ArcView. It allows developers to customize ArcView's interface, modify its standard tools and behaviors, create new tools and integrate ArcView with other applications. Avenue is a compiled object-oriented scripting language [2].

Along with the RPC server script, other scripts had to be created in order to implement the functionalities provided by the server. Those functionalities were defined according to the GIS requirements specification in Section 4.1.1. Appendix C lists the scripts created for each GIS functionality in PICGIS. Some of these scripts, such as the *setupLayout* one, need only be run once during the server initialization phase.

An ArcView project, named thesis.apr, was created as a container for PICGIS scripts and maps. This project has a special view, called Main View, created to contain all the coverage layers that can be accessed by the user during a certain session. PICGIS is not hard-

wired to any particular set of maps. Instead, it has access to all the maps in Main View. The developer/designer or system administrator can add/remove any number of coverages from Main View.

Since clients do not have direct access to the coverages available in Main View, a script called *getAllInfo* was developed to dump information about the current layers and their attributes into a text file, called allInfo.txt. This file is downloaded by the client as soon as communication with the server takes place.

The thesis.apr project also contains a layout, labeled Main Layout, which is directly linked to Main View. Main Layout automatically updates whenever changes take place in Main View. In addition, Main Layout also contains a scalebar and a legend that are also directly linked to the Main View active layers. As ArcView does not currently support direct exporting to graphical format files that are readable by a browser, Main Layout needs to be first exported to a postcript format file, then later converted to a gif file.

As for tables and databases, they are exported into either tab or comma delimited text files using appropriate scripts. With these simple format files, the client can easily access maps, associated information as well as underlying databases by downloading the generated text and gif formatted files supported by every browser.

5.1.2 The Messaging Server

The messaging server is developed using Java as a stand-alone application that should be continually running on the web server. The messaging server is always looping, waiting for clients to connect, and forwards messages to connected clients. The communication part is done using sockets because they provide a more reliable stream network connection than other networking methods such as datagrams. Such a stream communication

approach uses a connection-oriented protocol. When two sockets are connected, they can be used to send messages in either direction.

The server socket always listens to a TCP port for a connection from a client. When a client connects to that port, the *accept()* method of the server accepts the connection and creates another socket for the client to communicate through. The new socket is managed by a new thread created for this client. The server then goes back and listens to the original socket for other clients' connection requests.

As all the clients are represented by threads, the server keeps track of them through an array of *ConnectionThreads*. Forwarding messages to all clients becomes as easy as looping through all connected clients and sending the message through each client's output stream. The documentation of the Java server is provided in Appendix F.

The other mission of the server involves the forwarding of GIS commands from the clients to the ArcView server. As Java does not directly support Remote Procedure Calling, we found it convenient to write a C program that forwards the GIS commands to the ArcView server, hence taking advantage of the RPC libraries available for C. The function of the C program (Appendix F) is primarily to forward ArcView commands that are passed on by the messaging server.

The C program has another important role, namely that of converting the previously saved postcript layout file into gif format. A utility called *imconvert* available on Unix is used for this conversion. Although this transformation is necessary in order to make the file readable by commercial browsers, it is the most time consuming operation in this server implementation. Future versions of ArcView are expected to overcome this problem by providing built-in graphic exporting formats such as gif.

5.2 The Client

The client's code documentation can be found in Appendix E. The client is implemented as a Java applet, a program that can be embedded in a web page and hence is downloadable over the World Wide Web and executable within a web browser on the client's machine. Each client establishes a socket, similar to the standard unix socket, that handles the communication with the server.

A client connects when the user enters his/her username through the interface. This step is necessary for registering the user at the messaging server. Once connected, the client's first task is to retrieve the AllInfo.txt file and read in the layers and attributes names. This constitutes an essential part of the initialization of the client and the setup of the user interface.

The client components directly follow from the system architecture specification, hence consisting of a chat panel, an interface panel and a drawing panel. The display of returned gif formatted layouts or text tables occurs within separate frames of the same page. Each panel handles the user actions through the *action* method. Depending on the action, each panel constructs the appropriate message and sends it to the applet in order to be forwarded to the server. Similarly, whenever a message is received by the client, it is caught at the applet level and then forwarded to the appropriate panel's *processMsg* method, according to the message type.

The system setup, detailed message passing and processing, along with a description of the interface are presented in the next section.

5.3 Example Scenario

The functionality of the Internet-based collaborative GIS system can be better illustrated

through an example planning collaboration scenario. The framework, the system setup along with a typical conversation sequence are discussed below.

5.3.1 Scenario Framework

Planners often need tools that allow them to collaborate on a certain issue. In this case, we assume that two professional planners are using the system to analyze housing sales patterns in the Cambridge, Massachusetts area. For this purpose, the users might need the following coverages

- Cambbgrp: 1990 census data for Cambridge at the block group level.
- Sales89: residential sales for 1989 in Cambridge.
- Cambtigr: TIGER street files for Cambridge.
- Cambborder: Cambridge city boundaries.
- Cambzips: Cambridge subdivided by zip codes.

We will refer to the hypothetical planners as Planner A and Planner B. Planners A is located in Boston while planner B is located in New York and they decide to use PICGIS in order to get a better understanding of the project as well as of each other's points of view.

5.3.2 Server Setup

Most of the server setup steps have to be performed only once. The server should be running all the time on the web server. The steps consist of

1. Start ArcView 3.0 and open the thesis.apr project. As mentioned earlier, this project contains the scripts needed for the various GIS functions that are accessible to the clients.
2. Run the *RPCServer* Avenue script in order to initialize ArcView to accept Avenue requests through RPC protocol.
3. Create a new view, named Main View and add any themes needed for the particular session.

4. Run the *getAllInfo* script in order to dump the layers and attributes data into the AllInfo.txt file.

5. Run the *setupLayout* script in order to create a new ArcView layout directly linked to the Main View, with a scale bar, a legend and a north arrow. The layout's page setup is designed to conform to an A4 paper size in order to make it convenient for users to print and archive for future references.

6. Start the RPC C program and test whether connection is established.

7. Start the Java messaging server and test whether the connection between Java, C and ArcView is properly running.

The server is now ready to accept connection from the users.

5.3.3 Opening Screen

Figure 5.1 shows the opening screen of the project. The page is subdivided into four major parts:

- **The Control Panel**

  The control panel contains the interface to the GIS functions provided by PICGIS. The first row is a username input field where the user types in his/her username for the session. Next, the themes available are displayed. These are switches that turn any of the themes on and off. In the "select theme for Query" portion of the applet, the user selects a theme for database operations such as getting attribute information, retrieving the table or constructing a query. Finally, there is a "zoom choice" as well as a "zoom level" for the users to control the zooming extent. An "update layout" button has to be pressed whenever the user needs to refresh the screen to reflect the latest operations performed. The "edit layout" button is used in order to open a new drawing frame window where the user annotations can take place.

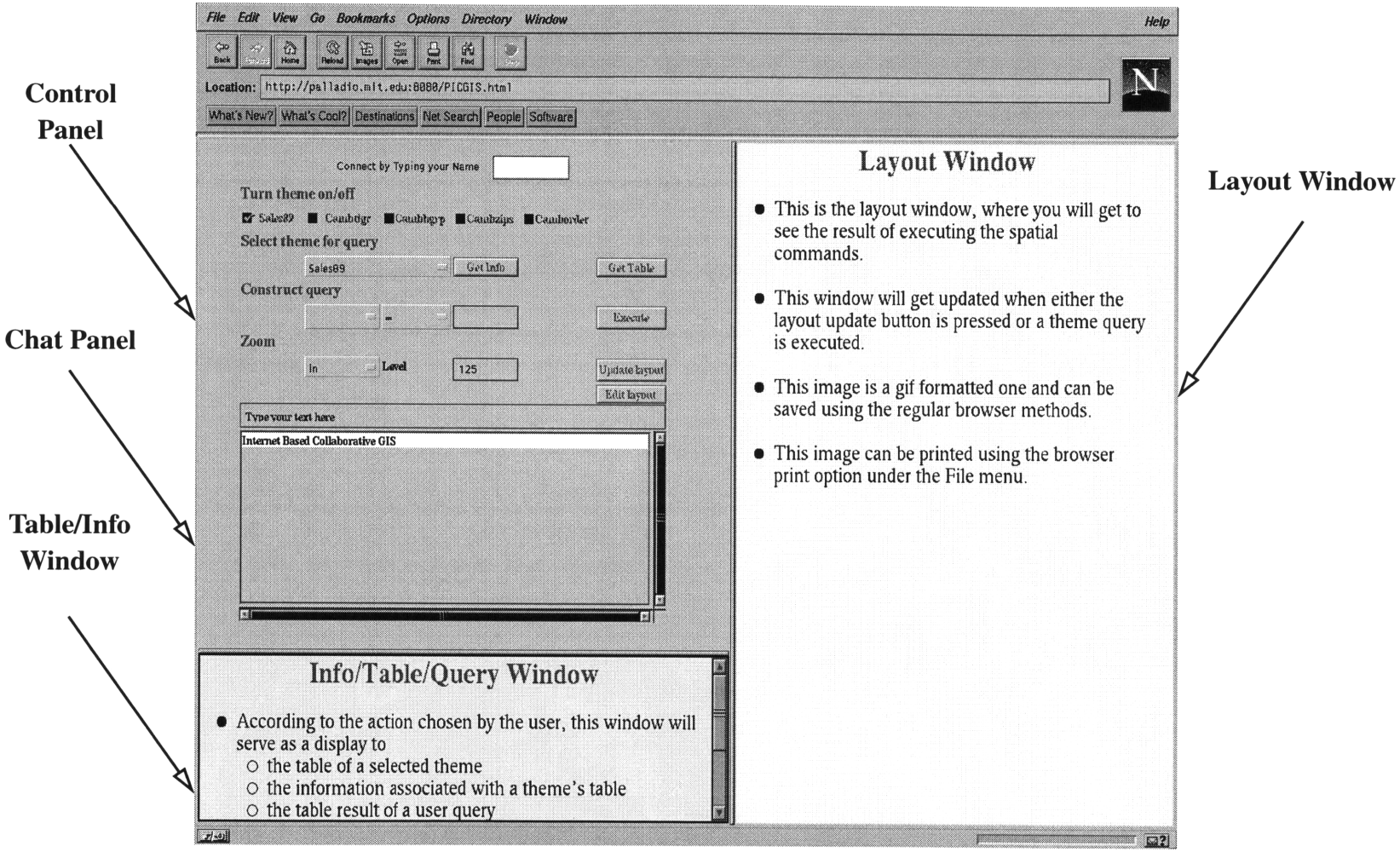

Figure 5.1: PICGIS's Opening Screen

- **The Chat Panel**

  The chat panel consists of two main parts: the message field, where the user can type his/her text, and the display area that is used for displaying all messages exchanged by the connected users.

- **The Table/Info Frame**

  This separate frame serves as a display window for the table returned by the server as a result of a table info, a table retrieval or table query request by one of the users

  The tables are either tab delimited or comma delimited text files that can be easily saved on the client's machine and opened using a spreadsheet application.

- **The Layout Frame**

  This separate frame is where the selected layers of maps get displayed along with their legend, scalebar and north arrow. As a result of a query, this window also highlights the selected features due to the query. The layout is a gif formatted image that can be saved on the client's local drive or directly printed. The A4 size image makes it convenient to print the image and save it for future meetings or for archiving purposes.

Until the user actually types in a name in the user name field, both the control panel and the chat panel are deactivated. When a user connects, the client applet starts by reading the AllInfo.txt file to initialize the on/off themes switches as well as their attributes in the "construct query" area. When both users connect, they can start the collaboration process.

