NOTE TO USERS

This reproduction is the best copy available.

A FRAMEWORK DESIGN FOR COLLABORATIVE GIS APPLICATIONS:

BASED ON HYBRID ARCHITECTRE

by

Zheng (Eric) Chang
Bachelor of Engineering in Wuhan University, CHINA, 1994
Master of Engineering in Whuan University, CHINA, 1997

A thesis presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Applied Science

in the Program of Civil Engineering

Toronto, Ontario, Canada, 2005

Declaration

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Zheng (Eric) Chang

Department of Civil Engineering

Ryerson University

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Zheng (Eric) Chang

Department of Civil Engineering

Ryerson University

Borrower's Page

Ryerson University requires the signatures of all persons using or photocopying this thesis.

Please sign below, and give address and date.

Name of Borrowers	Date	Address	Signature

A Framework Design for Collaborative GIS Applications: Based on Hybrid Architecture

Zheng (Eric) Chang

Master of Applied Science, 2005

Department of Civil Engineering

Ryerson University

Abstract

Geographical information systems (GIS) software tools that support synchronous collaboration efforts among distributed decision-making participants can be very useful in many application areas, such as urban planning, engineering design, disaster and emergency response, and distant learning. However, most existing GIS tools do not provide adequate support for group interaction on decision-making and design scenarios. Early efforts on developing collaborative GIS tools have focused on collaborative geospatial information sharing and presentation in a group environment, mostly adapted to centralized client-server architecture for specific applications.

This thesis presents the results of a research project, aiming at providing such GIS software tools over the Internet. Based on the analysis of two mainstream architectures used in collaborative applications: centralized architecture and replicated architecture, a hybrid architecture is selected to develop a collaborative GIS framework as the platform for prototyping the aforementioned GIS tools. The discussion focuses on synchronous collaboration where people interact with each other using the system at the same time from different places. The prototype system, called GeoLink, addresses some important design and development issues such as session management and floor control through a message sending approach.

Acknowledgements

First of all, I would like to express my great appreciation and thanks to my supervisor, Professor Songnian Li, for his valuable advices and ideas as well as constructive suggestions and productive comments on my thesis. I also thank him for providing me the great opportunity to continue my study in the field of GIS and to work with him as a research assistant during my years at Ryerson University. I greatly appreciate his efforts to provide me financial supports throughout my course and thesis studies. Without support, I would never complete this thesis work.

I also attribute my accomplishment to Professor Jonathan Li, Professor Michael A. Chapman, Professor Mohamed Lachemi and other faculty and staff members in the Department of Civil Engineering, for their help and support. Many thanks go to Desmond Rogan and Domenic Valle for their technical assistance for solving computer and software problems. Many thanks are also extended to Leah Stanwyk, Kim Kritzer and Dianne Mendonca for their administrative support.

The School of Graduate Studies of Ryerson University is acknowledged for providing me the Ryerson Graduate Scholarships and the opportunity to conduct my research at Ryerson University from 2004 to 2005.

I also wish to thank my fellow graduate students in the Program of Civil Engineering at Ryerson University, Lijun Gu, Wenhao Gu, Khushnud Yousafz, Iqbal Ahmed, Xiaohong Ma,

Wensong Hu, Xu Sun, Yu Li, Haibin Liu, Ruiqiu Li and Hongmei Zhao for their help. The study time with them always recalls my beautiful and happy recollections.

Last, but not least, I wish to express my gratitude to my parents for their unconditional love and support, and to my wife, Weihong Zhan, and my son Xiao Cheng for their endless love, patience, and encouragement. Without their help and understanding, I would never have completed this thesis.

Table of Contents

List of Figures

List of Tables

List of Acronyms

2D	Two-Dimensional
3D	Three-Dimensional
API	Application Programming Interface
AWT	Abstract Windows Toolkit
CORBA	Common Object Request Broker Architecture
COM	Common Object Model
CSCW	Computer Supported Cooperative Work
DBF	Database Format
ESRI	Environment Source Research Institute
EOC	Emergency Operations Center
GIS	Geographic Information System
GUI	Graphical User Interface
HTML	HyperText Markup language
IP	Internet Protocol
ISIS	Integrated Software for Imagers and Spectrometers
LAN	Local Area network
JSDT	Java Shared Data Toolkit
MOO	MUD, Object Oriented
MUD	Multiple User Dimension
MVC	Model-View-Controllor
OCX	OLE Control Extension
OGC	OpenGIS Consortium
PDA	Personal Digital Assistant
RPC	Remote Procedure Calls

QA	Quality Assurance
RDBMS	Rational Database Management System
SDK	Software Development Kit
SQL	Structured Query language
TCP/IP	Transmission Control Protocol/Internet Protocol
WCS	Web Coverage Services
WFS	Web Feature Services
WYSIWIS	What You See Is What I See
XML	Extensible Markup Language

Chapter 1 Introduction

With the improvement of software technologies especially in Object-Oriented Software Engineering, RDBMS, Internet and Web Services, Geographic Information System (GIS) has improved greatly through adopting these technologies. GIS has changed from stand-alone workstations to multi-user, Internet-based software which can be run not only on desktop computer but also personal digital assistant, cell phone, etc. The map data are also changed from the excluded format to open standards. The GIS software itself also offers common components like OCX, COM, or API which can be used by other software developers. The GIS is becoming more robust, easier and more open to use than before.

However, challenges still appear when GIS confronts more complex requirements: the users not only want to share data but also want to share applications at the same time when they are at the different locations. These requirements are usually required in urban planning, emergency management, group spatial decision support system, etc. The following section will describe the main problems in these work places.

1.1 Problems

GIS has penetrated most branches of governments and business to deal with routine work. All kinds of GIS applications in these branches are used to solve specific problems. They could work well in their branches to handle daily work and solve specific problems. However, when more complex system requirements appear, for example, the systems are used in site selection for urban planning or emergency management for Emergency Operations Centres (EOCs), the use of traditional GIS applications faces many problems.

In site selection, for instance, in order to evaluate and locate a new airport in an urban area, people have to consider a number of factors such as land acquisition and construction costs, accessibility, relationship to existing services, and different types of environmental. Different stakeholders, such as land owners near the proposed sites and environmental activists, would hold diverse viewpoints on its solution. Consequently, solutions are often formulated by groups, such as committees or task forces, in which individuals from diverse backgrounds bring their expertise to search for solutions and for the interpretation of results.

Emergency management also faces the similar situation. Large, complex emergencies often affect multiple departments or multiple agencies and require data to be collected and assembled from a variety of locations quickly under adverse conditions. Part of the Emergency Operations Center's role is to understand the details of the emergency, order the required response resources, coordinate with adjoining agencies (federal, provincial, and local), and determine the immediate actions necessary to contain the incident. In emergency operations center, there is local first responder expertise as well as a network of experts that can be tapped to provide additional support. This second layer of expertise is usually remote and separate from the wealth of information and situation awareness provided by the emergency operation system. These supporting elements frequently find themselves making decision without the benefit of some critical information that was available locally. Moreover, when major disasters happen, it may be nearly impossible to get all the personnel from their normal location to the emergency operations center in major metropolitan areas. Major disasters produce nearly instantaneous gridlock and

congestion. The key personal cannot afford to be struck in traffic during the transit time which could be the most critical minutes and hours of the emergency.

To solve such problems, GIS software not only should be adapted to the group decision making styles [Armstrong, 1994], which is named spatial decision support system, but also be adapted to remote and real time collaboration. In other word, at the same time the resources and data in different organizations which could be in different locations need to be shared as well as the operations run on different systems so that people can point to specific features, or circle an area on a map in order to make their intentions clear. This kind of collaborative computing technique is also termed the Computer Supported Cooperative Work (CSCW) application, or groupware application.

Currently, groupware developers are faced with the complex task of building multi-user, multi-computer systems on top of single-user, single-computer infrastructures. Most contemporary user interface systems and toolkits, such as X windows, Microsoft Windows and the Java Abstract Windows Toolkit (AWT) [Begole, 1997], were not designed with groupware and multi-user interfaces in mind.

As a corollary of the lack of support for presence and awareness, contemporary user interface systems did not have to deal with issues such as consistency between user interface objects or supporting the externalization of the user interface object state for latecomers. Even with support for presence, awareness and collaborative consistency management in place, it would be hard to reuse some single-user interface constructs, since their design may be based on the assumption

that there is only one user, who is doing only one thing at a time. For example, the standard bounding box that is typically displayed for objects being moved is based on assumption that only a single object is on the move at any time. When multiple objects are moved by multiple persons, it may become hard to see which object is going where [Hill, 1994].

Though many development difficulties are encountered, several frameworks, for example Habanero, Groove, GroupArc, and so on, had been developed to support collaborative GIS functions. These early efforts focused on combining GIS with CSCW hardware systems and software (groupware), or at least applying CSCW concepts in developing collaborative GIS systems (see examples from [Churcher, 1996]; [Faber, 1997]; [Jones., 1997]; [Nyerges, 1997]; [MacEachren, 2001]; [Li and Coleman, 2005]). These developments used either an existing commercial GIS system or in-house developed GIS viewer, and integrated it with a groupware system such as electronic meeting systems. But several important problems still have not been solved: 1) there are no efficient solutions to modify current single user GIS to collaborative GIS; 2) GIS components are highly dependent on the collaborative components which cause very complex system design and coding problems.

This thesis will discuss the design and development of a framework of a synchronous collaborative GIS system with an open standard software component. The objectives of the thesis project will be shown in the next section.

1.2 Research Objectives

This research focuses on the design and development of a framework for a synchronous collaborative GIS application with open standard software components. This research will achieve following objectives:

1. Analyze the core system requirements of collaborative GIS; and design, develop and implement a real-time prototype synchronous collaborative GIS system.

2. Develop scalable and extendable collaborative models for a synchronous collaborative GIS; and find an effective approach to extend single user GIS applications to collaborative GIS applications.

1.3 Research Methodology

The first phase of the research started with a literature review which searched relevant research papers both in CSCW domain and GIS domain. The solutions in these domains were evaluated and analyzed. The main purposes included building necessary terminology and knowledge for the research, summarizing the main-stream solutions from CSCW domain, analyzing and defining basic system requirements for collaborative GIS.

The second phase of the research was to find potential and logical solutions based on the first phase research. In this phase, high level system logical designs, which include architecture design, framework design and function definitions, were achieved.

In the third phase, some free of charge groupware tools downloaded from the Internet were tested and evaluated to decide which tool satisfied the system designs. Some other tools, like GIS tools, rapid development environments, and deployment approaches, were evaluated and chosed in the next development phase as well.

The fourth phase was the development phase in which the prototype was developed with the tools from last phase. At this stage, some programming work was done using JBuilder 9, MapObjects Java Edition and JSDT APIs. Sometimes this phase may go back to Phase 2 because the design problems may be found in this phase.

Last phase was the test phase. In this phase, the main functions and performance of the system were tested not only under controlled laboratory conditions but also outside laboratory through Internet. Figure 1.1 presents an overview of the research methodology described above.