5.3.4 Elementary Operations

Once the users are connected, a natural first step would be to understand the coverages provided in terms of their extent, and their underlying databases. In Figure 5.2, Planner A sends Planner B a message telling him/her that he would like to show him/her the sales and blockgroup data for Cambridge. Planner A then turns both Sales89 and Cambbgrp on

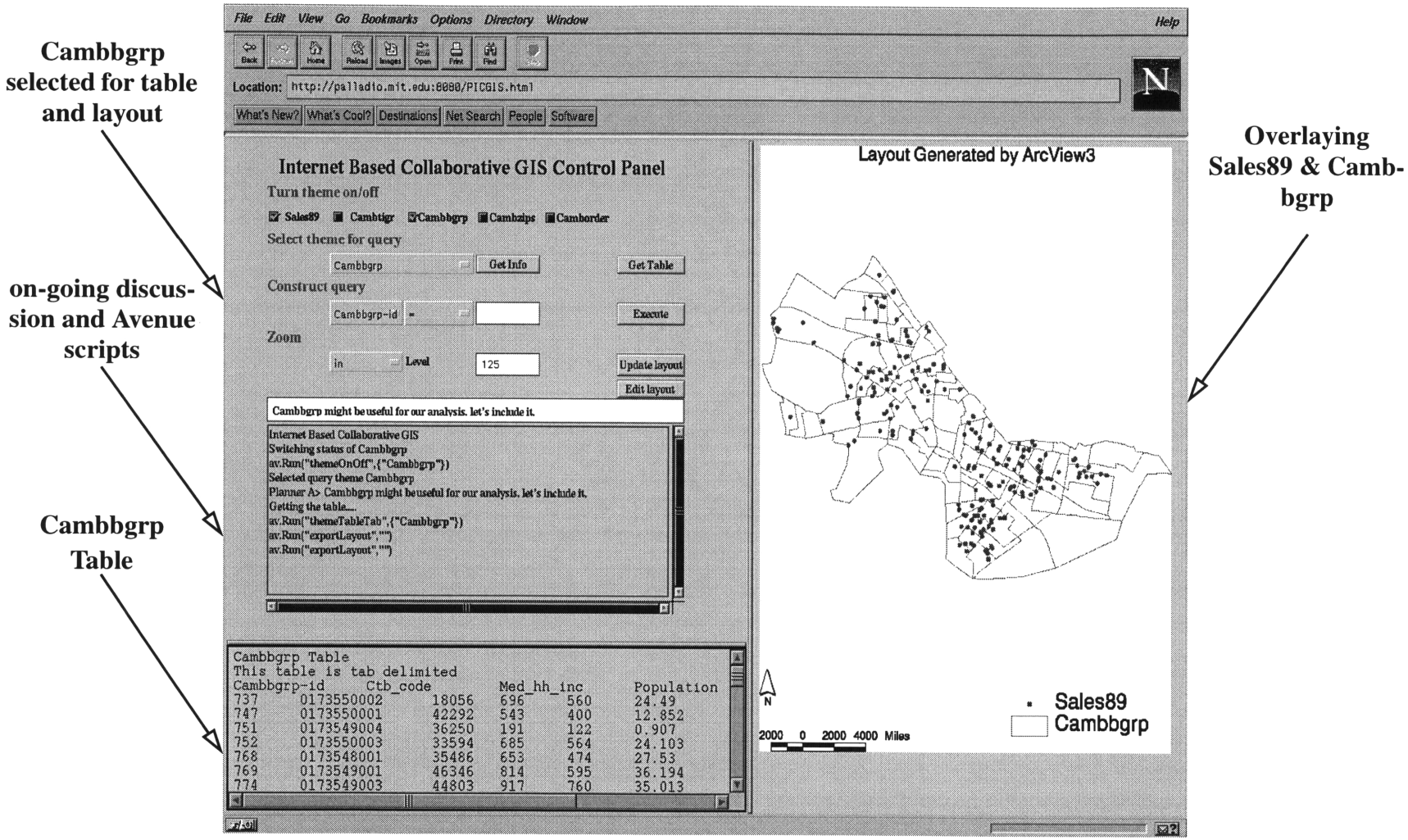

Figure 5.2: Displaying Cambbgrp Table and Layout.

and clicks on the "updateLayout" button [1]. This causes the map on the right of Figure 5.2 to appear on both planners' screens. At the same time, they both have the switches for Sales89 and Cambbgrp turned on. In order to better understand the data, Planner B also selects Cambbgrp and clicks on "GetTable", hence displaying the Cambbgrp table in the table window of both users.

A behind the scenes look at this process is necessary. The sequence of steps that takes place is as follows:

1. Switching a theme On
Planner A switches the Sales89 theme On.

- The action is caught by the *action* method of the control panel. In this method,
  - the *AVInterface* instance of the applet is updated to reflect that the current selected theme is Sales89.
  - the *action* method sends a call to a the method *themeOnOff*, where a Message of type THEME_ON_OFF is constructed and sent to the server. The message also contains the ArcView script that needs to be called on the server, retrieved from the *AVInterface* class.

- The server receives this message and decodes it to find out that this is a message that needs to be forwarded to the GIS server.
  - The message is first forwarded to the C program, which in turns forwards it to ArcView, which responds by running *themeOnOff* script and turns Sales89 on in Main View.
  - The server forwards the message to the other user

- Planner B's applet receives the message and decodes it to find that it needs to turn theme Sales89 on. It also updates its *AVInterface* to indicate that the current theme is Sales89.

2. Updating the Layout
Planner A clicks on the updateLayout button.

- The *action* method of user A control panel directs the action to an *updateLayout* method. There a message of type UPDATE_LAYOUT is constructed. The message

1. Note that Planner B has the option of turning any themes on and off, including those selected by Planner A.

also contains the ArcView script that needs to be run on the server.

• The server receives the message and forwards it to ArcView which runs the *exportLayout* script and saves the exported layout in a layout.ps file. The C program is then called to convert the ps file into a gif file. The server then forwards the message to the other user.

• Once a notification is received from the server, both clients download the layout.gif file and open it in the layout frame.

3. Retrieving the Table

Planner B chooses theme Cambbgrp and clicks the getTable button.

• A message THEME_QUERY_SELECTED is constructed by the *queryThemeSelected* method of the sender. This also updates *AVInterface* to indicate that the current active theme for querying is Cambbgrp.

• The server receives this message and decodes it as having to be forwarded only to the other users without passing it to the ArcView server.

• When the other user receives the message, it switches its selected theme to Cambbgrp and consequently updates its *AVInterface* class instance.

• When user B clicks on "getTable", a GET_TABLE message is generated. It is sent with the current active theme, which is retrieved from the *AVInterface* class.

• The server forwards the message to ArcView. The Avenue *themeTableTab* script is run in order to export the table of Cambbgrp into a tab delimited text file. The server also forwards the message to the users.

• Once the users, including the sender, get the GET_TABLE message, each client's applet uses the *showDocument* method to open the text file in the table frame.

5.3.5 Advanced Operations

Under advanced operations, we list the database querying of a selected theme as well as the opening and usage of the drawing frame.

• Theme Querying

A similar messaging scheme is used for querying the current active theme. When a user selects the attribute, the operation and the query expression value, these values are updated in the user's *AVInterface* class and then sent to the server to be forwarded to the other clients. Other clients then update their *AVInterface* and update their control panel to reflect the new changes.

Figure 5.3 shows the user screens when one of the users executes the query for all blockgroups with population greater than 1000. When the query is executed, both GET_TABLE and UPDATE_LAYOUT messages are generated. The table frame shows only the selected records, while the new layout is updated highlighting the selected blockgroups.

• Drawing

Once the results of the query are returned, the users decide to edit the latest version of the layout, perhaps to add some annotations or to visually explain certain things. This is when one of the users presses the "editLayout" button. When either user presses the button, a new drawing frame appears on the users' screens along with basic drawing tools as indicated in Figure 5.4. Each user is assigned a unique color, to enable users to identify who is drawing what on the screen.

With the available tools, the users can draw circles, rectangles, lines and textual annotation on top of the latest layer displayed in the layout frame. In Figure 5.4, Planner A draws the circle shown and indicates that this should be considered as an area of interest for further progress. PICGIS allows Planner B to select the same circle and move it to a different location, an operation which will also be visible at Planner A's screen. Using such a setup, both planners can create objects, annotate them and manipulate both their own and their partners' objects, creating an easy-to-use and effective visual collaboration scheme.

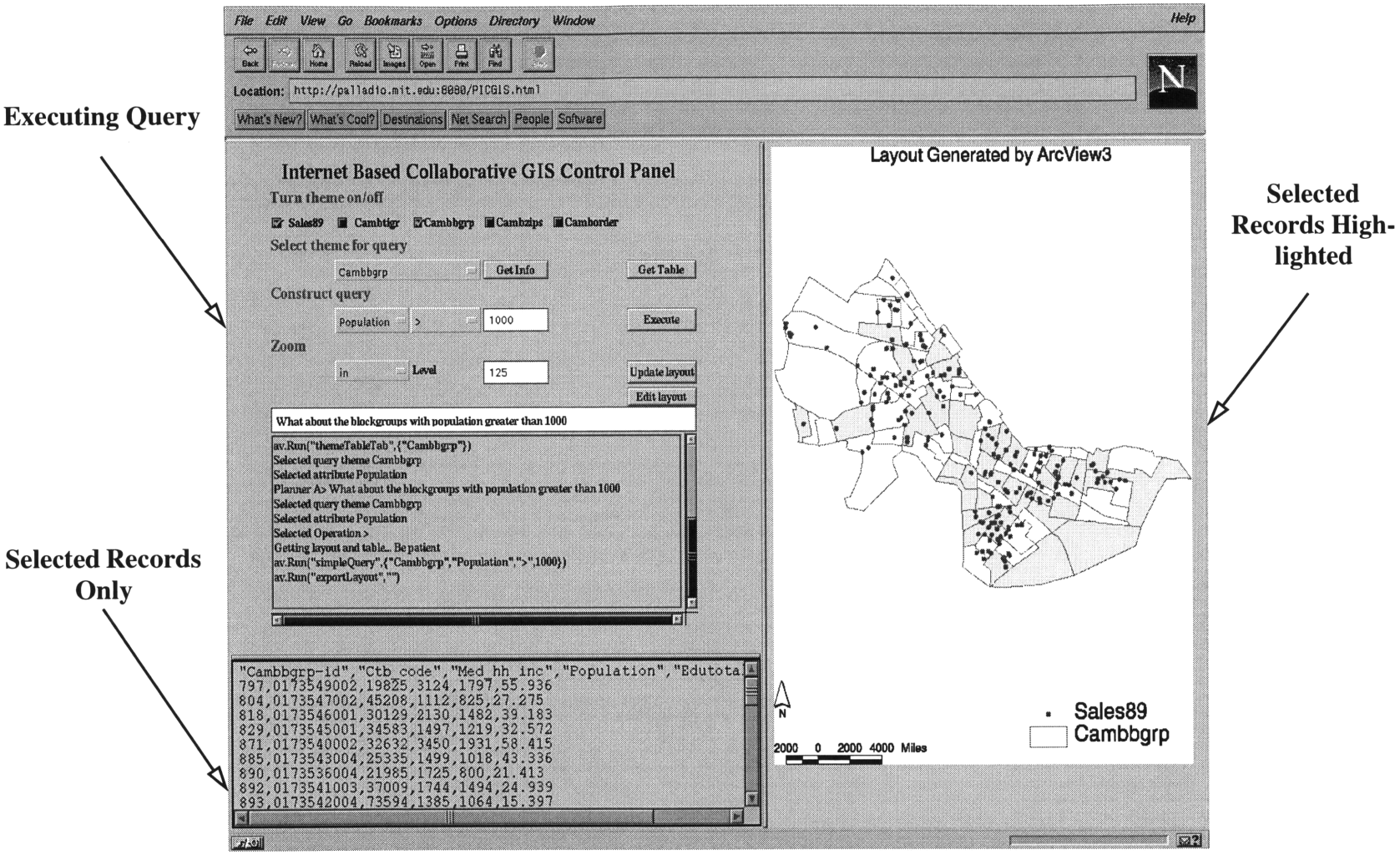

Figure 5.3: Executing a Query.

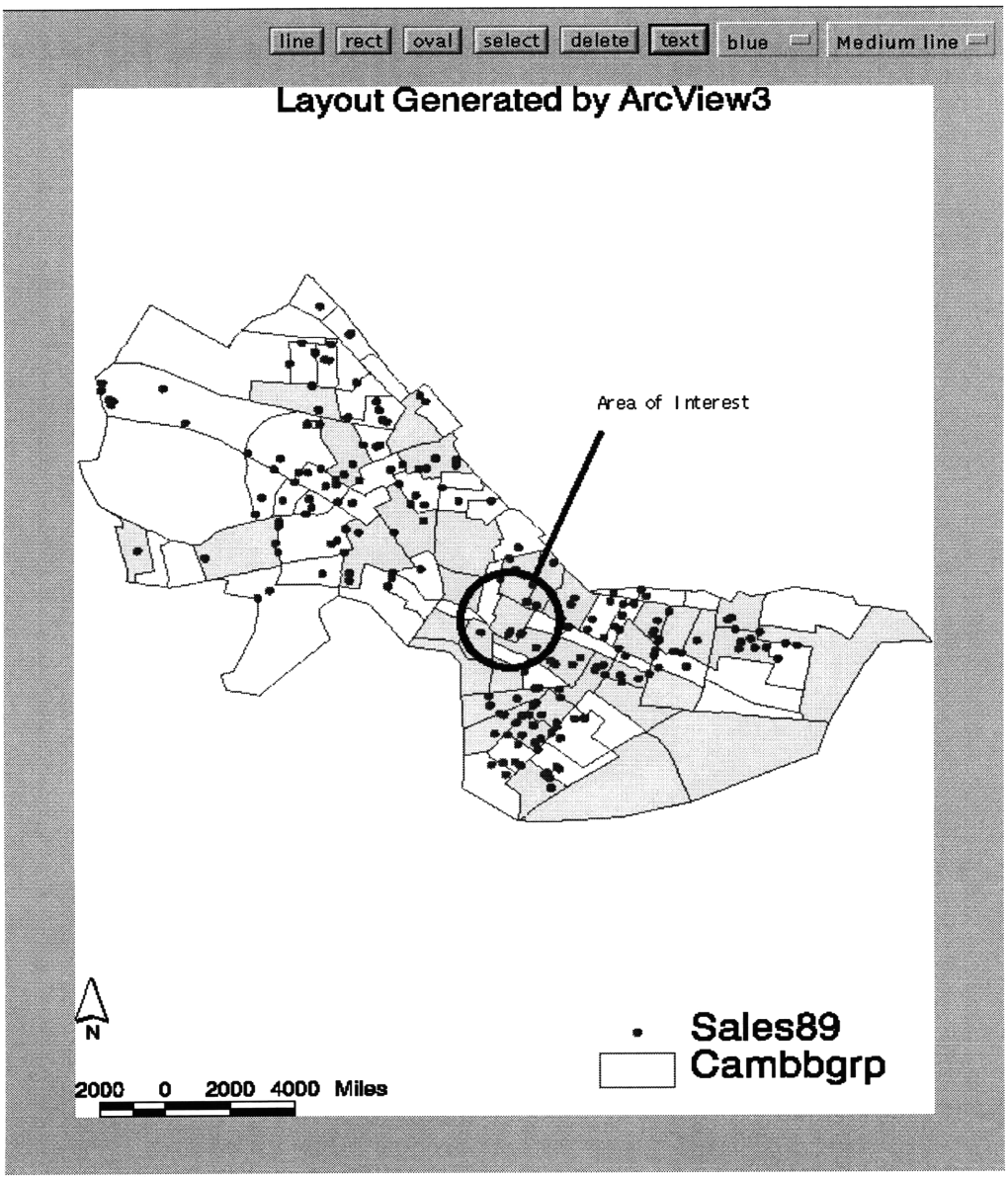

Figure 5.4: Drawing Frame.

In terms of message passing in the drawing frame, it follows a pattern similar to the one outlined in the discussion about elementary operations. The only difference is in the type of the messages sent, which in this case, is the DRAWING type (Refer to Appendix E.0.8).

5.4 Concluding Remarks

The proposed Internet-based collaborative geographic information system we described thus far, provides a useful collaboration forum for geographically distant planners, as well as students/teachers, trainers/trainees and customers/customer support staff. In this section, we assess the value of the system, and discuss the functional limitations of the prototype.

5.4.1 The Value of the Prototype

PICGIS demonstrated the viability of combining the three fields of collaboration, GIS and the Internet. The example scenario (with Planners A and B) suggests that the prototype can enhance the way they go about their planning work. This is achieved by helping both planners understand the other's point of view, as they were able to illustrate their ideas, in addition to seek and incorporate each other's opinions, on-line.

Planning is a diverse field that combines many activities, and therefore an associated project needs the expertise of several individuals, such as policy makers, environmental engineers and economists. Collaboration in such environments is necessary to ensure that all the aspects of a certain planning problem are covered. The major value of the system here is its provision of a meeting place for these often geographically dispersed experts to meet and negotiate in real time, with some capacity to discuss and manipulate graphics that have specific types of spatial content.

Although there are many Groupware products on the market, none of them specifically targets mapping related applications. As GIS moves towards being a very integral part of these applications, especially in the project preparation, analysis and presentation stages, it becomes necessary to have a groupware system that can directly deal with GIS data. Our

prototype attempts to meet those needs through a system that links experts together, while providing them with basic GIS functionalities.

The system uses the Internet as the medium for achieving the above goals. Consequently, a user does not have to go through any expensive setup or installation phase before "meeting" other users. All that is needed is a Java- enabled browser, currently available on most computers. Consequently, the prototype is easily accessible and easy to use.

Despite the valuable features of the prototype, it still has some functional limitations regarding the outlined goals. These limitations are discussed in the next section.

5.4.2 Limitations

Currently, PICGIS suffers from several limitations. Fortunately, most of these limitations can be overcome in future versions of the prototype. The free collaboration setup is the first of such limitations. With such a setup, it is hard to control the sequence of actions performed by a user. This is particularly true due to the Internet-based nature of the system, whereby theoretically, any client on the web can access the prototype and make any changes. Having said that, the free setup works adequately well when we have a small number of serious participants using the system.

The prototype is also limited at the editing/drawing level. Currently, this component is not directly linked to the GIS engine. Furthermore, users have only one drawing frame per session, preventing them from displaying and comparing several coverages/scenarios at once. In addition, the current setup lacks a way for directly saving maps (with the added annotations), in any format, for later use or reference.

The system also lacks mouse coordination. Each user can only see their own mouse without any indication of other users' mouse locations. The ability to keep track of all users' mouse locations is useful when the mouse is used to point at interesting features on

the map. The most recent version of Java, however, contains ways to overcome this problem by allowing Java applets to control the mouse location on the screen.

Finally, as a server-based system, PICGIS's performance is limited by the speed of the network and the processing power of the server, not to mention the time delay caused by the extra step of converting the map from postcript to gif (see Section 5.1.2).

Chapter 6

Conclusion

6.1 Evaluation of PICGIS

In light of our learning experience with PICGIS, we are now in a position to evaluate the concepts behind the prototype. Evaluating PICGIS is necessary because the prototype can be viewed as a valuable market research tool. Testing PICGIS with real life users can help point out lacking features as well as identify unnecessary ones. The evaluation is also important considering PICGIS can be used as a seed for the development of more advanced collaborative GIS products.

In this section, we start by presenting PICGIS as one prototype in a pool of collaborative GIS systems, and summarize our findings and experience.

6.1.1 PICGIS and Collaborative GIS

PICGIS was developed because there is a clear lack of collaborative GIS solutions that go beyond simple and limited approaches, such as Hypernews and screen dumping. However, before discussing the position of PICGIS in this field, we first note that, currently, the concept of collaborative GIS is not well defined. It is perceived simply as a system that can meet the GIS and collaboration needs of geographically dispersed users with often diverse backgrounds. The development of PICGIS gave us the opportunity to better understand

the concept of GIS collaboration, and to study the feasibility of implementing that concept in an easily accessible and integrated fashion.

In addition, for any given interpretation of collaborative GIS, there exists a variety of ways to implement that interpretation. This is primarily due to the continuing rapid advances in Information Technology. The right approach for any particular problem is highly dependent on the underlying application's goals, operating environment and targeted users. In Chapter 3 for example, we saw that there are several possible paths to be followed for each component of the system. While all these paths are feasible and applicable, they differ in their resulting trade-offs. In this context, it is therefore important to note that PICGIS represents just one approach for solving the open problem of collaborative GIS.

The GIS element of collaborative GIS strongly differentiates collaborative GIS from other groupware systems. A typical GIS system does not only specialize in the processing and display of spatial data, it also deals with underlying descriptive information associated with the data (refer to Section 2.2.2). Hence, specialized GIS tools are needed for visualization and analysis of GIS data. It is therefore important for a collaborative GIS system to be able to handle such data types, while providing a forum for collaboration among its users.

In this context, PICGIS demonstrated the viability of a collaborative GIS product. In the next section, we summarize PICGIS based on our experience, and present our findings regarding the trade-offs involved in its design and implementation.

6.1.2 Summary of PICGIS

In this thesis, we successfully achieved our goal of designing and developing PICGIS, a Prototype Internet-based Collaborative Geographic Information System. PICGIS provides

basic GIS functionalities to a group of geographically dispersed individuals, linking them through an on-line collaboration forum, whereby each user's actions are propagated to all other participants.

The design of PICGIS proved to be a complex task, as many heterogeneous software components needed to be integrated together. Although the design and development stages of PICGIS were difficult, benefits were reaped in the form of easy-to-use, flexible, platform independent system with state-of-the-art basic GIS functionalities.

PICGIS is a demonstration of the viability of combining the fields of GIS and collaboration via the Internet. As discussed earlier, the system does not require any software installation and is interoperable due to its underlying Java implementation. In addition, it supports all GIS data formats provided by ArcView 3.0. The system is targeted to be used by students, professionals and planners, in a GIS training or professional planning setting.

PICGIS is a distributed object-oriented system that uses the server-based approach (refer to Section 3.2) for its GIS component. The decision of following a server-based architecture was primarily motivated by the advantages of using a powerful and dedicated GIS engine on the server. In the specific case of PICGIS, ArcView provided us with a large range of supported data types, a wide set of GIS functionalities, and a way to access these remotely. In addition, as a result of the server-based approach, a desirable thin client implementation was also achieved.

On the other hand, the GIS server-based architecture imposed some limitations and unnecessary complexities on the prototype. For example, as a result of the limited exporting capabilities of ArcView, a theoretically unnecessary C module for converting postcript formatted maps to a gif format (refer to section 5.1.2) was developed to compensate for this limitation. This module ended up introducing the largest time delay in the system. In addition, with such a server-based architecture, the design of the overall system had to be

adjusted according to the GIS server element. In our case for example, the mere introduction of ArcView as a GIS server, imposed the use of RPC as the communication protocol. Consequently, the C module mentioned earlier was also used to compensate for Java's inability to directly support RPC. In addition, the performance of the whole system was affected by our choice of ArcView as a server, as ArcView cannot process more than one request at a time, causing queuing delays, especially with a large number of users. This limitation can be improved with a GIS server that follows a multithreaded architecture. It is important to note that these limitations are not specific to ArcView. In fact, in any server-based system, the design and implementation of the system will be constrained by the GIS server used, as it is the cornerstone of that system.

However, the above discussion does not imply that we should eliminate the server-based approach as an option for collaborative GIS. The choice of a suitable architecture depends on the system's potential users, in light of the trade-offs discussed in Chapter 3 and the above discussion. We believe that the benefits of a server-based architecture outweigh the limitations. A client-based or plug-in method, for example, would have required a high client horsepower and a long download time, especially with the typically large size of GIS data. A particularly attractive advantage of a server-based approach, that came to light from our experience, is that the data is transmitted in gif and txt formats, which are much smaller, in size, than the original coverages. We believe that the server-based approach suits today's average user, because a typical user's desktop has less than adequate processing power to handle GIS operations locally, given the size and complexity of underlying GIS layers. This indeed is the approach adopted in products such as the ArcView Internet Map Server, which rasterizes the data before sending it to the client. This consequently only requires the client to handle the User Interface elements.

In the short run, the market will converge to a balance between a client-based and a server-based architecture. The resulting architecture will be of a hybrid nature. Some functionalities will be assigned to the clients, such as those related to the graphical processing of GIS data, and other more complex tasks, such as extensive spatial analysis, will be assigned to the server. In the long run, clients will be equipped with local intelligence that will enable them to selectively fetch (according to their needs) subsets of GIS data, stored in spatially indexed databases on the server. In turn, the fetched data can be used by a middleware tool that is tuned to a particular application, such as collaboration.