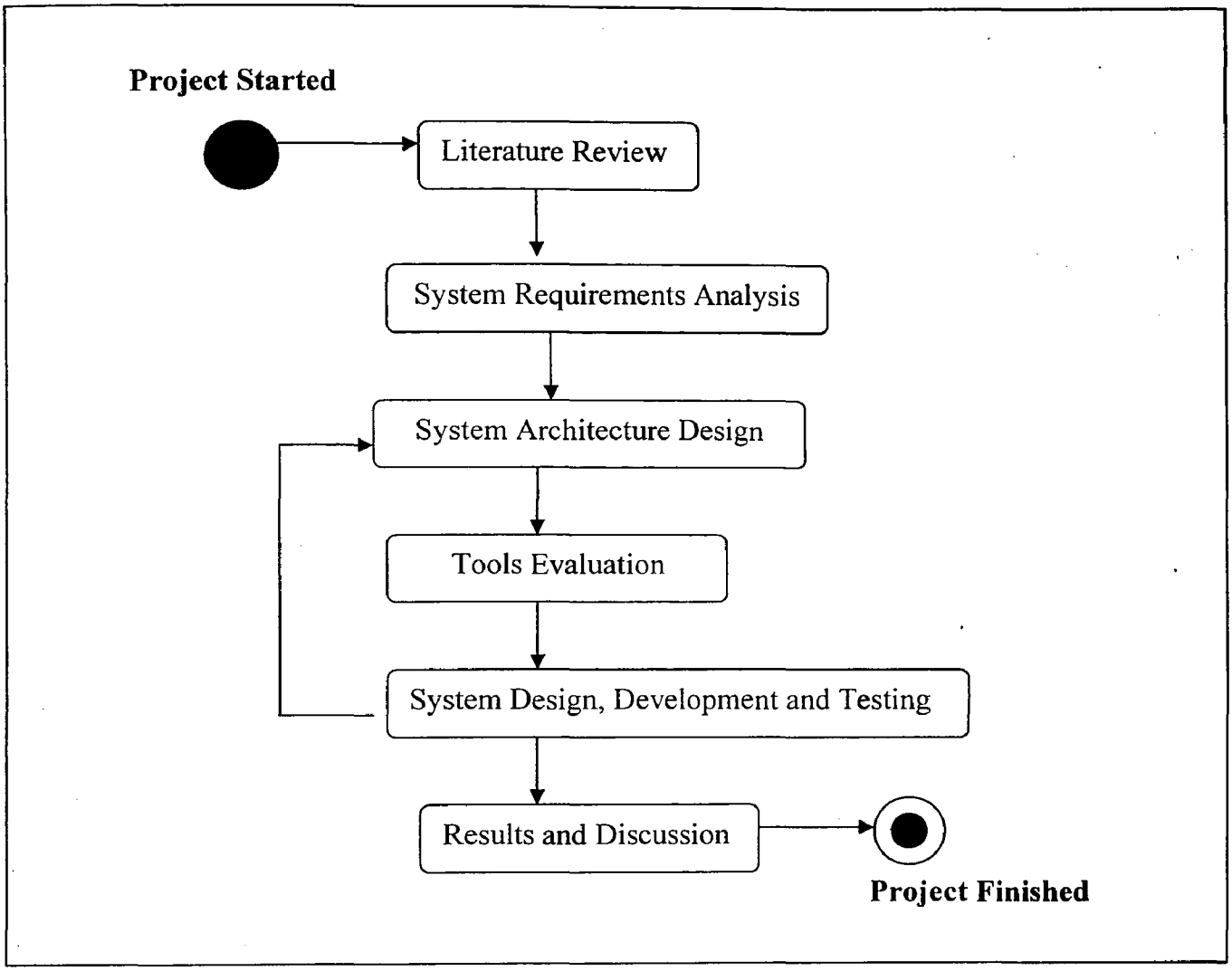


Figure 1.1 An overview of the research methodology

1.4 Organization

In Chapter 1, the problem description of collaborative GIS is provided. The research objectives and research methodology are described as well.

Chapter 2 describes the overview and background of collaborative applications. The definition and classification of CSCW are introduced, followed by an introduction of current synchronous

collaborative applications. Some specific design issues in this subject area are addressed in the last.

Chapter 3 introduces a case study: Emergency Operation Centre (EOC). The work flow of EOC is introduced and related system requirements derived from the workflow are discussed in this chapter.

Chapter 4 presents an architecture design for collaborative GIS application based on the hybrid model. The problems and related solutions about system consistency are also discussed in this chapter.

In Chapter 5, the detailed system design, development and deployment of a prototype: *GeoLink* are presented. A framework of the prototype is first given. Some specific problems confronted for collaborative GIS are discussed and the related solutions are provided. The deployment of the prototype is introduced. Finally, the experience of this prototype is represented.

Chapter 6 provides some concluding remarks and future work. The research contributions and limitations are addressed first. Then, future work is discussed afterwards.

Chapter 2 Background and Related Work Review

In this chapter, literatures that are relevant to the proposed research are discussed. First, computer support cooperative work (CSCW) and groupware are introduced, and then current collaborative systems and CSCW toolkits are discussed. The architecture design for CSCW system is introduced afterwards. Finally, the specific issues for the design of collaborative applications are discussed.

2.1 Computer Supported Cooperative Work and Classification

Computer supported cooperative work is a computer-based system that supports group of people engaged in a common task (or goal) and that provides an interface to a shared environment [Ellis, 1991]. The applications that are designed to support group work are often referred to as *groupware,* which has been defined as "technology that communicates and organize unpredictable information allowing dynamic groups to interact across time and space" [Cameron, 1995]

Both in academia and in industry, various groupware prototypes and products have emerged and provided particular functionality to users. Each groupware system is designed to support a particular cooperative work situation or a particular range of cooperative work situations. Cooperative work settings are very diverse in terms of task, duration, group, organizational context and culture [Hinssen, 1998]. Table 2.1 represents an overview of the categories, aliases and some representative systems according to the summary of Hofte [1998]. Table 2.2 describes the functions of some of these applications.

Table 2.1 Groupware categories and representative systems

Groupware category	**Roughly equivalent names**	**System Cases**
Computer conferencing system	Bulletin board systems, newsgroups	TeamTalk, Lotus Notes
Chat systems		Internet Relay Chat, BBS chatting room
Workflow management systems	Office procedure systems, coordination systems	Staffware, FlowMark
Electronic meeting systems	Group (Decision) support system (G(D)SS), electronic meeting rooms	GroupSystems
Application sharing systems	Screen/window sharing systems, desktop/data·conferencing systems	XTV, Netmeeting
Shared whiteboards	Shared drawing systems	GroupDraw, GroupShetch
Co-authoring systems	Collaborative/joint/shared editing systems	GROVE, Quilt
Multi-user hypermedia systems		NCSA Hypernews
Collaborative virtual environments	Multiplayer games, virtual worlds	Xtrek, Quake
Group scheduling systems	Group calendaring systems	Microsoft Schedule, Lotus Calendar
Audio conferencing		Netscape Conference,

system		Netmeeting
Video conferencing system	Multimedia conferencing system	ClearBoard, Netmeeting
Collaborative software engineering systems		GroupCRC

Table 2.2 Function descriptions of some groupware applications

Groupware category	**Functions Descriptions**
Computer conferencing system	Setting up sessions, audio, video, shared application connections Adding late joiners, more than 2-way connections Migrating to other ways of interaction (asynchronous, subgroups) Integration of other media (phone conference, PictureTel)
Chat systems	Providing text-based computer-mediated discussions between users. Each letter or sentence that is typed is immediately observable on the screens of other users, which facilitates rapid turn taking in discussions.
Workflow management systems	Coordinating asynchronous transfer and development of information Version control User permissions Synchronization Notification of new material Group calendaring

Application sharing systems	Taking an existing single-user application and makes it shareable Broadcasting graphics, mouse movements, and edits to all participants Input focus control sharing, floor control Telepointers and "Master" pointer Integrated with audio, video, text chat connections (session management)
Co-authoring systems	Different phases of authoring, e.g., brainstorming, doing research, planning, writing, and reviewing; Text-only documents, formatted documents or multimedia documents; Simultaneous document editing and/or sequential document editing; annotations, versions and revisions; Communication between authors about the document or the authoring process; Coordination of the authoring process.
Collaborative virtual environments	Creating a virtual place populated with avatars that can navigate and interact with other people and objects in the environment Persistent places Containment and tracking of objects User extensible Shared video and audio, spatialized audio, selective groupings of users

	Multi-user text chat, MOO
Group scheduling systems	A number of users are allowed to align their electronic calendars and schedule a meeting

The product-based classification above describes groups of systems with similar features that support a particular range of cooperative tasks. However, it is not unambiguous, nor exhaustive. For example, NetMeeting system, which is considered as a video conferencing system, also includes a shared application and shared whiteboard.

Groupware systems are therefore often classified according to the type of collaboration that they support. In the classification scheme, collaboration has a temporal and spatial dimension, and these dimensions are commonly shown using the time-space matrix in Table 2.3 ([Joha, 1998]; [Dix, 1996]).

Table 2.3 Time-space matrix (adapted from [Joha, 1998]; [Dix, 1996])

		Time	
		Same time	Different times
Place	Same place	Face-to-face (tabletop displays, meeting support tools)	Asynchronous interaction (project scheduling, coordination tools, shift work systems)
	Different place	Synchronous distributed (shared editors, video-and audio- conferencing tools)	Asynchronous distributed (email, newsgroups)

According to the matrix, groupware systems can be either synchronous or asynchronous based on a time dimension and can be co-located or distributed based on a place dimension.

1. Same time (synchronous) and same place (co-located) collaborative application
2. Different time (asynchronous) and same place (co-located) collaborative application
3. Different time (asynchronous) and different place (distributed) collaborative application
4. Same time (synchronous) and different place (distributed) collaborative application

The research reported here has focused on the design and development of the last group: synchronous collaborative GIS application. So the literature review will focus on synchronous collaborative systems.

2.2 Synchronous Collaborative Systems

2.2.1 Microsoft NetMeeting

Microsoft NetMeeting [NetMeeting, 2000] enables real-time voice and data communications over the Internet. This includes the ability for two or more people to share applications, transfer files, view and illustrate a shared whiteboard, and chat over standard connections.

With application sharing, a user can share a program running on one computer with other participants in the conference. Participants can review the same data or information, and see the actions as the person sharing the application works on the program (for example, editing content or scrolling through information). Participants can share Windows-based applications transparently without any special knowledge of the application capabilities.

The person sharing the application can choose to collaborate with other conference participants, and they can take turns in editing or controlling the application. Only the person sharing the program needs to have the given application installed on their computer. The shared clipboard enables a user to exchange its contents with other participants in a conference using familiar cut, copy, and paste operations. For example, a participant can copy information from a local document and paste the contents into a shared application as part of group collaboration.

The Microsoft NetMeeting Software Development Kit enables developers to integrate this conferencing functionality directly into their applications or Web pages. This open development environment supports international communication and conferencing standards and enables

interoperability with products and services from multiple vendors. Figure 2.1 represents the main components and relations of NetMeeting Software Development Kit.

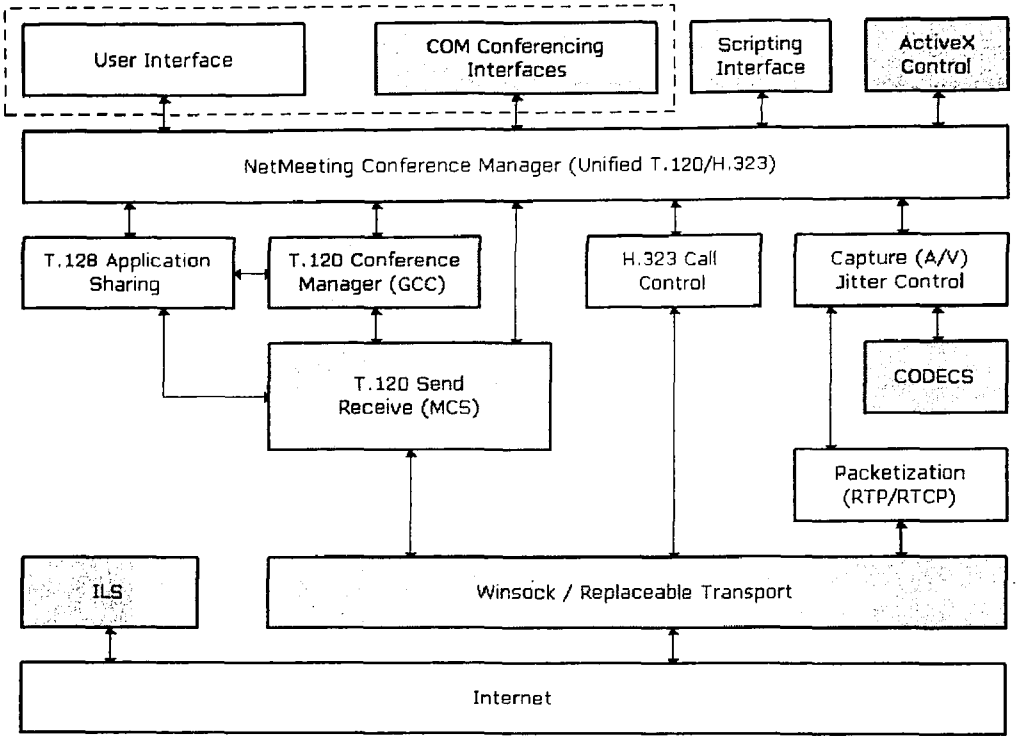


Figure 2.1 NetMeeting SDK components and relations [NetMeeting, 1999]

To support this dual purpose, NetMeeting capabilities are built on architecture of networking components. Each component communicates with and passes data to and from the component layer above and below. This architecture, which is based on industry standards, ensures that manufacturers can easily develop products and services that build on the NetMeeting platform and interoperate with NetMeeting client conferencing features.

At the core of the NetMeeting architecture is a series of data, audio, and video conferencing and directory service standards. Figure 2.1 shows how these standards work together with transport,

application, user interface, and the Windows NetMeeting Software Development Kit (SDK) components to form the NetMeeting architecture [NetMeeting 2.1 Overview, 2000].

NetMeeting is a centralized application. A central conference agent receives all user input to the application. The conference agent then distributes display updates to the participants' windowing systems. The architecture is centralized because collaboration is based on single user application being shared among participants.

2.2.2 GroupArc

GroupArc (see Figure 2.2) was initially developed to explore the potential of lightweight CSCW browsers for GIS applications. It is written in Tcl/Tk language [Ousterhout, 1994], runs on Unix, Macintosh and Windows platforms and uses GroupKit.