As for our experience with the collaboration aspect of collaborative GIS, although the server-based setup allowed us to establish a synchronous way for collaboration, this configuration suffered from certain performance limitations. The most visible one was the interface and status inconsistencies between clients. Since clients are not equidistant from the server, a message broadcasted from the server is received at different times at different machines, sometimes causing the inconsistencies mentioned above.

Despite its drawbacks, the server-based architecture of both components (GIS and collaboration), paid off by producing a system, that is scalable in terms of the number of system users and offered GIS functionalities. As a result of the scalability feature, PICGIS can be easily expanded in many directions, such as those suggested in the next section.

6.2 Future Work

This section focuses on a number of potential extensions to the Prototype Internet-based Collaborative Geographic Information System. The following suggestions for future work directions are not only limited to the prototype we built, but are rather reflections of our ideas regarding collaborative GIS features in general, based on the prototype experience. Each of the following sections describe a different category for future work.

6.2.1 Adding Advanced Functionalities

The value of collaborative GIS can be greatly enhanced by the addition of tools for saving and/or recording a session. This feature can be achieved by either saving all generated layouts or by saving the sequence of instructions that led to the current layout. If the above feature is added, it will become more convenient for the users to save a session in order to retrieve it later for further analysis, or to use it as the basis for a new session. This feature might also appeal to the users because it allows at least one of them to use the system for preparing the coverages that will be discussed in the "next meeting". However, this feature contradicts our earlier assumption that groups use the system for collaboration and simple analysis of maps prepared in advance, using more sophisticated GIS packages.

Therefore, this raises the question of whether a collaborative GIS system can or should be considered as a substitute for a stand-alone GIS system. Since we believe that the purposes of those systems differ, we argue that a collaborative GIS should not be thought of as an alternative for a complete GIS. We believe that collaborative GIS is more useful to a larger audience if it is instead used for discussion and collaboration purposes primarily, providing access to a reasonably simplified set of GIS tools. Loading the client with additional GIS functionalities implies higher client horsepower or excessive network communication. Neither option will be appreciated by the users who intend to use the system primarily for collaboration with a diverse set of colleagues.

Other advanced features include tools that allow users to add new GIS layers. In a server-based setup, these new layers are saved on the server to be used along other available layers. However, many database management issues will consequently arise regarding keeping track of all the layers, new and old.

6.2.2 Controlling the Collaboration Process

Another direction for future research involves increasing group efficiency by introducing elements to manage the collaboration process, and to control the work flow. The system can be upgraded to use any of the tested collaboration schemes, including [16]

- The democratic setup, where the choice of the current active user is determined according to a vote by the current participants.

- The chalk passing setup, where the most recent active user chooses the next person to take control.

- The chairman control setup, where one person is designated to moderate the current session.

In addition, the concept of chat rooms can be introduced to our system, whereby the server handles several distinct groups of users, with each group working on a separate project, in its own session workspace.

Thus far, we have only discussed the synchronous collaboration setting. However, this discussion about collaboration will not be complete without mentioning the equally important asynchronous collaboration approach. Such a scheme is useful in cases where users need to store complete or intermediate maps for other users to refer to at their own convenience.

Users following the asynchronous scheme can, for example, use standard GIS packages for off-line analysis, save their results in a convenient format (such as the Portable Document Format) and post them on a HyperNews[1] driven site specifically setup for the project at hand. This scheme is attractive, since in this example, HyperNews automatically handles the organization of posted messages, comments and replies by creating a hierarchical easy-to-use structure for these messages.

---

1. Refer to http://union.ncsa.uiuc.edu/HyperNews/get/hypernews.html for more information.

Although the Portable Document Format (pdf) offers the user some control in viewing, panning and zooming in/out of the maps, it cannot be inverted back to its original GIS format, and does not provide a reference to its underlying database. In addition, this HyperNews-based asynchronous method does not provide a real time conversation forum or an on-line collaborative analysis capability. A collaborative system, such as the one we have described in this thesis, can add value to the collaboration process by virtue of integrating the collaboration and GIS elements. Its value can be further enhanced by adopting some of the asynchronous approach characteristics. For example, PICGIS can be modified to allow the edit window to be saved and posted to a HyperNews site, in a pdf or gif format.

6.2.3 Investigating New Technologies

In light of the on-going advances in the Internet, GIS and collaboration fields, the system's components can be replaced or upgraded to incorporate these new technologies. For example in the case of PICGIS, we can replace our GIS component with the soon-to-be-released ArcView Internet Map Server [8]. This allows us to compare the performance and functionalities of both approaches. The wider choice of alternatives provides the developer of a collaborative system with more flexibility. It also presents the developer with the challenge to find the optimal combination of elements for a particular application.

Recent trends in both the GIS and collaborative systems markets will be discussed in further detail in Section 6.3.

6.2.4 Investigating Other Alternatives

As previously noted, PICGIS is just one implementation approach for a collaborative GIS prototype. In light of our updated knowledge, we feel it is necessary to revisit other solutions to the collaborative GIS problem. Unlike the approaches described so far, the ones below are client-based.

For example, the recent availability of VRML (Virtual Reality Modeling Language) [34] is an alternative that has not yet been explored for bringing GIS vector data onto the web. VRML is a scene description language, that can describe three dimensional environments over the Net, in terms of geometric models of spatial objects capable of accommodating some GIS elements. VRML web-based GIS has not entered the market yet for a couple of reasons. First, the GIS data model is different from the VRML one, primarily because VRLM currently focuses on three dimensional geometry representations. In addition, GIS operations require a sophisticated database oriented model, that is not yet available for VRML. Consequently, it will take some time before VRML can successfully handle GIS data in an interoperable fashion.

In addition, this client-based solution will be burdensome for the clients due to the typically large size of GIS data, and the complex spatial operations that will, as a result, be embedded in the client. A similar client-based alternative entails the use of a plug-in for vector data, as discussed in Section 3.2.2. However, due to the unavailability of a standard for vector data, it might be a while before these solutions become interoperable in the full sense of the word.

With the current high rate of technological advances, it becomes difficult to enumerate all the possible alternatives for this problem. One way around this difficulty is to examine the commercial market trends, as discussed in the next section.

6.3 The Status of Collaborative GIS

With the escalating rate of technological development, the increasing awareness of professionals to the benefits of collaboration, and the expanding role of the Internet, it is necessary to take a look at the status of the collaborative GIS market and its trends. This might help in determining whether the collaborative GIS market will soon experience the

emergence of specialized collaborative GIS products. Such efforts would normally be initiated by big GIS and Groupware companies. However, the current availability of authoring environments for the Internet, and the likely modularity of a collaborative GIS system, allow smaller players to enter the competition for collaborative GIS systems. Since collaborative GIS depends on the fields of GIS and collaboration, we examine the status of each of these fields separately.

6.3.1 The GIS Aspect

GIS is currently being pushed in two separate directions:

1. The move to the Internet, fueled by the growing need for dynamic maps through this medium.
2. The integration with other Information Systems, fueled by the growing need for an integrated working environment, and the availability of standard Relational Database Management Systems methods for handling persistent data in a multi-user transaction-oriented environment.

Regarding web-based GIS, it seems that it will be long before complete GIS systems start appearing on the web, because of the limitations discussed in Section 3.2. Hence, we expect that, in the short term, web-GIS will provide GIS functionalities in limited elements, in the same way desktop mapping tools provide address matching and thematic mapping.This phenomenon is further perpetuated by the efforts of the large GIS players to popularize the server-based approach, so that customers will not completely abandon their stand-alone packages in favor of a web-based one.

Although these players are trying to replicate the GIS interface on the web, they are not providing the complete set of GIS functionalities in their web versions. The reason behind this is that web GIS was not created to replace GIS, but rather to take advantage of the Internet to reach a larger number of customers. The advantage of web-based GIS is its ability to publish raw GIS data, and its provision of some functionalities to explore this

data. Advanced analysis still requires a full set of GIS functionalities provided by stand-alone packages.

Since the big GIS players are pushing towards a server-based approach, it follows that a considerable effort is being directed towards building more powerful GIS engines, such as the Spatial Database Engine (SDE), with an inherently better model for multi-user access. The benefits of such servers include a faster processing time and optimal data storage and access techniques, both critical in the case of GIS data processing.

In terms of collaboration, there are no efforts by the big GIS players towards achieving that goal. However, the current trend is to integrate GIS with information systems on the desktop. Several plug-and-play GIS products have entered the market, providing easy-to-use environments that allow developers to add mapping components to existing applications, build data viewing applications or create specialized mapping programs. The integration movement is motivated by the increasing use of GIS by IT managers, as an analysis and presentation tool. Accordingly, we expect that collaboration efforts will start emerging through applications that use GIS as only one of their many components.

6.3.2 The Collaboration Aspect

With the growing size of today's organizations and consequently the increasing demand for collaboration, there is a better appreciation of the value of collaborative systems. With this appreciation, it is expected that there will be a growing number of such systems in the next few years. The current competition between groupware companies is primarily driven by large organizations and is expected to rapidly move to the Intranet field.

As for the GIS incorporation in collaborative systems, it is likely that this trend will appear in workgroup office systems. As geographical tools are increasingly used to facili-

tate web navigation, we expect to see initial collaborative GIS efforts in the form of basic GIS elements used in an interactive setting. This trend, however, will be independent of the GIS server development trend, mentioned earlier.

In light of this discussion and the inevitability of continuing technological changes, there is every reason to believe that it will be many years before the collaborative GIS market settles down. Hence, it is not clear when the supply of collaborative GIS tools will catch up with the increasing demand.

Appendix A

State of the Art Web Based GIS

A.1 Tools for Web Based GIS Construction

- GeoMedia Web Map (Intergraph)
- MAPublisher(Avenza software, Burlington, Ontario, Canada)
- Environmental Systems Research Institute MapObject (with the Internet Server extension)
- MapQuest (GeoSystems Global Corp., Lancaster, Pa)
- WebMapServer (Etak's Etak Guide)
- MapInfo Pro-Server (MapInfo Corp., Troy, N.Y.)
- Spatial Query Server (Vision International)
- Spatial Web Broker (Genasys II Inc.), Fort Collins, Colorado)

A.2 Examples of Web Mapping Sites

- Argonne National Laboratory SmallTalk/Java GIS. URL: http://www.dis.anl.gov:8001/GIS/stgis.html.
- Argus Technologies "MapGuide". URL: http://www.argusmap.com/.
- BigBook, San Francisco. URL: http://www.bigbook.com.
- Canadian National Atlas Information Service: Make a map with NAISMap. URL: http://ellesmere.ccm.emr.ca/wnaismap/naismap.html.
- CARIS Internet Server. URL: http://caris0.universal.ca/demo/login/index.html.
- ESRI's homepage. URL: http://www.esri.com/.
- ESRI Internet Mapping Solutions. URL: http://maps.esri.com.
- Etak's Etak Guide. URL: http://www.etak.com/.
- Genasys II Inc., Fort Collins, Colo. Spatial Web Broker 96.URL: http://www.genasys.com/homepage/products/sample.html.
- GeoSystems Global Corp., Lancaster, Pa. MapQuest. URL: http://www.tripquest.com/.
- MapInfo Corp., Troy, N.Y. URL: http://www.mapinfo.com/.
- Michigan State University. URL: http://www.ssc.msu.edu/~geo/wwwgis.html.
- On-line Map Creation with Generic Mapping Tools. URL: http://www.geomar.de/personal/mwein/omc_intro.html.
- Spatial Query Server developed by Vision International. URL: http://www.autometric.com/AUTO/html/vision.html.
- Vicinity Corp., Sunnyvale, Calif. URL: http://www.mapblast.com.
- MapViewer. Xerox PARC. URL: http://mapweb.parc.xerox.com/map.
- Geographic Data on the Southeastern US. US Environmental Protection Agency. URL: http://www.epa.gov/region4/gispgs/reg4gis.html

• NAISMAP. Canadian National Atlas Information Services (NAIS). URL: http://www-nais.ccm.emr.ca/naismap/nais/naismap.html.

• Virtual Tourist. Kinesava Geographics. URL: http://www.vtourist.com.

• Real Estate Applications

1. Homebuyer's Fair. URL: http://www.homefair.com.
2. Cyberhomes. URL: http://www.cyberhomes.com.
3. Owners Network. URL: http://www.owners.com.

Appendix B

XWatchWin Man Page

NAME

xwatchwin - watch a window on another X server

SYNOPSIS

xwatchwin [-v] [-u UpdateTime] DisplayName { -w WindowID |WindowName }

DESCRIPTION

xwatchwin allows you to peek at a window on another X server. To use it, you must specify the display name of the machine you want to watch, then the name of the window on that machine. Xwatchwin will attempt to connect with the X server hostname:0.0, and if successful, will try to retrieve a copy of the window in which you specified interest.

You may specify the window you want to watch either by name or by its window id, usually a hexidecimal number. Usually specifying the window by name is simpler, although not all windows have names associated with them; in that case you must use the window id option.

If the window you want to watch is not in a viewable state, xwatchwin will tell you so and exit. If while you are watching a window it becomes 'unviewable', xwatchwin will wait until the window becomes 'viewable' again.

xwatchwin was written as an aid to a class for people learning to use X. The idea is that the instructor would type into an xterm window on his/her display and the students would use xwatchwin to see what the instructor typed. The students could then type the same thing in their own terminal windows. Hopefully others will find equally (if not more) constructive uses.

OPTIONS

-u updatetime

This option specifies how often (in seconds) you want to get a new copy of the window you're watching. It is in effect a 'sample rate'. By default, xwatchwin updates your copy of the window as often as it can. The time it takes to actually do the update is dependent on the speed of the X server on both machines, the speed of the intervening network, and other factors.

-w windowID

This option specifies the window you want to watch by number, for example, "0x50000b". Use the xlswins(1) command to get a list of window id's and possibly their names on the remote server.
You must specify a window to watch either by name or by id. Specifying a window to watch by name is usually easier if you know what you're looking for.

EXAMPLES

If there is an X server on the remote machine "crow" and if on that server there is a window called "X Terminal Emulator", you can watch that window by typing

xwatchwin crow X Terminal Emulator

If there is a window on "crow" that has no name but has a window id of "0x50000b", you can watch it by typing

xwatchwin crow -w 0x50000b

If you want to get new copies of a window only every 30 seconds, you can do so by typing

xwatchwin crow -u 30 -w 0x50000b

SEE ALSO

xlswins(1), xwininfo(1), xdpyinfo(1),

BUGS

xwatchwin doesn't support the -display option. You must set the display on which the xwatchwin window is created by changing your DISPLAY environment variable.

If the window you're watching is resized while xwatchwin is getting a new copy of that window, the program will crash. The smaller your update interval, the more likely you are to experience this bug (although it hasn't happened all that often to me).

xwatchwin can now deal with two displays of different depths. There is special-case code for the conversions between 1-bit displays and 8-bit displays (either direction) which may garble the image on some machines. The general case code should work on anything, albeit somewhat more slowly. One note: ABSOLUTELY no attempt is made to make the colors match up. If you're on a 5-bit display, and you're monitoring someone elses 8-bit display, the conversion just takes his 8 bits and chops the top 3 bits off, and puts it on the screen. Maybe in the next version...

COPYRIGHTS

Copyright 1992 - 1995, Q. Alex Zhao
Copyright 1989, George D. Drapeau

AUTHORS

Light-weight version by Q. Alex Zhao azhao@cc.gatech.edu.
Display depth conversion code added by John Bradley
bradley@cis.upenn.edu.
Original version by George D. Drapeau, Stanford University,
Academic Information Resources / Systems Development, drapeau@jessica.stanford.edu.

Appendix C

Avenue Server Scripts Used

The following Avenue scripts were used on the ArcView 3.0 RPC server running on the server component of the system. The references for this section consist of ArcView 3.0 Avenue on-line help/scripts and the Avenue user's guide [2].

C.0.1 RPCServer

```
' ****************************************************************
' Script Name : RPCServer
' Script Descritption: This script starts the RPC server with program
' number 0x4000001 and version number 1
' Script Inputs: none
' Comments: has to be the first script to be run on the server
'****************************************************************

RPCServer.Stop
RPCServer.Start(0x4000001,1)
```

C.0.2 testCommands

```
' ****************************************************************
' Script Name : testCommands
' Script Description: This script was used to incrementally test the other
' scripts of the project
' Scripts Inputs: none
'****************************************************************

' testing turning on and off themes
' result = av.Run("themeOnOff",{"Cambzips"})

' testing zooming in and out and to extent
' result1 = av.Run("zoom",{"in",125})
' result2 = av.Run("zoom",{"out",125})
' result3 = av.Run("zoomtoExtent",nil)
```

```
' testing getting attribute info about a theme
' result4 = av.Run("themeAttr",{"Cambzips"})
' msgbox.info(result4,"")

'testing exporting the theme table to a txt file
'result5 = av.Run("themeTableComma",{"Cambbgrp",FALSE})
'result6 = av.Run("themeTableTab",{"Cambbgrp"})

' testing setting up and exporting the layout
'av.Run("setupLayout","")
'av.Run("exportLayout","")

' testing the creation of simple queries with strings and
' numbers as arguments
av.Run("simpleQuery",{"Cambtigr","Rt","=",1})
av.Run("simpleQuery",{"Cambtigr","Fname","=","Alewife Brook"})
```

C.0.3 themeOnOff

```
'*****************************************************************
' Script Name: themeOnOff
' Script Description: This script turns a theme on if it is off and vice versa.
' Script Inputs: theme name as String
'*****************************************************************

' Checking for the correct number of parameters
if (self.Is(List).Not) then
  return nil
elseif (self.count < 1) then
  return nil
else
  themeString = self.Get(0)

  ' Get the Main View and activate it
  theView = av.GetProject.FindDoc("Main View")
  theView.GetWin.Activate

  ' find the theme
  theTheme = theView.FindTheme(themeString)
```

```
  ' if the theme is on, set if off
  if (theTheme.isVisible) then
    theTheme.setVisible(false)
    theTheme.setActive(false)
  else
    ' if the them is off, set it on
    theTheme.setVisible(true)
    theTheme.setActive(true)
  end
  ' update the layout
  av.Run("exportLayout",nil)
  return 1
end
```

C.0.4 zoom

```
' ***************************************************************
' Script Name : zoom
' Script Description: This script allows the user to zoom in and
' out the Main View
' Script Inputs: 1- in/out (for zooming in or out), 2- the zoom level as integer
'***************************************************************

' Checking for the correct number of parameters
if (self.Is(List).Not) then
  return nil
elseif (self.count < 1) then
  return nil
else
  zoomNature = self.Get(0)
  zoomLevel = self.Get(1)

  ' Get the Main View and activate it
  theView=av.GetProject.FindDoc("Main View")
  theView.GetWin.Activate

  if (zoomNature = "in") then
    theView.GetDisplay.zoomIn(zoomLevel)
```

```
  else
    theView.GetDisplay.zoomOut(zoomLevel)
  end
  ' update the Layout
  av.Run("exportLayout",nil)
  return 1
end
```

C.0.5 zoomtoExtent

```
' ********************************************************************
' Script Name: zoomtoExtent
' Script Descritption: This script zooms the Main View to the full extent of
' active themes
' Script Inputs: none
'*********************************************************************

' Get the Main View and activate it
theView = av.GetProject.FindDoc("Main View")

' Get the extent of all active themes
theThemes = theView.GetActiveThemes
r = Rect.MakeEmpty
for each t in theThemes
  r = r.UnionWith(t.ReturnExtent)
end

' set the extent of the display
if (r.IsEmpty) then
  return nil
elseif ( r.ReturnSize = (0@0) ) then
  theView.GetDisplay.PanTo(r.ReturnOrigin)
else
  theView.GetDisplay.SetExtent(r.Scale(1.1))
end

' update the layout
av.Run("exportLayout",nil)
```

C.0.6 themeAttr

```
'*********************************************************************
' Script Name: themeAttr
' Script Descritption: This script returns a string containing all the attributes
' pertaining to a certain field
' Script Inputs: a selected theme
'*********************************************************************

' checking for the correct number of attributes
if (self.Is(List).Not) then
  return nil
elseif (self.count < 1) then
  return nil
else
  themeString = self.Get(0)
  ' find the main view and activate it
  theView = av.GetProject.FindDoc("Main View")
  theView.GetWin.Activate

  ' find the them and get its table
  theTheme = theView.FindTheme(themeString)
  theTheme.SetActive(true)
  theTable = theTheme.GetFTab

  ' get the fields and append them to mystring
  theFields = theTable.GetFields
  mystring = ""
  for each f in theFields
    mystring = mystring ++ f.AsString
  end
  'msgBox.info(mystring,"")
  return mystring
end
```

C.0.7 getAllInfo

```
'*********************************************************************
' Script Name: getAllInfo
```

```
' Script Descritption: This script goes through all the themes of the view and gets all
' all their table attributes then dumps them into an info.txt file.
' Script Inputs: none
'*****************************************************************

' get the main view
theView = av.GetProject.FindDoc("Main View")

' create the output file
theWriteFile = LineFile.Make("/mit/nadinesa/thesis/code/info.txt".AsFileName,
#FILE_PERM_WRITE)

for each t in theView.getThemes

    ' get the table and the fields for each theme
    theTable = t.getFTab
    theFields = theTable.getFields

    ' write the name of the theme followed by the number of visible attributes
    theWriteFile.WriteElt("*")
    theWriteFile.WriteElt(t.getName)
    count = 0
     for each f in theFields
     if (f.isVisible) then
     count = count + 1
     end
     end
     theWriteFile.WriteElt(count.asString)

    ' write the names of the attributes each on a separate line
    for each f in theFields
         if (f.isVisible) then
             theWriteFile.WriteElt(f.getName)
        end
    end
end
```

C.0.8 themeTableComma

```
' ***************************************************************
' Script Name : themeTableComma
' Script Description : This script exports the table of a theme into a comma
' delimited text file
' Script Inputs: the theme, flag showing whether to export only selected
' records or the whole table(FALSE)
'***************************************************************

' Checking for the number of parameters
if (self.Is(List).Not) then
  return nil
elseif (self.count < 1) then
  return nil
else
  themeString = self.Get(0)
  selectedOnly = self.Get(1)

  ' Get the Main View and Activate it
  theView = av.GetProject.FindDoc("Main View")
  theView.GetWin.Activate

  ' Find the Theme and activate it
  theTheme = theView.FindTheme(themeString)
  theTheme.SetActive(true)

  ' Get the associated table and export it
  theTable = theTheme.GetFtab
  exportTab = theTable.Export("/mit/nadinesa/thesis/code/comma.txt".AsFileName,
    DText,selectedOnly)
  return 1
end
```

C.0.9 themeTableTab

```
' ***************************************************************
' Script Name: themeTableTab
' Script Description: This script exports the whole table of a theme into a tab
```

```
' delimited text file
' Script Inputs: name of the theme
'*****************************************************************

' Checking for the number of parameters
if (self.Is(List).Not) then
  return nil
elseif (self.count < 1) then
  return nil
else
  themeString = self.Get(0)

  ' Get the Main View and activate it
  theView = av.GetProject.FindDoc("Main View")
  theView.GetWin.Activate

  ' Find the theme and activate it
  theTheme = theView.FindTheme(themeString)
  theTheme.SetActive(true)

  ' create/overwrite the outpt file
  theWriteFile = LineFile.Make("/mit/nadinesa/thesis/code/tab.txt".AsFileName,
    #FILE_PERM_WRITE)

  ' write the headings into the file
   theWriteFile.WriteElt(themeString ++ "Table")
   theWriteFile.WriteElt("This table is tab delimited")

  aTab = 9.AsChar
   ' Get the table and the fields
  theTable = theTheme.GetFTab
  theFields = theTable.GetFields
  theExportFields = List.Make

  ' Initialize the first line
  theCopy = ""
  for each aField in theFields
    if (aField.IsVisible) then
      ' Append the field into the line
      theCopy = theCopy + aField.GetAlias.AsString + aTab
```

```
        theExportFields.Add(aField)
      end
    end
  ' remove the last tab from the line and write it to file
    theCopy = theCopy.Left(theCopy.count -1)
    theWriteFile.WriteElt(theCopy)

    if (theTable.GetSelection.Count  = 0 ) then
      theRecords = theTable
      numRec = theRecords.GetNumRecords
    else
      theRecords = theTable.GetSelection
      numRec = theRecords.Count
    end


    ' Go through each record of the table
    for each theRec in theRecords
      theCopy = ""
      for each aField in theExportFields
          theCopy = theCopy + theTable.ReturnValue(aField, theRec).AsString.Trans-
late(10.aschar," ") + aTab
      end
      'remove the last tab character
      theCopy = theCopy.Left(theCopy.Count - 1)
      theWriteFile.WriteElt(theCopy)
    end

    theWriteFile.close
    return 1
  end
```