GroupKit [Roseman, 1992] is a toolkit for building a general class of collaborative applications and includes a number of awareness widgets for use in GroupKit-based applications [Churcher, 1996]. This toolkit will be introduced later.

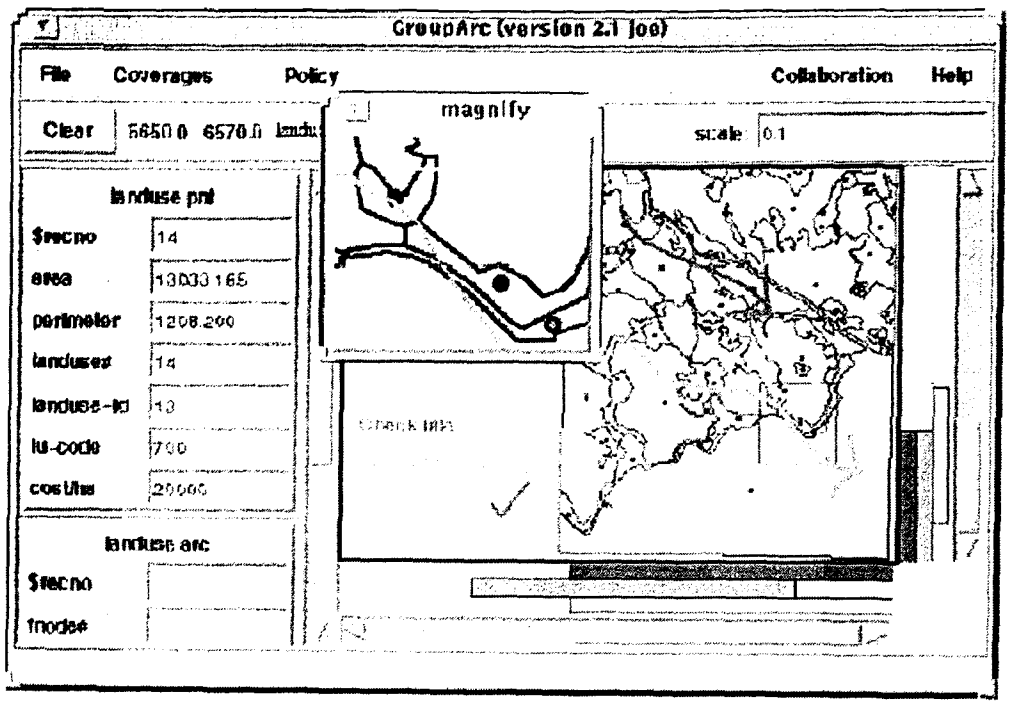


Figure 2.2 Interface of GroupArc [Churcher, 1996]

When GroupArc is running, GroupKit manages the registration of conference participants (who may enter or leave at any time) and communication between the GroupArc replicas on individual participant's workstations (see Figure 2.3).

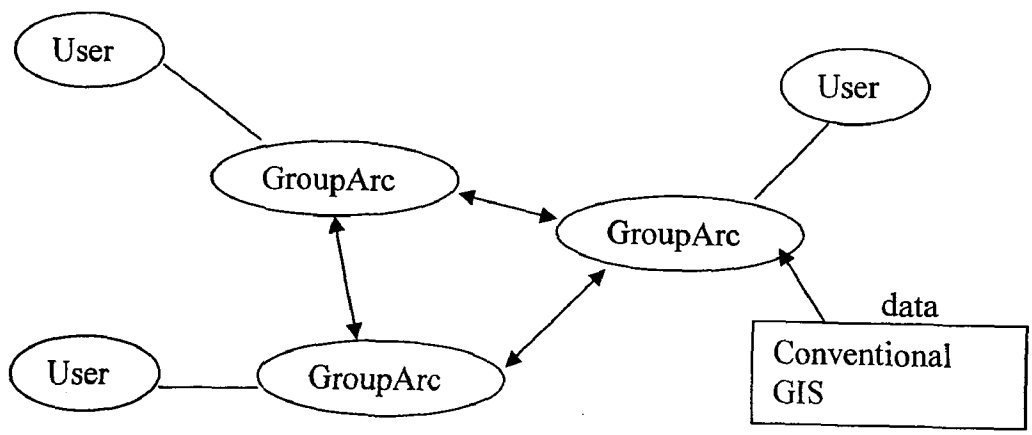


Figure 2.3 Users communicate through GroupArc [Churcher, 1999]

GroupArc allows physically separated users concurrently to browse and annotate GIS data in a cooperative way. Each participant must have Internet access and also their own copy of the GroupKit and GroupArc software. The participants can therefore be in different buildings, cities out in the field and all participate in the discussion [Churcher, 1999]. Therefore GroupArc is a replicated collaborative GIS system.

2.2.3 Habanero

Habanero is a collaborative framework and environment containing a set of applications. Through Habanero one can interact with other people on the Internet using a variety of applications that share state and events. Habanero is written in Java, and runs under any operating system that supports Java 2 and JINI v1.0 [Chabert, 1998].

The Habanero client, server and applications provide the necessary environment to create collaborative workspaces and virtual communities. The server hosts sessions and connects the clients that interact with the sessions using a variety of applications called Hablets. Sessions can be recorded, persistent, access restricted and even anonymous. The Habanero client provides the interface to define, list, create, join and interact with a session. The client provides session information, user identification, a notification mechanism, record and replay capabilities, security, a list of active users and tools, an address book and a capability to easily create session templates. The client has two modes: one is used to pre-define sessions off-line and the other is used to interact with active sessions on-line in real-time. The Habanero server is capable of hosting multiple sessions and the client is capable of joining multiple sessions with multiple unique instances of the tools (see Figure 2.4) [Chabert, 1998].

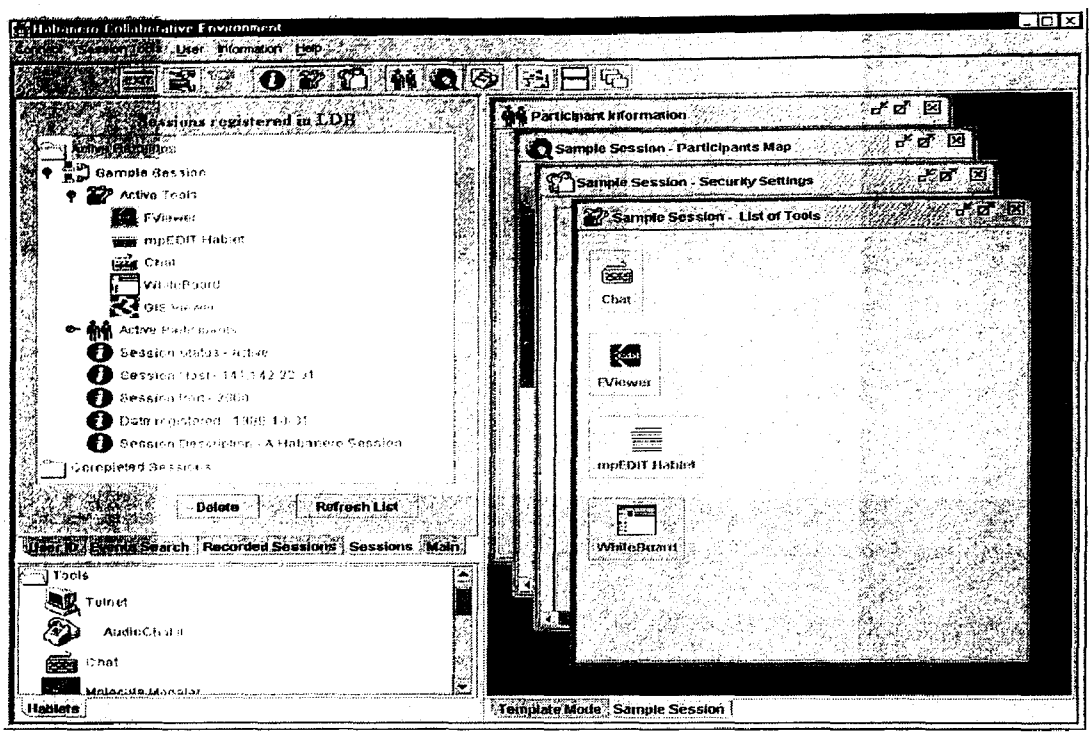


Figure 2.4 Habanero interface [Chabert, 1998]

2.2.4 CollabWorx

CollabWorx. Int. based in Syracuse, New York, is an end-to-end provider of secure, high performance and multi-platform collaboration, communication and distance learning solutions.

Two critical components of the platform are the plug-in (middleware), and the Meeting Engine (The notion of plug-in is used here in a loose sense. Technically, TI implementation for the Internet Explorer does not use a plug-in). Middleware has been introduced to ensure high platform reliability. It also plays an important role in platform security. The crucial idea behind middleware is that collaborative tools cannot communicate directly with the messaging server. Instead, they use middleware method calls for such communication. The middleware acts as a message filter and multiplexer, preventing applications from sending ill-formed or illegal

messages. An ill-behaved tool might not work but it would not disturb the overall platform operation. (see Figure 2.5)

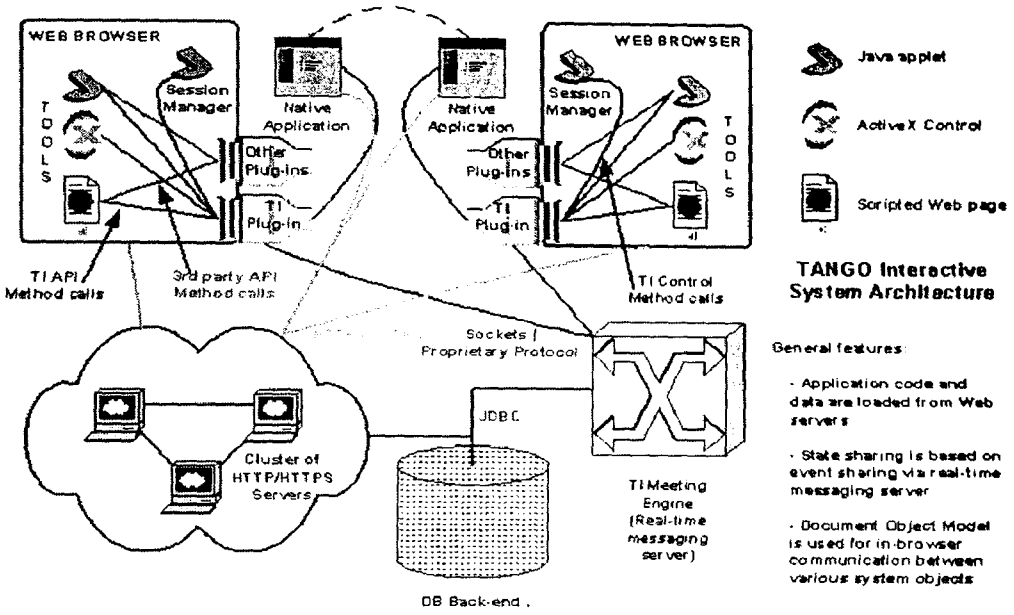


Figure 2.5 Architecture of the CollabWorx web-based collaboration platform [Collabworx]

Use of the plug-in as data multiplexer simplifies implementation of the messaging server. Each platform client connects to the engine just once. A proprietary protocol provides all services necessary to identify message sources and addresses. In absence of the multiplexing plug-in, each application would have to open a separate connection to the messaging server. Such a solution practically precludes implementation of strong and coherent security mechanisms. In the TI platform secure communication is implemented between the plug-in and the messaging server. The messaging server is a central element of the platform. It is primarily responsible for message routing. A collaboratory Engine accepts permanent connections and it holds a considerable

amount of state. The Engine works together with all active instances of CollabWorx Session Managers to establish a fault-tolerant state of the collaboratory sessions.

2.3 Synchronous Collaborative Toolkits

Collaborative toolkits are used to develop collaborative systems. These toolkits include GroupKit, DisEdit, Java Shared Data Toolkit (JSDT), GroupIE, etc. This section will present these tools.

2.3.1 GroupKit

GroupKit [Roseman, 1992] is a toolkit for developing real-time synchronous groupware applications. It is based on Berkeley's public domain Tcl/Tk [Ousterhout, 1994] language. GroupKit extends Tcl, and it uses Tk for its user interface and Tcl-DP for its communication needs.

GroupKit is based on a purely replicated architecture except for a central registration server called the registrar. The registrar maintains a list of sessions and users, and it provides functions that allow users to join and leave sessions via registrar clients. A registrar client provides a user interface that allows a client to join/leave a session. Figure 2.6 shows the architecture for a GroupKit application run by two clients.