C.0.10 setupLayout

```
  ' ************************************************************
  ' Script Name: setupLayout
  ' Script Description: This script is intended to be run only once in order to setup
  ' the layout of the project
  ' Script Inputs: none
```

```
' Comments: The view, scalebar and legend are automatically updated once the
' view changes
'*****************************************************************

' Get the Main Layout
_theLayout = av.GetProject.FindDoc("Main Layout")

' if the setup wasn't run before
if (_theVFrame = nil) then

  theDisplay = _theLayout.GetDisplay
  thePageExtent = theDisplay.ReturnPageExtent
  thePageOrigin = thePageExtent.ReturnOrigin

  ' creating the view frame
  ' the view frame is automatically updated every time the view changes

  theRectOrigin = thePageOrigin + (0.5@2)
  VRect = Rect.Make(theRectOrigin,(7.5@8))

  _theVFrame = ViewFrame.Make(VRect)
  _TheVFrame.SetView(av.GetProject.FindDoc("Main View"), true)

  _theLayout.GetGraphics.Add(_theVFrame)

  ' creating the scale bar
  ' the scale bar will also be automatically updated by the view

  theScaleOrigin = thePageOrigin +(0.5@0.5)
  SRect = Rect.Make(theScaleOrigin,(2@1))
  theSBFrame = ScaleBarFrame.Make(SRect)
  theSBFrame.setViewFrame(_theVFrame)
  theSBFrame.SetStyle(#SCALEBARFRAME_STYLE_ALTFILLED)
  theSBFrame.SetUnits(#UNITS_LINEAR_MILES)
  theSBFrame.SetIntervals(2)
  theSBFrame.SetInterval(2000)
  theSBFrame.SetDivisions(2)
  _theLayout.GetGraphics.Add(theSBFrame)

  ' setting up the north arrow
```

```
  ' no need to redraw it.
theNArrowOrigin = thePageOrigin + (0.5@1.25)
  NRect = Rect.Make(theNArrowOrigin,(0.4@0.6))
  theNArrow = NorthArrow.Make(NRect)
  theNArrODB = ODB.Open("$HOME/north.def".AsFileName)
  theList = theNArrODB.Get(0)
  theNArrowGraphic = theList.Get(7)
  theNArrow.SetArrow(theNArrowGraphic)
  _theLayout.GetGraphics.Add(theNArrow)

  ' creating the legend

  theLegendOrigin = thePageOrigin + (5.2@0.5)
  LRect = Rect.Make(theLegendOrigin,(3@1.9))
  theLgFrame = LegendFrame.Make(LRect)
  theLgFrame.SetViewFrame(_theVFrame)
  _theLayout.GetGraphics.Add(theLgFrame)

end
_theLayout.Invalidate
_theLayout.GetWin.Activate
```

C.0.11 exportLayout

```
' ****************************************************************
' Script Name: exportLayout
' Script Description: This script exports the main layout of the project into
' a specified postcript format file
' Script Inputs: none
' ****************************************************************

' Get the Main Layout
_theLayout = av.GetProject.FindDoc("Main Layout")

' Set the format to Postcript
theFormat = "PostScript (EPS)"

' Specify the output file name
theFileName = "/mit/nadinesa/thesis/code/layout.ps".AsFileName
```

```
' Export the layout into the file in the format specified
theExportDisplay= _theLayout.GetDisplay.ExportStartFN(theFileName,theFormat)
theExportDisplay.Invalidate(FALSE)
_theLayout.GetGraphics.SetDisplay(theExportDisplay)
_theLayout.GetGraphics.Draw

' Reset the graphics
_theLayout.GetGraphics.SetDisplay(_theLayout.GetDisplay)
theExportDisplay.ExportEnd
```

C.0.12 simpleQuery

```
' ***************************************************************
' Script Name: simpleQuery
' Script Description: this script constructs a simple query on one of the themes
' it updates both the table and the view
' Script Inputs: the theme, the attribute, the symbol (= < > ), the number
' ***************************************************************

' Check the number of arguments
if (self.Is(List).Not) then
  return nil
elseif (self.count < 1) then
  return nil
else
  ' Get the arguments
  themeString = self.Get(0)
  fieldString = self.Get(1)
  symbolString = self.Get(2)
  argumentNumber = self.Get(3)

  ' Find the view and activate it
  theView = av.GetProject.FindDoc("Main View")
  theView.GetWin.Activate

  ' find the theme and activate it and make it visible
  theTheme = theView.FindTheme(themeString)
  theTheme.SetActive(true)
```

```
    theTheme.SetVisible(true)

    ' Get the table and construct the query depending on the type of the argument
    theTable = theTheme.GetFtab
    if (argumentNumber.is(Number)) then
      expr = "([" + fieldString + "]" ++ symbolString ++  """ +argumentNumber.asString +
""" + ")"
    else
      expr = "([" + fieldString + "]" ++ symbolString ++  """""" +argumentNumber + """""""
+ ")" p = theTable.GetSelection
    end

    ' query the table and update the view and table
    theBitMap = theTable.GetSelection
    theTable.Query(expr,theBitMap,#VTAB_SELTYPE_NEW)
    theTable.UpdateSelection
    ' update the layout and the table
    av.Run("exportLayout","")
    av.Run("themeTableComma",{themeString,TRUE})
  end
```

Appendix D

System's Java Class Hierarchy

This Java class hierarchy was created using the Application Programing Interface documentation utility JavaDoc provided with Java. The class hierarchy includes all the classes and interfaces that were used for the project. They are presented in a top to bottom approach, where the higher order classes in the hierarchy are the base classes for the lower order ones. In order to differentiate between the built in Java classes and those created and used by the author for the Internet Based Collaborative Geographic Information System, we have typefaced the customized classes in bold.

class java.lang.Object

- **class .AVInterface**
- interface java.applet.AppletContext
- interface java.applet.AppletStub
- interface java.applet.AudioClip
- class java.awt.BorderLayout (implements java.awt.LayoutManager)
- class java.lang.Character
- class java.lang.Class
- class java.awt.Color
- class java.awt.Component (implements
- java.awt.image.ImageObserver)
    - class java.awt.Choice
    - class java.awt.Container
        - class java.awt.Panel
            - class java.applet.Applet
                - **class .GISApplet (Runnable)**
            - **class .DrawingPanel**
            - **class .MainPanel**
                - **class .ChatPanel**
                - **class .InterfacePanel**
        - class java.awt.Window
            - class java.awt.Frame (implements MenuContainer)
                - **class .DrawingFrame**
- interface java.awt.peer.ComponentPeer

- interface java.awt.peer.ContainerPeer
  (extends java.awt.peer.ComponentPeer)
- interface java.io.DataOutput
- class java.util.Date
- class java.util.Dictionary
  - class java.util.Hashtable (implements java.lang.Cloneable)
- class java.awt.Dimension
- class java.awt.Event
- class java.awt.FlowLayout (implements java.awt.LayoutManager)
- class java.awt.Font
- interface java.awt.peer.FramePeer (extends
- java.awt.peer.WindowPeer)
- class java.awt.Graphics
- **class .GraphicsUtil**
- interface java.awt.image.ImageObserver
- class java.io.InputStream
  - class java.io.FileInputStream
  - class java.io.FilterInputStream
    - class java.io.BufferedInputStream
    - class java.io.DataInputStream (implements java.io.DataInput)
    - class java.io.PushbackInputStream
- class java.awt.Insets (implements java.lang.Cloneable)
- interface java.awt.LayoutManager
- class java.lang.Math
- class java.awt.MenuComponent
  - class java.awt.MenuBar (implements java.awt.MenuContainer)
- interface java.awt.peer.MenuComponentPeer
- interface java.awt.MenuContainer
- **class .Message**
- class java.lang.Number
  - class java.lang.Double
  - class java.lang.Integer
  - class java.lang.Long
- class java.io.OutputStream
  - class java.io.FilterOutputStream
    - class java.io.DataOutputStream (implements java.io.DataOut-

put)

    - class java.io.PrintStream
- class java.awt.Point
- class java.awt.Polygon

- class java.util.Random
- class java.awt.Rectangle
- interface java.lang.Runnable
- class java.lang.Runtime
- class java.lang.SecurityManager
- class java.net.ServerSocket
- class java.net.Socket
- class java.net.SocketImpl
- class java.lang.String
- class java.lang.StringBuffer
- **class .StringInChoice**
- class java.lang.System
- class java.lang.Thread (implements java.lang.Runnable)
    - **class .KillThread**
    - **class .delayThread**
- class java.lang.ThreadGroup
- class java.lang.Throwable
    - class java.lang.Error
        - class java.lang.LinkageError
            - class java.lang.IncompatibleClassChangeError
                - class java.lang.NoSuchMethodError
        - class java.lang.ThreadDeath
    - class java.lang.Exception
        - class java.lang.ClassNotFoundException
        - class java.lang.CloneNotSupportedException
        - class java.io.IOException
            - class java.net.MalformedURLException
            - class java.net.SocketException
            - class java.net.UnknownServiceException
        - class java.lang.IllegalAccessException
        - class java.lang.InstantiationException
        - class java.lang.InterruptedException
        - class java.lang.RuntimeException
            - class java.lang.ArithmeticException
            - class java.lang.IllegalArgumentException
                - class java.lang.IllegalThreadStateException
                - class java.lang.NumberFormatException
            - class java.lang.IndexOutOfBoundsException
                - class java.lang.ArrayIndexOutOfBoundsException

    - class java.lang.SecurityException
- class java.awt.Toolkit
- class java.net.URL
- class java.net.URLConnection
- class java.net.URLStreamHandler
- interface java.net.URLStreamHandlerFactory
- interface java.awt.peer.WindowPeer (extends java.awt.peer.ContainerPeer)

Appendix E

Class Descriptions for Java Client Code

This appendix includes the API for all the Java classes created by the author for the Internet Based Collaborative Geographic Information System project. Each class starts with a short explanation of the class's purpose and role in the overall system then presents the class's method and arguments, along with a brief description of the functionality of each method.

E.0.1 Class MainPanel

```
java.lang.Object
  |
  +----java.awt.Component
        |
        +----java.awt.Container
              |
              +----java.awt.Panel
                    |
                    +----MainPanel
```

- **class MainPanel extends Panel**

  This is the base class for the interface panel and the chat panel. It identifies the general behavior and basic methods that the two classes share.

- **public synchronized boolean processMsg(Message msg)**

  Each panel overwrites this to appropriately process messages sent to them.

- **public synchronized void sendServerMsg(String msg)**

  Panels use this method to forward a message to the server.

E.0.2 Class ChatPanel

```
java.lang.Object
  |
  +----java.awt.Component
        |
        +----java.awt.Container
              |
              +----java.awt.Panel
```

```
                    |
                    +----MainPanel
                         |
                         +----ChatPanel
```

- **public class ChatPanel: extends MainPanel**

The chat panel is, as its name indicates, the part of the system where two or more users can engage in a real time text driven discussion.

- **public void activate()**

activate() is called when the user has successfully connected to the server. It makes the message area editable.

- **public synchronized boolean action(Event event, Object arg)**

The action method is called when the user makes a carriage return in the message area. The text gets selected and sent to the server to be forwarded to the other currently connected users. Overrides: action in class Component.

- **public boolean processMsg(Message msg)**

The ChatPanel processMsg takes the incoming string from the server and displays it in the text area of the applet. Overrides: processMsg in class MainPanel.

- **public boolean processMsg(String msg)**

processMsg with a string argument is used for inter panel communication, for debugging purposes only.

E.0.3 Class DrawingFrame

```
java.lang.Object
  |
  +----java.awt.Component
       |
       +----java.awt.Container
            |
            +----java.awt.Window
                 |
                 +----java.awt.Frame
                      |
                      +----DrawingFrame
```

- **public class DrawingFrame extends Frame**

The DrawingFrame is an integral part of the GISApplet. It is used to open a separate window for the collaborative painting part of the system. It contains the DrawingPanel that takes care of all the drawing messages sent by the server from other panels.

- **parent**

  public GISApplet parent: keeps a pointer to the originating applet.

- **DEBUG**

  public boolean DEBUG: when true the applet prints the debugging statements in the java console window of the browser.

- **public boolean action(Event e,Object arg)**

  If the event target is the menu item exit, the frame destroys itself. Overrides: action in class Component

E.0.4 Class DrawingPanel

```
java.lang.Object
  |
  +----java.awt.Component
        |
        +----java.awt.Container
              |
              +----java.awt.Panel
                    |
                    +----DrawingPanel
```

- **public class DrawingPanel extends Panel**

  The drawingPanel is where all the painting action takes place. It handles the user mouse clicks, key downs and choices while in the painting mode. It deals with the drawing server messages and processes them accordingly. This class is a modification of a JavaDraw class (http://cs.nyu.edu/phd_students/hain/javadraw/JavaDraw.java) to handle collaborative drawing and drawing on top of a background image.

- **public void paint(Graphics g)**

  The paint method repaints the image as well the current objects on the screen. Overrides: paint in class Component.

- **public void update(Graphics g)**

  The update method calls the paint method to avoid flickering. Overrides: update in class Component.

- **public boolean action(Event e, Object arg)**

  The action method takes care of the users' actions including choosing certain buttons for drawing, selecting and deleting objects as well for typing text. Any action is sent to the server according to the Message class. Overrides: action in class Component.

- **public boolean mouseMove(Event e, int x, int y)**

Mousemove is only used for debugging Overrides: mouseMove in class Component.

- **public boolean mouseDown(Event e, int x, int y)**

If the drawMode is in select, get the object selected. If it is in regular drawing mode, create the object(line, oval or rectangle). In any case, the message is sent to the server through the Message class to be forwarded to the other users. Overrides: mouseDown in class Component.

- **public boolean mouseDrag(Event e, int x, int y)**

If the drawMode is in select, move the selected object around. If the drawMode is in draw, update the boundaries of the current object and repaint it. Again, everything is forwarded to the server. Overrides: mouseDrag in class Component.

- **public boolean mouseUp(Event evt, int x, int y)**

When the mouse is up, refresh the screen. Send mouse up to the server as well. Overrides: mouseUp in class Component.

- **public boolean keyDown(Event e,int key)**

If the mode is textMode, then display whatever the user is typing on the canvas. Send the characters being typed to the server. This should be used only for annotation purposes (and not for chatting). The user should use the chat applet instead. This method is less efficient that the chat panel for chatting since in this case, the system is "painting" each character and sending it as a separate message to the server. Overrides: keyDown in class Component.

- **public synchronized void sendServerMsg(String msg)**

sendServerMsg is called by all the above methods to send a message to the server to be forwarded to the other clients.

- **public synchronized boolean processMsg(Message msg)**

Depending on the nature of the message sent by the server, processMsg calls different methods (the message can be mouse up, down, or drag as well as a change in color, in drawMode or in line width).

- **public void dealWithMouseDown(int x, int y)**

Called by the processMsg method, dealWithMousDown simulates a mousedown on the client in order to replicate the sender's changes. See mouseDown for detailed behavior.

- **public void dealWithMouseDrag(int x, int y)**

Called by the processMsg method, dealWithMouseDrag simulates a mousedrag on the client in order to replicate the sender's changes. See mouseDrag for detailed description.

- **public boolean dealWithMouseUp(int x, int y)**

Called by the processMsg method, dealWithMouseUp simulates a mouseup on the client in order to replicate the sender's changes. See mouseUp for detailed description.

- **public void dealWithKeyDown(String s)**

Called by the processMsg method, dealWithKeyDown simulates a keyDown on the client in order to replicate the sender's typing messages. See keyDown for detailed description.

E.0.5 Class GISApplet

```
java.lang.Object
  |
  +----java.awt.Component
        |
        +----java.awt.Container
              |
              +----java.awt.Panel
                    |
                    +----java.applet.Applet
                          |
                          +----GISApplet
```

- **public class GISApplet extends Applet implements Runnable**

The GISApplet is the client applet that comprises all the other elements of the system. It handles the connection with the server (sockets, ports, Input and Output streams, forwarding and receiving data).

- **public void init()**

The GISApplet comprises of the InterfacePanel, the ChatPanel and the DrawingFrame. The layout is initialized to a borderLayout one. Overrides: init in class Applet.

- **public synchronized void start()**

In this method, we initialize the receiveThread of the applet, the one responsible for retrieving all the messages from the server. Special care is taken for the case of applet resizing. Overrides: start in class Applet.

- **public synchronized void stop()**

When the stop method is called, the receiveThread is suspended. Overrides: stop in class Applet.

• **public synchronized boolean action(Event event, Object arg)**

The only action processed directly by the applet is that of user connection (by name entering). The panels are activated and the sockets and input/output streams are initialized. Overrides: action in class Component.

• **public void sendServerMsg(String str)**

This method is used by the panels to forward messages to the server.

• **public void destroy()**

destroy is called when the user leaves the page. It closes all streams and sockets. Overrides: destroy in class Applet.

• **public void run()**

This method catches the messages sent by the server and forwards them to the appropriate panel that knows how to handle the incoming commands/data.

E.0.6 Class AVInterface

java.lang.Object

|

+----AVInterface

• **public class AVInterface extends Object**

The AVInterface class contains the latest status of any ArcView related objects in the system. Using the latest versions of all selected objects through the user interface, it creates the ArcView calls to be forwarded to the ArcView RPC server.

• **public String runThemeOnOff(String theme)**

runThemeOnOff accepts a string and constructs an ArcView command that turns the theme with that name either on or off depending on its current status.

• **public void setZoom(String IO)**

setZoom accepts a string (in, out or toExtent) and populates the corresponding variable inOut with the string value. This value is later used when the zoom command is sent to the ArcView RPC server.

• **public void setzoomLevel(String level)**

setzoomLevel sets the value of the zoom level needed for the zoom command sent to ArcView.

• **public String runZoom()**

runZoom returns a string representing a call to the user defined ArcView zoom script.

- **public String runZoomToExtent()**

runZoomToExtent returns a string representing a call to the user defined ArcView zoomtoExtent script.

- **public void setQueryTheme(String theme)**

setQueryTheme sets the current theme to the string variable theme.

- **public void setAttribute(String att)**

setAttribute sets the current attribute to the string variable att.

- **public void setOperation(String op)**

setOperation sets the current operation to the string variable op (currently limited to = , < ,>).

- **public void setQueryValue(String value)**

setQueryValue sets the current expression value of the query to the string variable value (if value is a string, it has to be put between quotation marks).

- **public String runQuery()**

runQuery returns a string representing a call to the user defined ArcView script simpleQuery. It uses the current values of theme, attribute, operation and query expression.

- **public String runGetTable()**

runGetTable returns a string representing a call to the user defined ArcView script themeTableTab. The script exports a table into an ASCII tab delimited format file. The function uses the current theme.

- **public String runGetInfo()**

runGetTable returns a string representing a call to the user defined ArcView script themeAttr. The script currently only returns the names of the fields associated with the current table.

- **public String runUpdateLayout()**

runGetTable returns a string representing a call to the user defined ArcView script exportLayout. The script exports the main layout into a postcript file that is later on converted by the C server into a gif formatted file to be displayed on a homepage. This function doesn't require any parameters since the main layout in ArcView is automatically updated whenever the corresponding main view is changed.

E.0.7 Class InterfacePanel

```
java.lang.Object
  |
  +----java.awt.Component
        |
        +----java.awt.Container
              |
              +----java.awt.Panel
                    |
                    +----MainPanel
                          |
                          +----InterfacePanel
```

- **public class InterfacePanel extends MainPanel**

The interface or control panel is top portion of the applet (the bottom one being the chat panel one). This panel contains all the user interface elements needed by the user to communicate with the ArcView server and to update and edit the layout and tables.

- **public void setupInterface()**

setupInterface takes care of placing the components initialized by the constructor in their proper places. It also disables them waiting for a signal from the applet that the user established communication with the server. The layout is a GridBagLayout, used for its flexibility.

- **public void editLayout()**

When the edit layout button is pressed, the drawingFrame is shown on the screen.

- **public void queryThemeSelected()**

When a queryTheme is selected, the appropriate attributes show up in the attributes drop down menu. The attributes along with the themes are first retrieved from the ArcView server upon log in to the system.

- **public void attributeSelected()**

When an attribute is selected, the AVInterface is updated to reflect the current attribute and a corresponding attribute message is sent to the server to be forwarded to the other clients so that their interfaces also switch to the currently selected attribute.

- **public void operationSelected()**

The operation selected method is very similar to the attribute selection one.

- **public void zoomSelected()**

If the zoom level is either in or out, the AVInterface status is updated accordingly (an ArcView command is only sent when the update layout is pressed in this case because

the AV command needs the zoom in/out level as well). If the zoom level is toExtent, an arcview command is sent to ArcView to zoom to the extent of active themes.

- **public void getTable()**

  getTable sends a message to ArcView to export the table of the currently selected theme. The server forwards this message to all the clients in order to access the exported file.

- **public void executeQuery()**

  See query Value.

- **public void updateLayout()**

  updateLayout sends a message to ArcView to export the layout into a postcript file. The C program residing on the server then takes the newly created ps file and converts it into a gif file. All clients access the gif file to display the latest version of the layout.

- **public void getInfo()**

  getInfo is similar to getTable. However, instead of exporting the whole table, a message is sent to export the field names of the table of the current theme.

- **public void themeOnOff(String theme)**

  themeOnOff sends a message to the server to turn a certain theme on and off (depending on its current status). The layout is not updated. To update it the user has to press Updatelayout.

- **public void zoomLevel()**

  When the zoom level is entered, ArcView is instructed to execute the zoom.

- **public void queryValue()**

  When a query value expression is entered or the executeQuery button is pressed, ArcView is instructed to do the query and the message is passed on to the server.

- **public synchronized boolean action(Event e, Object arg)**

  The action method directs the action to the appropriate component depending on the type of the target and the type of the argument. All the previously discussed commands are called from within this action method. Overrides: action in class Component.

- **public boolean processMsg(Message msg)**

  When a new message arrives, depending on the type of message, the panel handles it by either updating the status of the AVInterface, or by selecting a choice menu, or by displaying the appropriate table and/or layout. The functions that follow are called from within processMsg. Overrides: processMsg in class MainPanel.

- **public void turnOnOff(String theme)**

turnOnOff is called if the incoming message is a themeonOff message. It basically selects/deselects the theme provided by the method argument.

- **public void selectQueryTheme(String atheme)**

selectQueryTheme is called if the incoming message is a themeSelected (for query) message. It basically selects/deselects the theme provided by the method argument.

- **public void selectAttribute(String attr)**

selectAttribute is called if the incoming message is an attributeSelected (for query) message. It basically selects/deselects the attribute provided by the method argument.

- **public void selectOperation(String op)**

similar to selectAttribute and selectOperation.All these methods also update the status of the class's AVInterface.

- **public void selectZoom(String z)**

similar to selectAttribute and selectOperation.

- **public void showTable(String file)**

showTable retrieves the table file and displays it in the info frame of the applet.

- **public void showLayout()**

showLayout retrieves the layout gif file and displays it in the layout frame of the applet.

- **public void disableAll(int txn)**

This method disables all the components. It is called when the user first logs in to the page and whenever the user presses (or receives) an execute query or update layout command. The reason disableAll is called when these events occur is because these events take some time and we don't want to user to mess up with the status of the system while we are fetching the previous results.

- **public void activate(int txn)**

This method activates all the components and calls getAllInfo.

- **public void getAllInfo()**

This method read an info file on the server, containing all the information necessary about the names of the themes and their corresponding attributes.