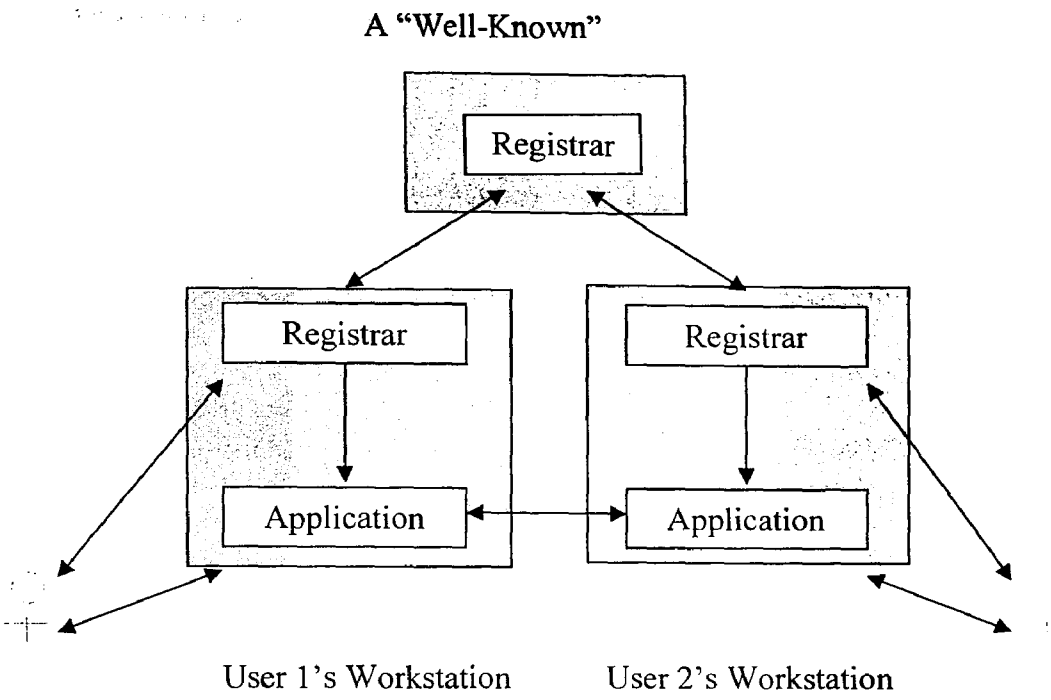


Figure 2.6 GroupKit run-time system with two applications

GroupKit contains useful groupware widgets for real-time synchronous applications. Some of the widgets are as follows:

- a multi-user scrollbar that shows the scrollbar positions for not only the local but the remote users
- a participant widget that shows a user's information and status
- a telepointers widget that allows users to track remote users' cursors
- a radar overview that provides a global view of the application along with colored overlays representing each user's local viewing region

2.3.2 DisEdit

Another interesting toolkit is DistEdit [Knister, 1990] which is used to build shared text editors. Shared editors are used for collaborative writing, in which users can simultaneously edit a

document. DistEdit allows existing editors to be extended with group editing features. It uses distributed communication architecture to replicate user's changes to text to the other users.

The ISIS (Integrated Software for Imagers and Spectrometers) distributed programming toolkit [Birman, 1990] is required along with a Tcl/Tk parser for DistEdit. ISIS is a software system for processing and analyzing the data sets produced by imaging spectrometers. ISIS provides the core communications for DistEdit, and Tcl/Tk is used by DistEdit to display conference information within status windows.

In order to use DistEdit with a text editor, the editor must supply routines that allow DistEdit to access its internal text buffer, query and move the text cursor, call DistEdit's undo and open file procedures instead of its own, etc.

DistEdit provides many different and interesting features that can be integrated into an editor. The following describes these features:

- Region Locking where a user can lock a region of text assuming it does not already have a lock on it by another user,
- Lock-Step Mode that enables users to couple their cursors together so when a cursor is move by one user the others' cursors are moved to the same position, and
- Local and Global Undo where a user can undo his last change (local) or can undo another user's last operation (global).

2.3.3 JSDT

A more recent toolkit is the Java Shared Data Toolkit (JSDT) [Burridge, 1997]. This toolkit defines a multipoint data delivery service for collaboration-aware Java applications. Like the previous two toolkits, it is a third generation toolkit that requires programmers to learn a set of APIs.

Each application that wishes to use the JSDT must implement a Client interface. The Client interface provides methods that return the name of the client and perform authentication of a new client joining to a session. A list of Client objects are contained within a Session object which is initiated by a Server object. The JSDT also makes use of a Channel object that allows for multi-party communications between two or more clients. Channels reside on a Server, and each Client obtains a proxy of the Channels from the Server. The JSDT also provides methods for locking resources via token passing.

Unlike the previous toolkits, JSDT is Java based, which means it can be used on any system that supports Java. This is a significant advantage over the others since many groupware applications need to run on heterogeneous platforms. However, unlike GroupKit, JSDT does not provide any groupware widgets that allow for easy construction of multi-user GUI elements.

2.3 Specific Design Issues in Synchronous Collaborative System

Several specific challenges confronted with the complex task of building multi-user, multi-computer systems on top of single-user, single-computer infrastructures. Due to the specific

nature, the development of synchronous collaborative application involves many issues that introduce additional technical problems compared to the development of single-user applications and conventional distributed applications.

2.3.1 Architecture Model Design

According to the design pattern of Model-View-Controllor (MVC), three architectures are classified: centralized, replicated, and hybrid (semi-replicated). Suthers [2001] classified the fourth architecture: distributed. Because it is still a mix of centralized architecture and replicated architecture, it can be assigned into classification of hybrid. Following will discuss the benefits and disadvantages of these models.

Centralized

In a centralized architecture, only one instance of application runs in a central server. The server is responsible for controlling all input and output to the distributed end-users. A sequence of events generated by end-use interaction are collected and sent to the central server. The output of the shared application must be broadcasted to all participating users for visualization (Figure 2.7).

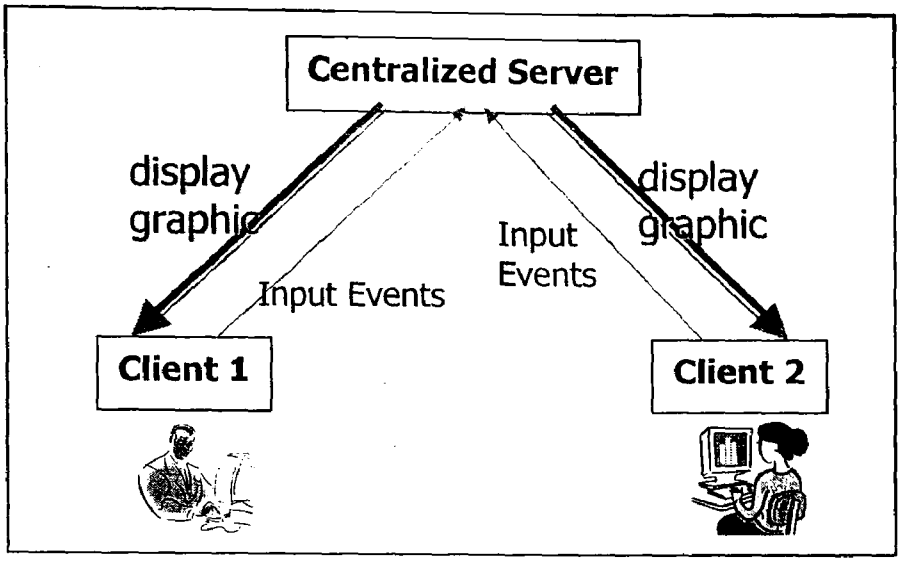


Figure 2.7 Centralized architecture

Examples of applications based on the centralized architecture are NetMeeting [Microsoft, 1999], SharedX [Garfinkel, 1994], and XTV [Chung, 1994]. An important advantage of a centralized architecture is that it guarantees consistency of shared data. The disadvantage is that it requires higher bandwidth to distribute display information to all end-users, Strict What You See Is What I See (WYSIWIS) interface, less responsive to user input, and less fault tolerance [Begole, 1999].

Replicated

In a replicated architecture, the entire application is installed and run on each client machine; and some means of synchronization between them is provided [Suthers, 2001]. Figure 2.8 illustrates how the replicated collaborative application works over IP Network.

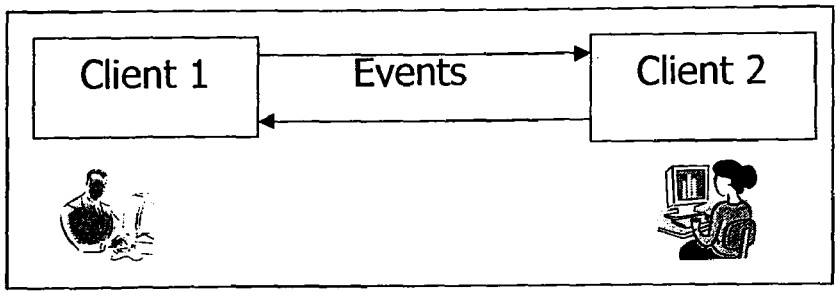


Figure 2.8 Replicated architecture

Both the input events and the graphics output are processed locally. Examples of applications based on this architecture are GroupKit and DistEdit. As opposed to a centralized architecture, a replicated architecture requires lower bandwidth because output is only locally transmitted. This increases performance and scalability. The disadvantage is the increased complexity in handling data source. Because there are multiple copies of shared data in a replicated architecture, it is generally expensive to keep the state of shared data replicas synchronized.

Hybrid (semi-replicated)

Applications are decomposed by many components. Some components may require to be shared while others may be required to be replicated according to functionalities. Therefore a hybrid or semi replicated architecture are introduced. Hybrid architecture does not represent a "pure" new category, but represent the composition of centralized and replicated architecture.

Dewan [1999] presented a multi-layered model or hybrid architecture to represent a generic framework for collaborative applications (Figure 2.9).

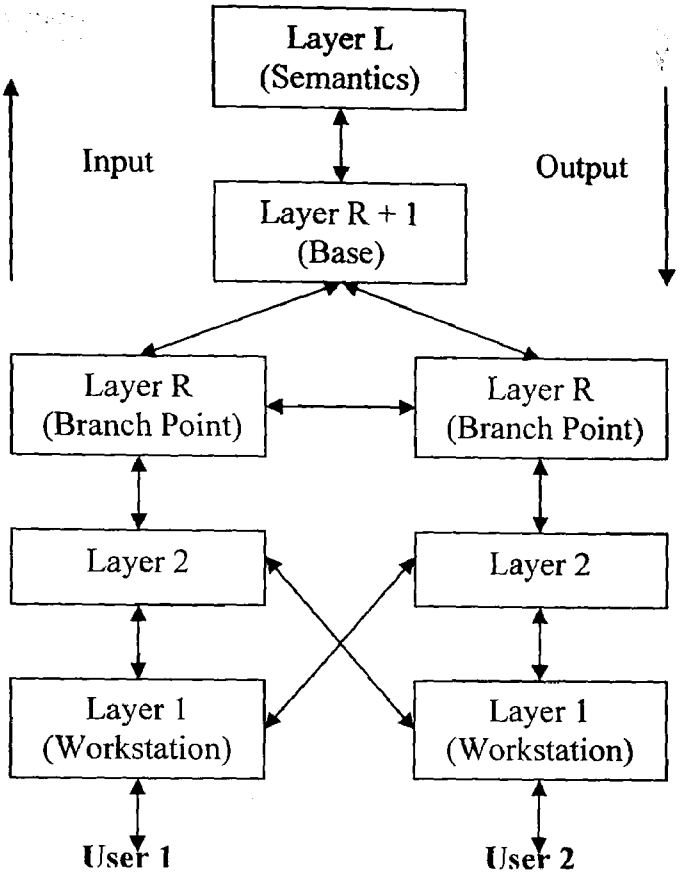


Figure 2.9 Dewan's generic architecture

A layer can be seen as a component corresponding to a specific level of abstraction. At the bottom we have the hardware component, and at the top we find systems semantics. Above Layer R we find the architectural stem (the stem consists of the shared layer L and layer R+1). The stem includes what is called the base of the systems - layers that are shared between users. At some point, the base gives rise to branches which are replicated for every user (this includes the replicated layers R to 1), these are called the peers. Communication between layers is called interaction events (vertically information flow between layers) while communication between peers is called collaboration events (horizontally information flow between layers).

2.3.2 System Consistency

The groupware system that contains a shared workspace function, will maintain some kind of consistency between two or more representations of the shared workspace, despite concurrent activity of human users. One of the problems faced by groupware developers is that such concurrent user activity may lead to concurrent actions on the shared workspace, which may cause inconsistent representations of the shared workspace. In order to describe the emergence of inconsistency, an example mentioned by Hofte [1998] is given.

There are two users taking two actions to a variable *S* respectively. User 1 adds one to the variable *S*, and User 2 multiplies the variable S by two (see Figure 2.10).

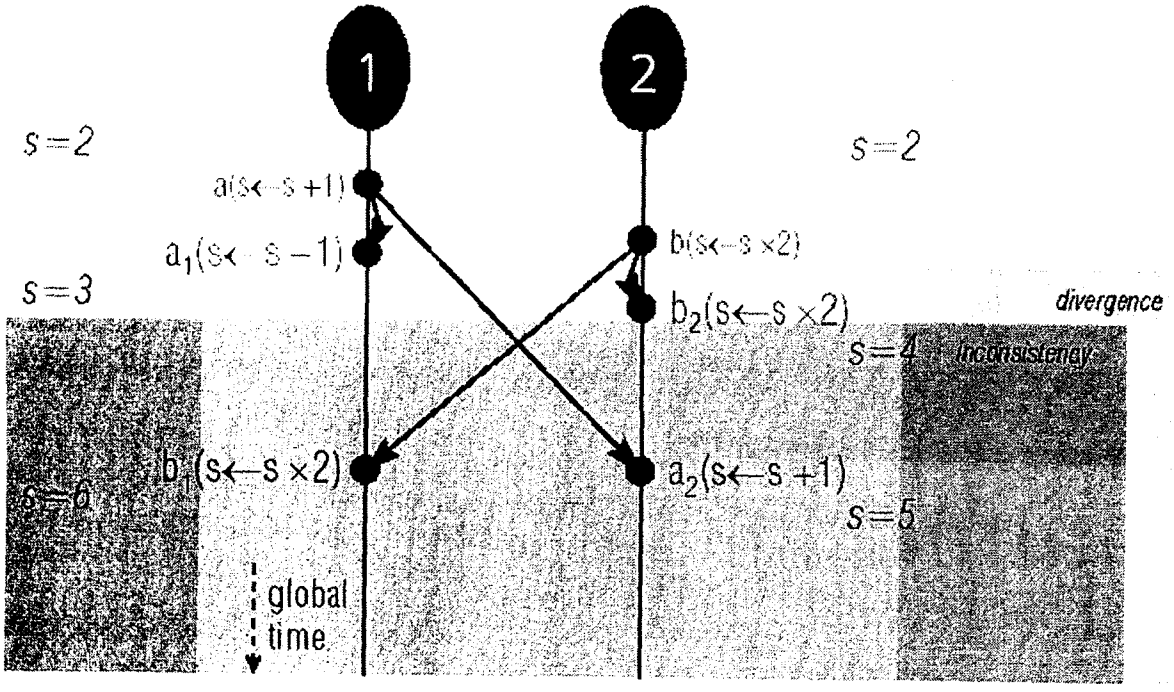


Figure 2.10 Emergence of inconsistency due to the ordering of the actions

At the beginning, action "*a*" (a request to add one to the variable s) at location 1 happens and causes a *feedback* action a1. This action "a" also causes a feedthrough action "*a2*" at location 2.

While action "b" (a request to multiply the variable by two) at location 2 happens. This action "*b*" causes feedback action "*b2*" and Feedthrough action "*b1*" at location 1.The divergence emerges when a1occurs at location 1: two users can observe different values for s (user1 obtains value 3 and user 2 obtains value 4). The concepts of feedback and feedthrough will be introduced in the next section.

This kind of inconsistency situation can be seen in many resource sharing systems. The typical solution, for example in database systems, is to prevent the emergence of inconsistency by making sure that only one user at a time can gets the privilege to modify the database. Other users who want to modify the database are denied. However, in groupware development, the issues involved in dealing with inconsistency are fundamentally different from those in conventional database applications. In some specific situation, for example, when several plans are discussed by experts, these plans are different from each other and therefore the inconsistency is reasonable and sometimes is necessary.

Therefore, there are two main strategies to deal with emerging inconsistencies in groupware systems: avoiding or allowing the emergence of inconsistency. From a technological perspective, strategies to avoid inconsistency generally increase response and notification times, and reduce the availability of the medium for actions, compared to strategies that allows for inconsistency.

There are two ways in which groupware systems can avoid inconsistency: ordering and locking.

- Ordering: one strategy to avoid inconsistencies is to accept actions that may cause inconsistency, but postpone their execution to a moment that will not cause the

emergence of inconsistency. When users take actions, these actions always follow the same ordering. For example, in Figure 2.11, the ordering of the actions in User 1 is the first User 1's action (Add one), followed by User 2's action (multiply by two); the ordering of the actions in User 2 is the first user 1's action (Add one), followed by User 2's action (multiply by two).

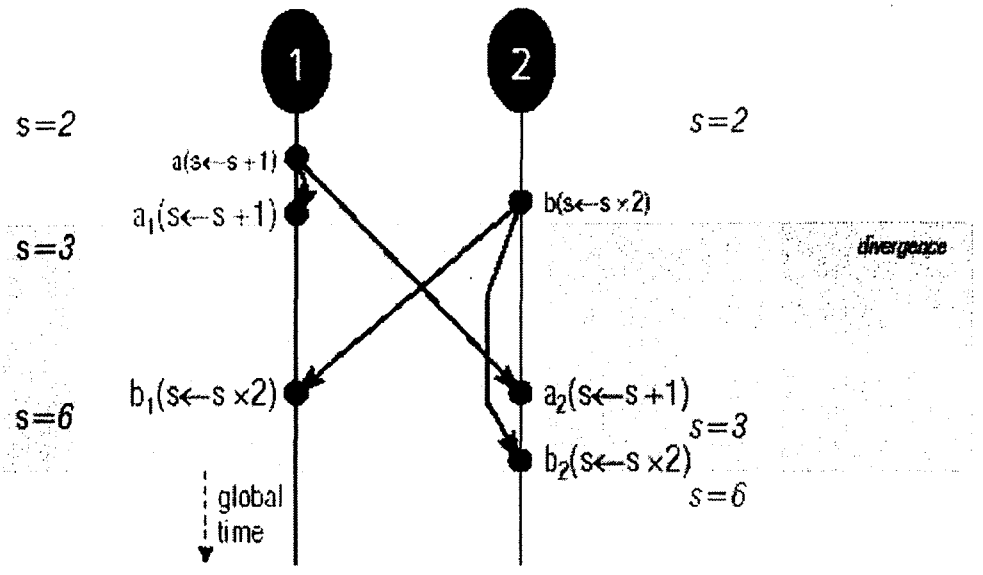


Figure 2.11 Avoid inconsistencies with ordering method

- Locking: another strategy is not to accept actions that may cause inconsistency through denying some actions by some users. This may be realised by locking. Locking is a form of coordination that gives only one user at a time the privilege to initiate actions as long as the user holds the lock.

Allowing inconsistency allows the emergence of inconsistency and support different, parallel versions for each user in the groupware system. The reason for allowing inconsistencies includes several aspects:

- Improve response and notification times.
- Make media available for action during the periods of disconnection in mobile systems.
- Make the consistencies on purpose. For example, when two users are working on alternatives for a chapter of a book, or when each user is temporarily allowed to control his scrollbar position (Multi-user scrollbars) on a shared workspace, instead of sharing scrollbar positions.

The consistency in this situation can be maintained through providing additional support to synchronize the multiple parallel versions. Two methods are used to detect the emergence of inconsistency:

- Automatic notification. As soon as the system detects the emergence of inconsistency, the users may be notified, who may then take appropriate action to synchronous the parallel versions themselves, or to use the parallel versions individually, thus postponing the consistency re-establishment.
- User-requested notification. If the system cannot provide automatic inconsistency notification, it may offer to compare the different parallel versions upon user request and detect whether there is a difference or not (e.g., by sending over the complete state of different versions and comparing them).

2.3.3 Feedback, Feedthrough and Awareness

Feedback is a kind of message that is produced by itself and gets the response message from other participants. Feedthrough is a kind of message that is produced by other participants. The participants will, under normal circumstances, be able to receive feedback of their own actions and receive feedthrough from the actions of others (see Figure 2.12).

To be able to receive feedthrough from the actions of others is essential in many cooperative situations. Dix [1996] claims that this feedthrough is many times more important than the direct communication.

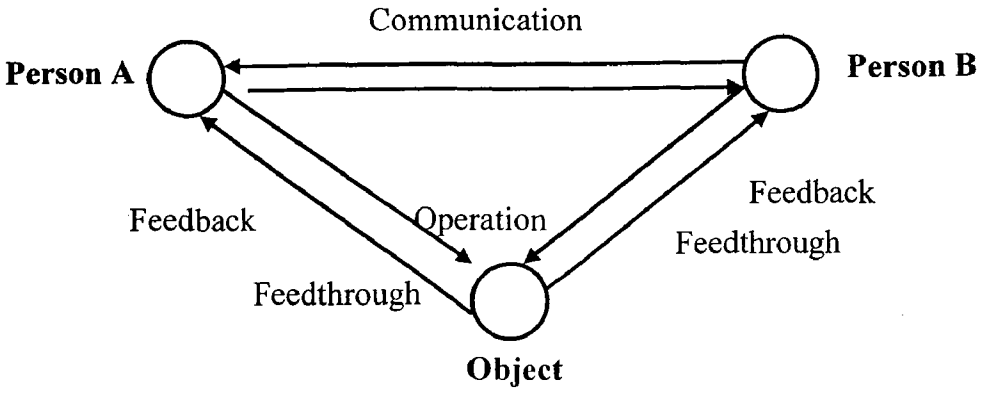


Figure 2.12 Feedback and feedthrough

In a cooperative setting not only is it important to see one's own updates, but also to see the effects of other people's actions. This is feedthrough. The presence of feedthrough effectively creates an additional channel of communication through the artefacts themselves. In real life, cooperating over physical objects, this communication through the artefact is often more important than direct communication. For example, imagine you are moving a large piano. You may say things to each other – "move your end up a bit", "careful of the step" - but in fact the most important thing is the feel of the other person's movements through the movements of the piano. This sort of communication is effective partly because it is tied so intimately to the work itself, and partly because it is implicit, unconsciously noticed and acted upon [Dix, 1996].

Feedthrough is often weak in electronic cooperation, and this is worsened by the need to make objects "close" to the person updating them. Effective feedthrough is essential to fluid collaboration and must be a major issue for any cooperative application.

The word awareness is used frequently within CSCW. Usually it refers to awareness of the presence of other people. That is, awareness of who is around and their availability for cooperative activity (see Figure 2.13). Feedthrough is also a form of awareness, in this case awareness of what has happened. However, there may often be several possible causes of a change and in order to complete the picture we need awareness of how the change happened, which, together with our conversation with other people and understanding of the context, allows us to infer why it happened [Dix, 1996].

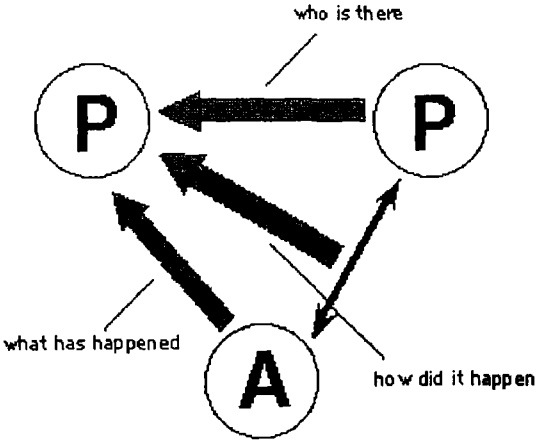


Figure 2.13 Awareness [Dix, 1996]

To be able to cooperate, users must be aware of other's presence and actions. One of the main tasks of any groupware system is to provide users with information to maintain such awareness, even though users may not be continuously working in the same room, or working at the same time. Usually, some functions are specified for the awareness, for example radar views,

telepointers, multiuser scrollbar, etc. A radar view shows collaborators' locations in the shared workspace by representing their views as rectangles in a miniaturized version of the shared workspace. The rectangles are of different colors, each corresponding to a collaborator. A radar view is often used with telepointers. A telepointer is a trace of remote mouse cursor movements, and each rectangle in a radar view may use a telepointer to show the mouse movements of a user whose view of the shared workspace is represented by the rectangle. A multiuser scrollbar also identifies users' locations in a shared workspace. Instead of using a miniaturized shared workspace, however, the multiuser scrollbar is directly incorporated into the shared window that displays the shared workspace.

Chapter 3 Collaborative GIS System Requirements Analysis

Some system requirements and functionality of CSCW applications have been mentioned in Chapter 2. But they are not enough to specify the detailed functions of a collaborative GIS system. Some specific features in GIS domain are also needed. This chapter begins with the system requirement analysis of collaborative GIS based on the case study of the work flow of Emergency Operations Centre (EOC). Then, some system core functions are analyzed and derived from this study. Finally, the data source used in all kinds of systems is discussed.

The requirements derived from the case study are an abstract specification which is not only used for EOC, but also is specified as the core functions for any applications like EOC or other spatial decision support systems. Some detailed requirements especially related to EOC itself are not involved because EOC system itself is also a very complex system which is beyond the scope of this paper.