E.0.8 Class Message

java.lang.Object

|

+----Message

- **public class Message extends Object**

  This class handles all the message passed between the clients and the server.

```
/*************************************************************************
Constants used by the AV related commands
AV : forward to arcview
AF : don't forward to arcview
*************************************************************************/
        public final static String THEME_ON_OFF                 = "AVTO";
        public final static String SET_THEME                    = "AFST";
        public final static String THEME_QUERY_SELECTED         = "AFTQ";
        public final static String ATTRIBUTE                    = "AFAT";
        public final static String OPERATION                    = "AFOP";
        public final static String ZOOM_SELECTED                = "AFZS";
        public final static String ZOOM_TO_EXTENT               = "AVZE";
        public final static String QUERY_VALUE                  = "AVQV";
        public final static String SET_QVALUE                   = "AFQV";
        public final static String SET_ZOOM                     = "AFSZ";
        public final static String ZOOM_LEVEL                   = "AVZL";
        public final static String GET_INFO                     = "AVGI";
        public final static String GET_TABLE                    = "AVGT";
        public final static String EXECUTE_QUERY                = "AVEQ";
        public final static String UPDATE_LAYOUT                = "AVUL";
        public final static String EDIT_LAYOUT                  = "AFEL";

/*************************************************************************
Constants used by the Drawing Panels
*************************************************************************/

        public final static String DRAWING_MODE                 = "DMOD";
        public final static String TEXT_MODE                    = "TEXT";
        public final static String NO_TEXT_MODE                 = "NOTX";
        public final static String LINE_MODE                    = "LINE";
        public final static String RECT_MODE                    = "RECT";
        public final static String OVAL_MODE                    = "OVAL";
        public final static String SELECT_MODE                  = "SLCT";
```

```
        public final static String DELETE_MODE                  = "MELT";

        public final static String DRAWING_COLOR                = "DCOL";
        public final static String DRAWING_WIDTH                = "DWID";

        public final static String MOUSE_DOWN                   = "DMDN";
        public final static String MOUSE_DRAG                   = "DMDR";
        public final static String MOUSE_UP                     = "DMUP";

        public final static String KEY_DOWN                     = "DKDN";
/*************************************************************************
Constants used by the Chat panel
*************************************************************************/
        public final static String CHAT                         = "CHAT";
```

- **public Message(String msg)**

  Used when a message is received. It extracts the different parts of the message depending on the type of the message.

- **public String toString()**

  Overrides: toString in class Object.

- **public boolean isAV()**

  Returns true if the message is an ArcView message to be forwarded to the ArcView RPC Server.

- **public boolean isAF()**

  Returns true if the message is an ArcView message NOT to be forwarded to the ArcView RPC Server.

- **public boolean isChat()**

  Returns true if the message is a chat message.

- **public boolean isDR()**

  Returns true if the message is a draw message.

- **public boolean isSender(int port)**

  Returns true if the receiver of the message is the same as the message's sender. We don't want actions to be replicated in this case.

- **public boolean isUpdateLayout()**

  Returns true if the message is an update layout. used by the server to instruct the C program to convert the ps file into a gif one.

- **public boolean execQuery()**

  Returns true if the message is an exec query message.

E.0.9 Class KillThread

java.lang.Object

|

+----java.lang.Thread

|

+----KillThread

- **public class KillThread extends Thread**

  This code was taken from the java tips of Javaworld on-line java magazine. This code is needed in order to stop the applet from disconnecting from the server when the user resizes the applet.

- **public void run()**

  if the user leaves the page for five seconds, destroy the applet. Overrides: run in class Thread.

E.0.10 Class DelayThread

java.lang.Object

|

+----java.lang.Thread

|

+----delayThread

- **public class delayThread extends Thread**

  This class was added in order to provide some delay for the applet before retrieving the layout and table files. I wanted to make sure that the applet gets the latest complete update.

Appendix F

System's Server Code

F.1 Server Java Code Documentation

This is the java server of the system. It takes care of receiving commands from the connected clients and forwarding them back to the other clients. In case of a getlayout command, it sends a message to the C program to execute a ps to gif conversion. In case of an AV command, it also sends the commands to the C program to be forwarded to the ArcView server.

This class was originally taken from the Sun JavaSoft site. It was modified to deal with more than two clients at a time. It was also modified to provide a two way connection to the C program residing on the server.

The JavaServer class instantiates a ConnectionThread for every client that connects. This class is not shown here. The ConnectionThread class has not been modified. It basically takes care of creating a new thread for each client that connects. The thread also handles the input and output on the socket corresponding to its client.

- **public void start()**

Keep listening for clients trying to connect to the server. If the connection is successful, start forwarding the data to them.

- **protected ConnectionThread connectToClient(ServerSocket serverRSocket)**

The actual portion of the code where the server establishes a new connection when a new client connects to the rendez-vous port. The server then opens a new socket for this client and sends the port number to the client.

- **int everythingIsOk()**

This function checks if everything on the users' side is ok so that data forwarding can occur normally. If something is not ok, such as finding out that a client disconnected, the corresponding port and socket are closed for that client.

- **void cleanup(ConnectionThread tst)**

Closes the socket, killing its thread, and its input and output data streams.

- **public void forwardString(String string, ConnectionThread requestor)**

If a string is an ARcView command, only send it to the C program. If the command is execute query or update layout, then also export the ps file for the exported layout to gif format using the command imconvert.

- **void connect_to_C(String string)**

Establishes a connection with the C RPC client running on the server. Any ArcView command is forwarded to the C program and later on to the ArcView server for

processing.

F.2 Server C Code Documentation

F.2.1 AVClient.h

AVClient.C is not included in this appendix. The code can be downloaded from ESRI's homepage. The following class provides us with the functions along with their parameters needed to establish and use and RPC connection with ArcView.

```
#include <sys/types.h>
#include <sys/time.h>
#include <sys/socket.h>
#include <rpc/rpc.h>
#include <rpc/pmap_clnt.h>
#include <netdb.h>
#include <stdio.h>

typedef struct _RPCClntRec {
 CLIENT      * Client;
 char *      ErrStr;
 struct rpc_err LastErr;
 struct timeval TimeOut;
} RPCClntRec, *RPCClntPtr;
```

- **extern char * GetError();**

  GetError returns an error string if there is a current error on the client. If GetError returns NULL, then anRPCClient's last operation succeeded.

- **extern RPCClntPtr MakeClient();**

  MakeClient creates an RPC client connected to the RPC server specified by aMachine, aServer and aVersion and then returns the RPCClntPtr.Use the GetError function to determine if the creation succeeded.

- **extern char * ExecuteClient();**

  ExecuteClient sends the Avenue command string in aCmd to the ArcView RPC server to which anRPCClient is connected and returns the result (a string from ArcView). If the return string has 0 length, then the Avenue command was malformed.

- **extern void CloseClient();**

  CloseClient closes the client anRPCClient.

- **extern void SetTimeout();**

  SetTimeout changes the timeout period for anRPCClient to aTime. aTime is a number of seconds.

- **extern u_long GetTimeout();**
  GetTimeout returns the timeout period in seconds for anRPCClient.

F.2.2 Net.h

NetUtilities.c is not shown here. This class however provides us with the necessary interface needed to construct the final C program on the server.

- **int net_connect(char * host, char * service, int port, char * protocol);**

General network connection request function.

- **int tcp_open(char * host, char *service, int port);**

Open a TCP connection.

- **int writen(int fd, char * ptr_to_buffer, int nBytes);**

Write "n" bytes to a descriptor.

- **int readn(int fd, char * ptr_to_buffer, int nBytes) ;**

Read "n" bytes from a descriptor.

- **int net_init(char * service, int port);**

Initialize the network connection for the server

- **int net_open(int socket_file_descriptor);**

Initiate the server's end, accept connections .

- **int close_socket(int socket_id);**

Close an open socket.

References

[1] ArcNews Article. *GIS for Everyone- Now on the Web.* Vol. 18 No.4. pp.1-3.

[2] Avenue Customization and Application Development for ArcView. *Using Avenue Manual.* Environmental Systems Research Institute.1996.

[3] Berry, Joseph. *The Unique Character of Spatial Analysis.* GIS World. Vol. 9 No. 4. April 96. pp. 29.

[4] Black, James. *Fusing RDBMS and GIS.* GIS World. Vol 9 no 7. July 96. p. 44.

[5] Cappuccio, David. *Thin Clients, Fat Servers: The Real Skinny.* Data Communications Vol. 26 No. 4. March 21, 1997.

[6] Chiu, Dah Ming & Griffin, David. *WorkGroup Web Forum:Tools and Applications for WWW-Based Group Collaboration.* Digital Equipment Corporation Report. URL: http://www.asia-pacific.digital.com:80/info/internet/brochure/forum.html.

[7] Cranor, Lorrie. *Internet Collaboration: Good, Bad, and Downright Ugly.* The Association for Computing Machineray Magazine. February 1996.

[8] ESRI's homepage. URL: http://www.esri.com/.

[9] ESRI Internet Mapping Solutions. URL: http://maps.esri.com.

[10] ESRI. *Understanding GIS The Arc/Info Method.* Environmental Systems Reserach Institute. 1994.

[11] Flanagan, David. *Java in a Nutshell.* O'Reilly & Associates. 1996.

[12] Game, Adam & Owens, Stephen. *GIS on the Information Superhighway: Integration of GIS with Interactive Multimedia Directory Services.* Esri User Conference Proceedings. 1995.

[13] General Electric. *Software Engineering Handbook.* McGraw-Hill Book Company. 1986.

[14] Gustafon, Paul. *Transitioning to Collaborative Groupware Environments.*

[15] Hecht, Louis. *Choices Abound for an Internetworked Economy Using Spatial Information.* GIS World. March 96. Vol 9 no 3.pp 30.

[16] Hussein, Karim M. *Communication Facilitators for a Distributed Collaborative Engineering Environment.* Master Thesis submitted to the Department of Civil and Environmetal Engineering. MIT 1995.

[17] Huxhold, William E. *An Introduction to Geographic Information Systems.* Oxford University Press. 1991.

[18] MapObjects. *Embeddable GIS Components.* URL: http://www.esri.com/products/mapobjects/mapobjects.html.

[19] Murray, Don. Lutz, Dale. *ESRI's Spatial Database Engine: A Seamless GIS Solution.* ESRI User Conference 96.

[20] Netscape Homepage. URL: http://home.netscape.com/.

[21] Ohler,T. and Widmayer, P. *A Brief Tutorial Introduction to Data Structures for Geometric Databases.* Advances in Database Systems. Implementations and Applications. 1994.

[22] Ohler,T. and Widmayer, P. D*ata Structures and Algorithms for Geographic Information Systems: Selected Topics.* Advances in Database Systems. Implementations and Applications. 1994.

[23] Ohler,T. and Widmayer, P. *Geographic Information Systems: An Example.* Advances in Database Systems. Implementations and Applications. 1994.

[24] Perkins, Charles & Lemay, Laura. *Teach Yourself Java in 21 Days.* Sams.net. 1996.

[25] Plewe, Brandon. *Mapping on the Web: A Primer on Creating Geographic Services.* GIS World. Vol. 9 No. 1. January 96. pp. 56.

[26] Rajani, Purvi. *Moving towards the MainStream-Slowly.* GIS World. Vol9 no 9. September 96. pp. 86.

[27] Rajani, Purvi. *The Year of the Web Approaches.* GIS World. Vol 9 no 10. October 96. pp.9

[28] Schneider, Daniel K. and Block, Kayla. *The World-Wide Web in Education.* ANDREA. Vol. 2 No. 5. June 12, 1995.

[29] SDE. The Spatial Database Engine User's Guide Version 2.0

[30] Strand, Eric. *Java Creates New Channels for GIS Information.* GIS World. Vol. 10 No. 5. May 97. pp.28.

[31] Strand, Eric. *Open GIS Client/Server Products Remain Elusive*. GIS World. Vol 9 no 3. March 96. pp.36.

[32] Strand, Eric. *Spatial Data Plug into the Internet*. GIS World. Vol 9 no 10. October 96. pp.30.

[33] Strand, Eric. *Windows Applications Open to Geoprocessing*. GIS World. Vol 9 no 7. July 96. pp 30.

[34] VRML Web Site. URL: http://vrml.sgi.com/basics/.

[35] Tebbe, Mark. *Sorting the Strategies*. InforWorld. February 24, 1997. pp. 64.

[36] Thrall, Grant Ian. *The Web-ulous World of GIS*. Geo Info Systems. March 1997. Vol. 7 No. 3.

[37] Worboys, Michael F. *GIS A Computing Perspective*. Taylor & Francis. 1995.