3.1 Case Study

The Emergency Operations Centre (EOC) is a facility designated for managing the disaster emergency. It is where the Incident Management Team makes decisions to allocate and coordinate resources, provide for incident communications coordination and direct the overall disaster emergency response. It provides for the centralized locating of the five functional sections of the Incident Management Team: Command; Operations; Planning; Logistics; and Finance. The workflow of EOC is presented to demonstrate how it works according to KPB-Emergency Response Plan in the Kenai Peninsula Borough [KPB, 2004] (see Figure 3.1).

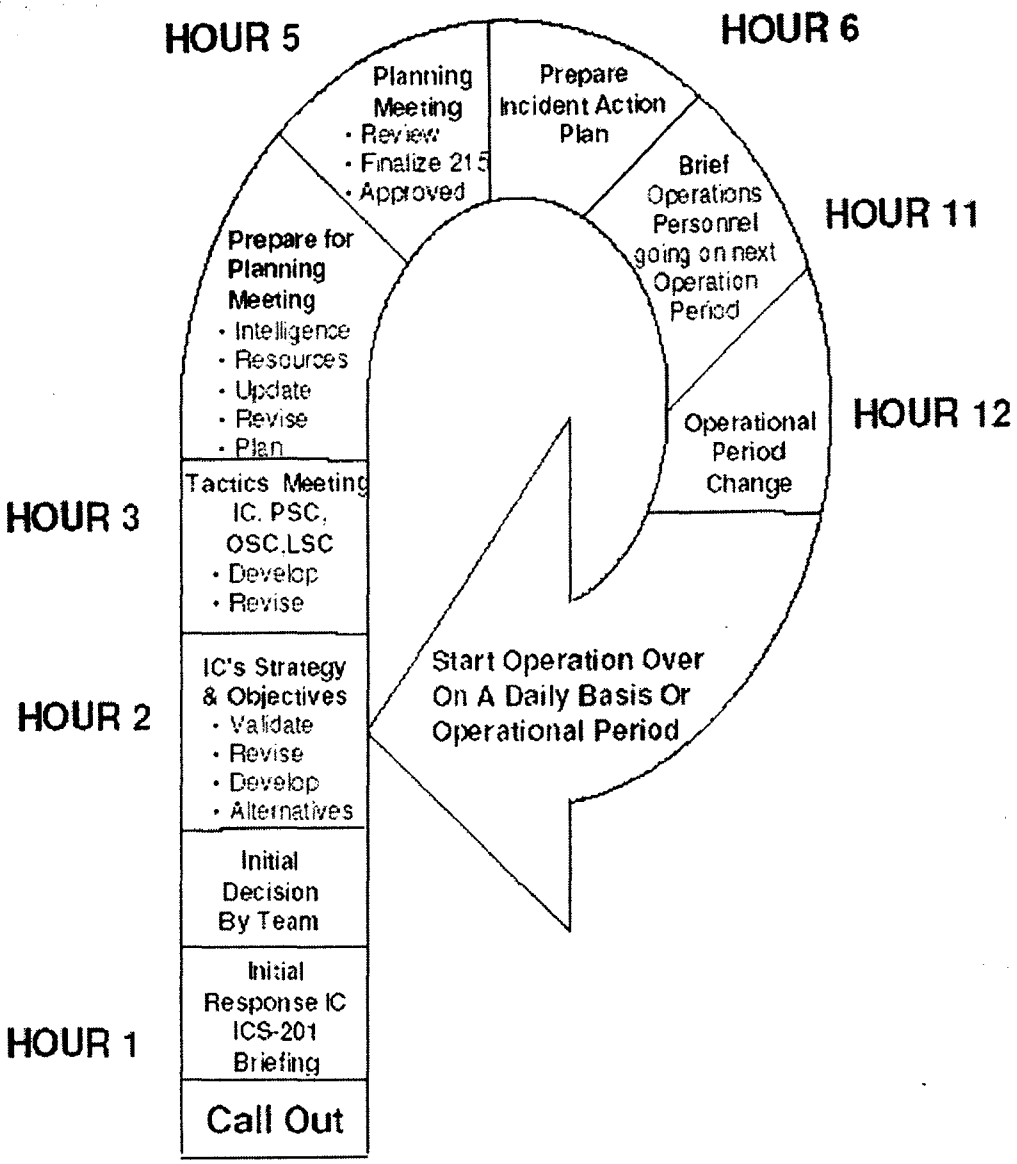


Figure 3.1 Emergency operations centre daily work flow [KPB, 2004]

There are four major phases in the 12 hours shift: Setting Incident Objective; Tactic Meeting; the Planning Meeting; Finalizing, Approving, and Implementing Incident Action Plan.

The first phase is to set incident objectives and goals which will be finished in the next 12 hour shift.

In the Tactic meeting, the direction of how resources will be deployed to the incident objectives will be provided. This blueprint of tactical deployment for the next operational shift will be developed and revised before the planning meeting where formal deployment of resources and work assignments will be determined. Some control line locations are determined and Division/Branch boundaries for geographical assignments for next operational period will be established as well. After determining divisional boundaries, specific work assignments for each division/group will be developed for the next operational period.

Before the planning meeting, make sure that the participants, locations, time, planning maps, forms, resource, and situation status are available and up-to-date. In the planning meeting, a briefing on current situation and resources status are provided first. Next, specify resource needed by Division-Groups, specify operations facilities and reporting facilities, place resource and personal orders, and consider Communications, Medical, Safety, and Transportation Plan requirements, etc.

In the last phase, the Planning Section Chief is responsible for seeing that the Incident Action Plan is complete and accurate. Then, select operations shift briefing location; attend the operation shift briefing. In this shift briefing, all pertinent personnel will receive the Incident Action Plan, and a brief rundown on incident as of current time is given and shown on display map.

Several characteristics will play important roles in related system design in the workflow. From the workflow, we can find out that a lot of forms, maps and reports are needed to record the work status, resource management and distributions, and work assignments. These resources are located in different places and departments, like communication, safety, medical, transportation, etc. The participants for the planning meeting have to prepare a lot of forms, maps and reports to present their information. The participants coming from EOC are limited in some special field like fire, medical, or other technical fields so that the evaluation of the incident status may be limited as well.

According to the problems mentioned above, the system functions with an ability to solve these problems will be derived and implemented in the next section.

3.2 System Functions Analysis

First, a collaborative GIS system should possess basic file/database management functions. In EOC, for example, a lot of information including work assignment, resource management and distribution, etc. needs to be input into a database. Therefore, these kinds of functions are necessary for collaborative GIS application. Detailed functions are listed as follows:

1. Forms/tables can be created, read, written and deleted.
2. Forms/tables can be searched and browsed.
3. System access right assignment and management.
4. Work report design and output.

Second, a collaborative GIS system should possess basic GIS functions. In EOC, a lot of information is related to map or location. Some examples include: (1) how to locate the emergency event so that the control line can be determined; (2) how to locate the damage facility and identify the type and amount of damages so that an accurate assessment can be obtained; and (3) how to trace the supplies (medical, food, water, etc.) so that appropriate amounts to be assigned to the shelters. These GIS functions are listed in details as follows:

1. Map display and query
2. Map zoom-in, zoom-out, Pan
3. Map attribute query and edit
4. Extendable map analysis function, like buffer analysis, pipeline network analysis, etc.
5. Load all kinds of map format from different data source.

Third, a collaborative GIS system should possess collaborative functions. In EOC, the emergency personnel will face many hard situations in which the decision is not easy to make because much special knowledge is needed. Therefore, many specialists or experts are needed to help the emergency operators to make the decision. At the same time, the specialists and emergency operators are required to work and communicate on the same system in distributed places. These kind of collaborative functions are detailed as follows:

1. Shared view, control and object selection of geographical information.
2. Annotation and mark-up of geographic (map) features with multimedia data in the form of text, graphics, photos, and audio/video clips;

3. Interactive exploration of geographical data for spatial problems;
4. Awareness of other collaborators and their outcomes.

3.3 Data Source Analysis

In EOC, the data can be categorized into several types according to data contents, data access privileges and data access approaches. These categories may affect the design of system.

One approach is to categorize the data into map data and no-map data according to the data contents. The map data includes all kinds of GIS format, for example shape format files, MapInfo format files, high-resolution satellite imagery, ArcSDE database, etc. The size of the data may change from several thousands bits to hundreds megabits. For example, a small city road map in shapefile format is a small file, while a QuickBird image on this area could be hundreds megabytes. These data usually need GIS tools to handle. Non-map data includes all kinds of forms, tables, pictures, and even multimedia data. These data can be handled with common developing tools and are easily retrieved in common RDBMS.

Another approach to categorize the data is with internal data and external data. The internal data is created, lived and can be accessed by EOC. For example, the work plan assignment, schedules are saved in the system in the EOC.

The external data is coming from other data source. The data is possessed by other organizations. These data are not available to access until some responsible authorities have to be assigned to the EOC. For example, a city underground pipeline data can be possessed by the municipality of

the city. Some agreement with the municipality of the city has to be reached if the EOC wants to access this data.

Data can be also classified through how to load the data. For example, file based data and connection based data. File based data, for example Shape files, image files, and DBF format files, can be read and written through loading the whole data, while RDBMSs or Web Map services are accessed through a connection. Especially in web-based, replicated computing environment, handing single file is totally different from handing database and data services.

The design of collaborative GIS has to consider how to handle these data sources separately because all the data whatever map data, non-map tables, single file, database, inside data and outside data are so different that different design solutions are needed. These problems will be discussed in the next two chapters.

Chapter 4 Architecture Model Design for Collaborative GIS Application

The architecture of a software application characterizes the components of the application, the function implemented by each component, and the interaction among these components [Shaw, 1996]. It is more important and complex in the design of collaborative application because there are more interactions between users and related components.

4.1 System Consistency and Event Distribution Model

As discussed in section 2.3.2 those collaborative systems, especially in replicated systems, confront consistent situations. There are two main strategies to deal with emerging inconsistencies in groupware systems: avoiding or allowing the emergence of inconsistency. In this section, how to avoid inconsistency will be discussed.

There are two approaches in which collaborative systems can avoid inconsistency: Ordering and Locking. Locking approach can be used through floor control and session management which will be mentioned in the next chapter. Ordering approach can be used through controlling event distribution orders.

Figure 4.1 describes two approaches about event distribution when Client 1 operates the application and wants to send event messages to other clients (Client 2 and Client 3). Approach (a) will cause inconsistent situation when Client 2 also is operating the application at the same time.

Approach (b) will not cause inconsistency because the Client 1 first sends the messages to a Server and then the Server sends the messages to all the clients. Because of the server, all the messages produced by the clients will be broadcasted with certain order.

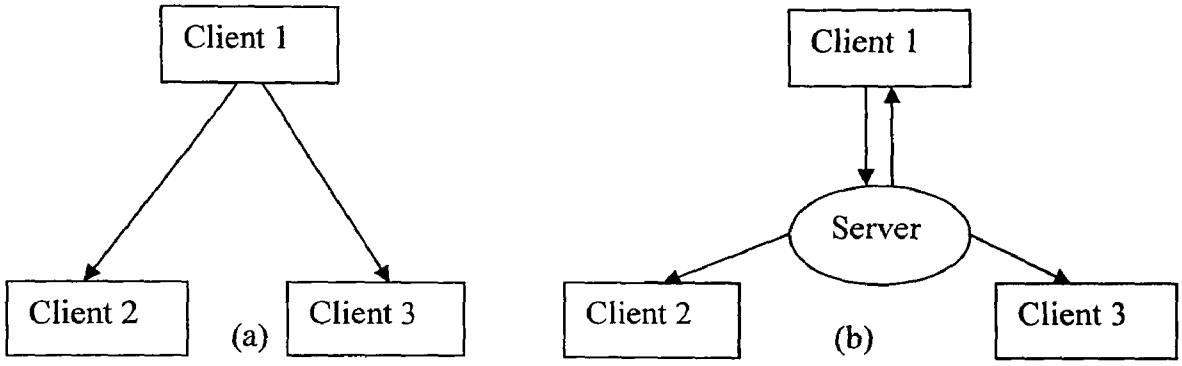


Figure 4.1 Event distribution ordering

How the inconsistency is avoided with approach (b) through the example can be seen in Figure 2.10 and Figure 2.11. Suppose that Client 1 wants to add 1 to variable S (equals 2 at the beginning) and at the same time Client 2 wants to multiply variable S with 2. If approach (a) is used, because of the interaction between Client 1 and client 2, variable S in Client 1 may be 6 (Action Order: Add 1 then Multiply 2 (2+1)*2=6), while variable S in Client 2 may be 5 (Action order: Multiply 2 then Add 1 (2*2) + 1 = 5).

If approach (b) is used, the example is shown in Figure 2.11, when Client 1 wants to add 1 to variable S, the request is sent to the server, and then sent to all clients. The Client 1 and Client 2 follow the same order: either Add 1 and Multiply 2 or Multiply 2 and Add 1. Client 1 and Client 2 get the same value of variable S.

4.2 Hybrid (Semi-replicated) Model

The main advantage of the centralized architecture is its ease of synchronization of shared data. Because there is only one single copy of the shared data, concurrent accesses can be easily serialized by the server process. It is also easy to handle later comers because the server in a centralized architecture can maintain the membership of collaborating processes and notify the existing clients about the new client and transfer the state of the shared data. The main disadvantage of the centralized architecture is that it requires higher bandwidth to distribute display information to all end-users, which may not be practical for widespread use on the Internet. Another disadvantage is not easy to modify current single user system to collaborative system.

On the other side, the main advantage of the replicated architecture is its performance and scalability. Because the output of the application is also the event messages which are smaller than display graphics. According to Smith [1996], analysis of network traffic for Kansas,W an X-based shared environment, found that the ratio of graphics to events was nearly 10:1. Another important advantage of the replicated architecture is to make the collaboration transparency possible. Applications developed for a single user may be used collaboratively by modifying either the application or its runtime environment. After modification, multiple users may share the view and interact with the application. An environment that provides this application-sharing capability is called a collaboration transparency system because the shared single-user application is "unaware" that more than one user is interacting with it. The main disadvantage is to maintain consistency among application copies.

According to Roth and Unger [2000] the main problem with centralized model and replicated model is that they see the application as a whole, meaning that no component-division, is provided and no attention is given to different aspects such as specific group-oriented services and GUI-specific services.

As far as collaborative GIS application is concerned, it is composed with several components according to the analysis of requirements: GIS component, data source component, collaborative component, and interface component, etc. Some components have to become shared components because of the specific features. For instance, data source component could be a high resolution satellite image file or an Oracle database which are impossible to be replicated into end-user machine to carry out collaborative activities. These kinds of data sources have to be shared by other clients. On the other hand, some components like interface components, even whole GIS components (excluding database) could become a replicated components because of the complicated interaction between end users. Figure 4.2 shows conceptual hybrid architecture. Data proxy server, which is used to handle all kinds of data source, is a shared component. Other components, like collaborative component, GIS component, GUI component are replicated in every client.

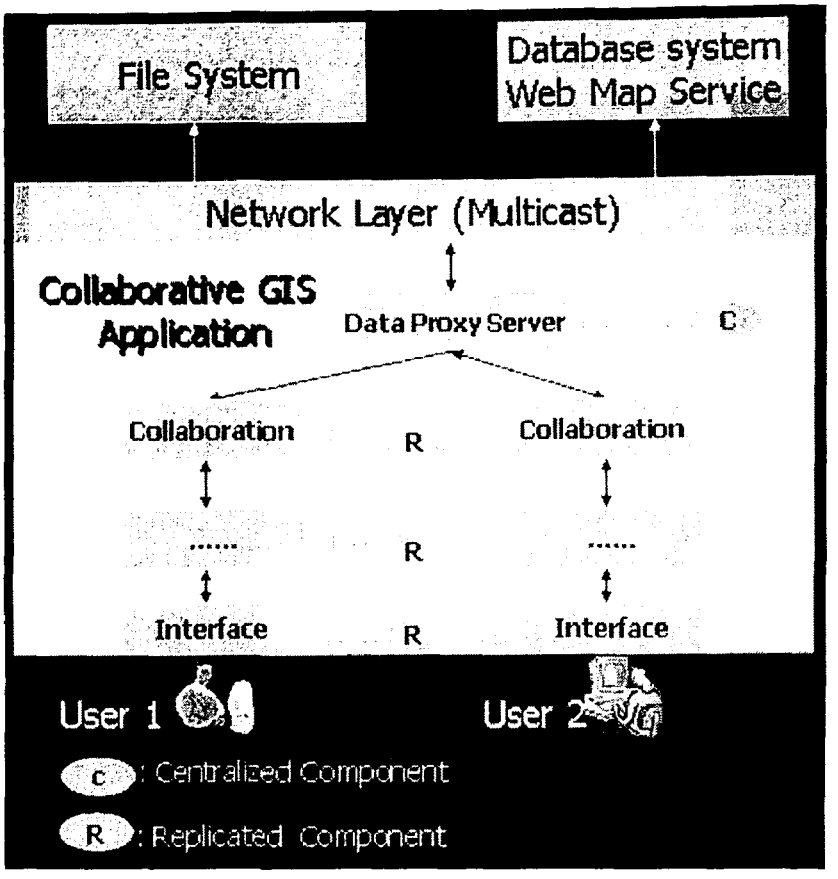


Figure 4.2 Hybrid architecture for collaborative GIS

Following will introduce hybrid architecture to implement a collaborative GIS application: *GeoLink*. A synchronous collaborative GIS application could be a client server structure. The client is a replicated part in end user's machine. The collaborative server is a shared part in a server machine.

The client or replica is composed by two components: GIS component which handles GIS functions, and Collaboration component which handles replica collaborative functions including floor control, session management, message sending, parsing, and reconstructing.

The server is a very simple application which provides a web address, for example IP address, and session name so that each client who wants to join this session could connect to the server by inputting the IP address and session name. The main function is to receive the incoming messages from replicas and broadcast them to the identified replica or client.

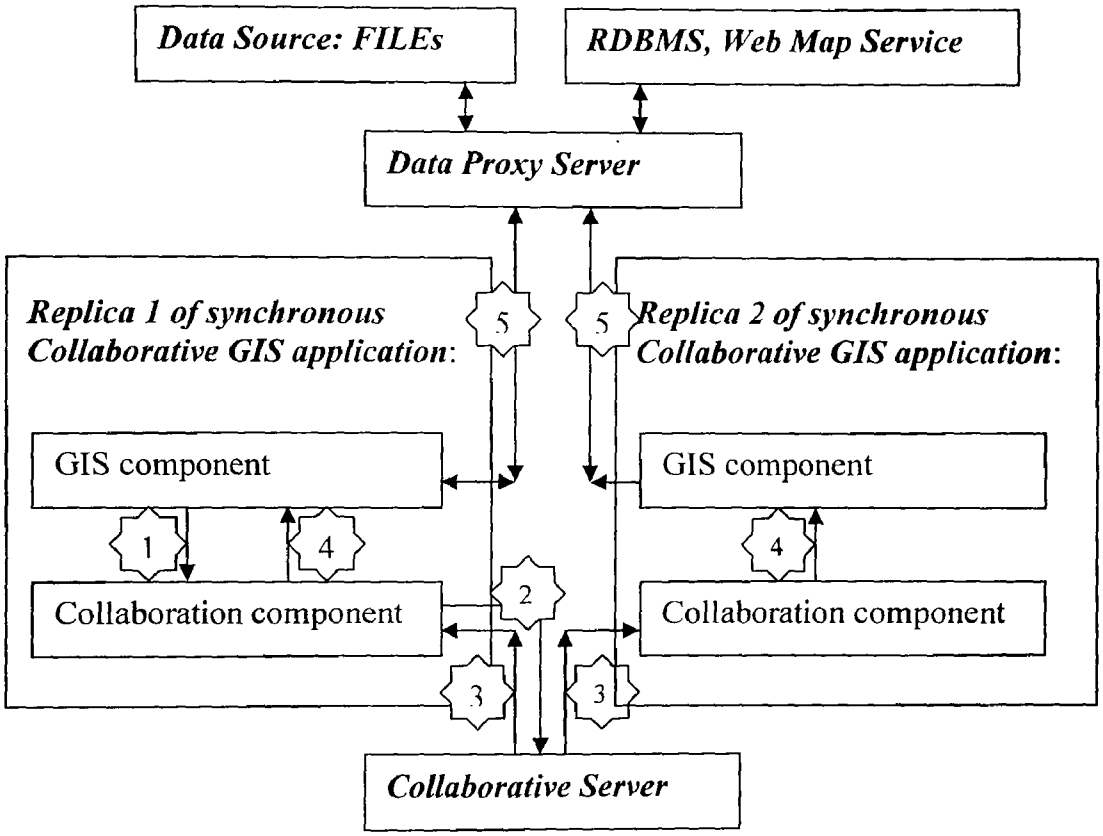


Figure 4.3 Hybid architecture in synchronous collaborative GIS application with a single server

The process of message distribution based on the hybrid architecture is shown in Figure 4.3. The brief description of the process is as follows when Replica 1 operates its interface:

1. When a user, for example Replica 1, operates GUIs of Java applets or client application that handle GIS contents in one client, the event messages are first triggered and sent to the collaboration component at the same client.
2. Collaboration component serializes and sends them to the collaborative server in the server side.
3. The collaborative server distributes the messages to the collaborative component of every client who joins the same session including the client who creates the event.
4. After reconstructed, these events are finally sent to the GIS component to carry out the same functions as host client does.
5. The GIS component may go to single shape file or Web Map Service with data proxy server to load data.

4.3 Peer to Peer

Since the server is a light-weight application, it can be embedded in the client part or replicated part (see Figure 4.4). Therefore every client has both collaborative server and collaborative client. Only one user can be a real server to handle incoming messages in a session. Other users who are in this same session will communicate with the server.

This approach helps peer-to-peer collaboration instead of a fix server. Any client who tells its IP address and session name to other clients can become a server, and then other clients can join the session.

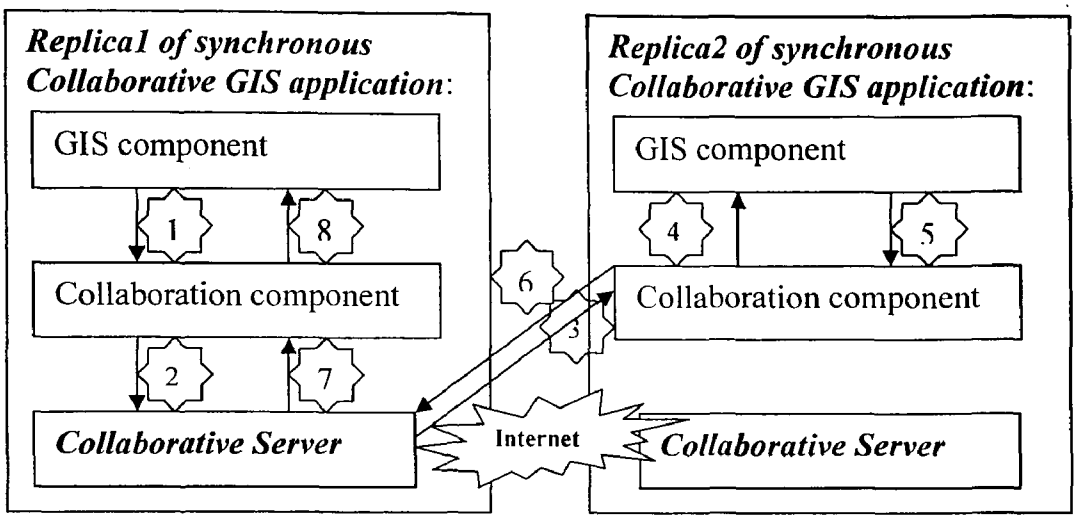


Figure 4.4 Peer-to-peer architecture in synchronous collaborative GIS application with multiple servers

The problems come from the deployment of the application. As we know, if a copy of the client is downloaded from the website, according to the security policy of Java, all the messages have to come from the same host. Some special efforts have to be implemented to carry out this kind of peer-to-peer model. However, this approach supports location independent ability: the user need not know where the server is, which has potential functions in mobile, cell phone, and games.

The event distribution model can be simplified as follows (Figure 4.5):

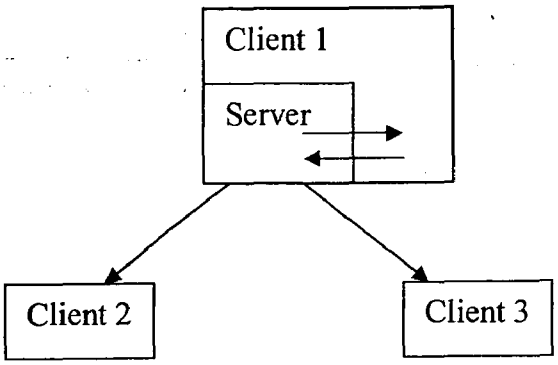


Figure 4.5 Peer-to-peer event distribution model

When Client 1 operates the application and cause event messages, the messages are first sent to the server which may be embedded in one of the client. The server then sends the messages to every client.

4.4 Single-user Application Support and Collaboration Transparency

Most of the GIS applications are single user applications. It is a big challenge to modify these applications to multi-user collaborative applications. How to develop a collaboration-transparent application is the key problem. Li [1999] classified three main techniques to solve this problem (see Figure 4.6). Approach (1) slightly modifies the application code to relate to the collaboration framework class. Approach (2) substitutes the underlying window system or graphics toolkit. Approach (3) interposes an agent to control the system event queue between the application and the window system. Each approach has its benefits and shortages. *GeLink* uses a hybrid approach that combines (1) and (3). Some components use approach (1) and others use approach (3).

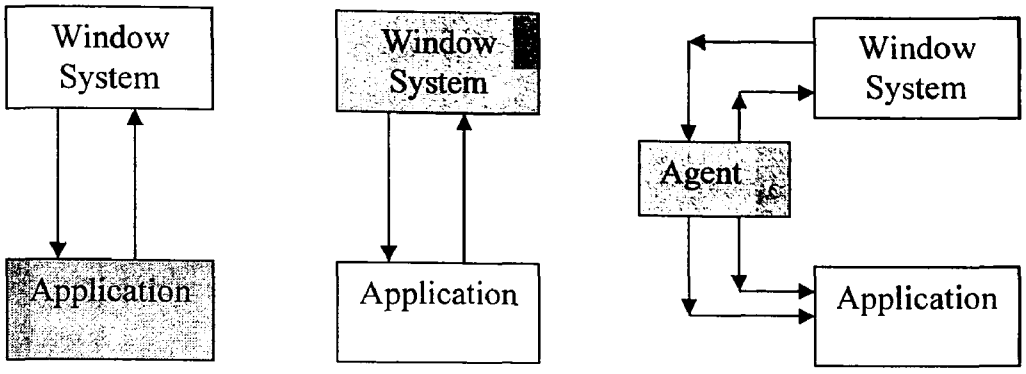


Figure 4.6 Common techniques for collaboration-transparent application

As we discussed in section 4.2 that the replicated client has two components: GIS component, and collaborative component. GIS component possesses mainly interactive GUIs and is the main workplace for the session participants to work together. Most collaborative operations, for example, view and zoom to the hotspot, annotation, buffering and so on, happen in this component. It is very important for the participants to view others actions even just moving mouse. This component uses approach (3), which will be a pure collaboration-transparent component. The agent could be a pseudo-layer in which every mouse event or key board event will be captured and sent to other clients. The pseudo-layer is a Glass pane in a frame in *GeoLink* (see Figure 4.7).

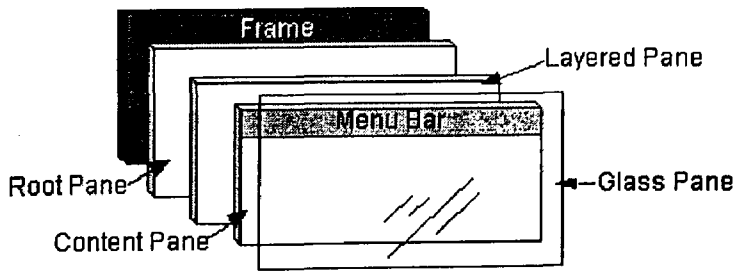


Figure 4.7 Glass Pane [Sun, 2005]

The GIS component intercepts all the user events (mouse, keyboard, input focus events) using a transparent GUI component called Glass Pane, which is available in the Swing toolkit. It is at the topmost Zorder to cover the bean's GUI area and intercepts all the user's events, without occluding the underlying Java component. The Glass Pane can also be dynamically shrunk or expanded, thus allowing easier management of public and private areas of the same workspace. An additional benefit is the accompanying visual effects on the remote sites as well (e.g., mouse click results in a depressed button).

The Collaborative component handles replica collaborative function including floor control, session management, message sending, parsing, and reconstructing and so on. This component uses approach (1). Only command event, like button click, can be sent to other clients. Other events like mouse moving will not be responded.

A Single-user GIS application therefore can be modified to collaborative GIS application through the following three steps:

1. Add the collaborative component into the single-user GIS application.
2. Add the class pane over the main user GUIs in single-user GIS application.
3. Integrate the last two steps.

4.5 Message Structure

The messages which are sent among the clients are divided into three types according to the purposes: Event messages, Floor Control messages and System Environment messages. Through these three types of messages, all clients obtain other clients' information and synchronously work together.

Event messages are triggered by mouse and keyboard when a client operates GIS interfaces. For example, when a user clicks the mouse to zoom a map, the click event which includes the x and y position of the mouse will be sent to the collaborative server, and then sent to every client. The client then obtains the messages through collaborative component and reconstructs the event message to implement a GIS operation.

The structure of Event Message includes Client ID, Event Body, and Message Type. Client ID identifies the original client or host client number; Event Body is the class of Event including mouth event, keyboard event, etc. Message Type shows which type this message belongs to. Here is labelled as "Event Message".

Floor Control Messages include Message Type, Floor Control variables and Clients information. Floor Control Variables include the telepoints status, client operation status, conversation status, etc. Clients Information includes client name, user ID, etc. Message Type is labelled as "Floor Message".

System Environment Messages include Message Type, Data Source, Map Layer, Map Legend, Zoom Scales, Menu Status and Windows status. This kind of message is used to synchronize the late coming clients. When the new client joins a session, the clients who are already in the session will send the message to it. Message Type is libelled as "Sys Message". Following is the example about how these messages are organized and preceded in GeoLink.

```
// class for the messages
public class GeoLinkmessage {
        private String aClientID = null;
        private String aServerID = null;
        private string aMessageType = null;
        // constructor
        public GeoLinkMessage(string clientID, string ServerID, aMessageType MessageType )
{
                aClientID = clientID;
                aServerID = ServerID;
                aMessageType = MessageType
        }
        // get Client ID
        public string getClientID(){
                return aClientID;
        }
        // Set Client ID
        public void setClientID(string ClientID){
                aclientID = ClientID;
        }
        ......
}
// event message class
```

```
public class EventMessage extends GeoLinkmessage {

        MouseEvent aMouseEvent = null;
        // constructor
        GeoLinkmessage (MouseEvent MouseEvent,string clientID, string ServerID,
aMessageType MessageType ){
        super(string clientID, string ServerID, aMessageType MessageType);
         aMouseEvent = MouseEvent;
        }
        // get Mouse event
        public MouseEvent getEvent(){
                return aMouseEvent;
        }
        // set mouse event
        public void setEvent( MouseEvent e){
                aMouseEvent = e;
        }
        .....
}
```

Chapter 5 GeoLink: Prototype Design and Development

Based on the conceptual model design mentioned above, a collaborative GIS application framework, *GeoLink*, is implemented. In this chapter, the first section is to select suitable developing tools both in CSCW toolkits and GIS tools. Then, a framework of the prototype is given. Some specific problems confronted for collaborative GIS are discussed and the related solutions are provided. The deployment of the prototype is introduced. Finally, the development experience of this prototype is presented.

5.1 Selection of Developing Tools

Different toolkits have their advantages and disadvantages. Which one is better mainly depends on what systems are implemented. As far as synchronous collaborative GIS applications are concerned, following issues need to be considered:

- Is it suitable to work on the Internet?
- Is it easy to work together with other tools like CSCW tools, GIS tools, 3D tools and other multimedia tools?
- Is it still active? Because if it is not active, it may not compatible to current developing tools.
- Is it simple to implement?
- Can the toolkit be used in a desired integrated development environment (IDE)?

After research and test, Java with JBuilder as main developing language is adapted; Java Shared Data Toolkit (JSDT) is used to develop collaborative functions; MapObjects is used to develop GIS functions.

JSDT, developed by Sun Microsystems, Inc. [Sun Microsystems, 1999], provides a development library that allows developers to add collaboration features to applets and applications written in the Java programming language. With the JSDT toolkit, it is possible to have three implementations using TCP/IP sockets, reliable multicast frameworks (LRMP or RMF/RAMP), or HTTP protocol. JSDT uses a centralized-server architecture that consists of four main objects: session, client, data, and channel, where a ***session*** is a conceptual collection of ***clients*** which communicate ***data*** (array of bytes) through ***channels***.

An important reason of selecting the JSDT toolkit is because of its "openness" and "transparency", meaning that it follows open standards and does not require that the developed applications be worked with some kind of proprietary "engine" (e.g., a meeting engine) or installation of any software parts (other than the developed components) on each participant's computers.

Unlike the previous toolkits, the JSDT is Java based, which means it can be used on any system that supports Java. This is a significant advantage over the others since many groupware applications need to run on heterogeneous platforms. However, unlike GroupKit, JSDT does not provide any groupware widgets that allow for easy construction of multi-user GUI elements.

GIS toolkits are collections of software components, targeting at GIS developers not end users to develop customized GIS applications. Both commercial and open source toolkits are available. In addition, software components linked to specific vendors software, which may not separately sold as toolkits, are also available together with that software license, such as the components with Intergraph's GeoMedia Suite technology. Other GIS toolkits are by-products of Java graphics toolkits, such as J/GIS from Interactive Network Technologies, MapObjects-Java Edition from ESRI and JLOOXGis from LOOX Software Inc. These toolkits provide support for the development of Java applets, applications or servlet and enables access to data from multiple sources.

MapObjects-Java Edition is a powerful collection of client- and serverside components that developers can use to build custom, cross-platform GIS. The key features include following aspects:

1) MapObjects- Java Edition has ability to combine multiple distributed data sources, for example, shape files, ArcSDE, ArcIMS image and feature services and all kinds of image formats, etc.
2) It has a wide range of GIS capabilities including thermatic mapping, multiple map layers, specifying projections.
3) It has ability to deploy applications over the Internet.

5.2 Framework of Collaborative GIS Application: GeoLink

GeoLink is a light-weight, object-oriented and extensible application. The application was composed of three tiers, replicated client tier, shared server tier and data tier (see Figure 5.1).

The replicated client is downloaded from a known web server. Every user who wants to launch GeoLink will get the same client. This client is composed of several components including GIS component, collaborative component, and multimedia conferencing component, etc. the GIS component obtains basic GIS functions. The collaborative component is responsible for the synchronous collaboration between the users. The multimedia conferencing component helps to communications among users by audio, video, and chat.

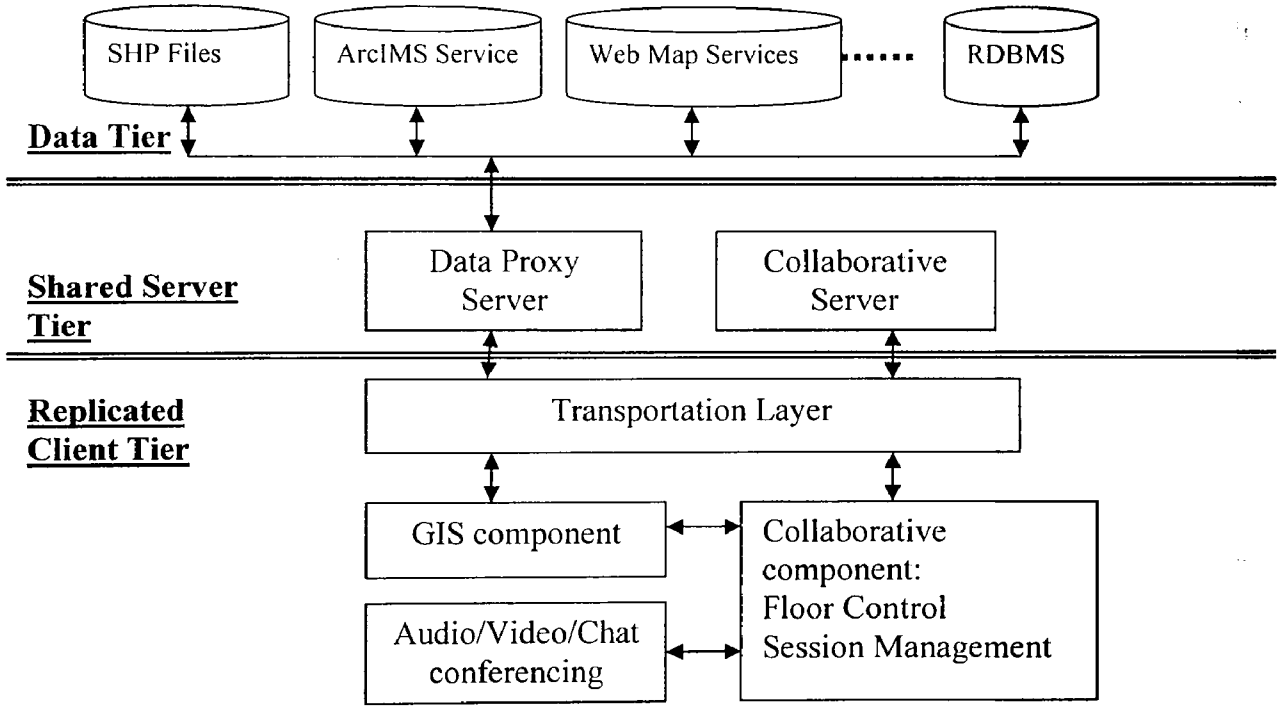


Figure 5.1 Framework of *GeoLink*

The shared server tier includes two servers: Data Proxy Server and Collaborative Server. These two servers are run in a known server machine, and are shared by all clients. The Data Proxy Server handles data source related issues including data source connection, data consistency

handling, data access and retrieve, etc. The Collaborative Server receive incoming messages and multicast to others clients.

The data tier is the data sources that GeoLink can handle through the Data proxy Server. There data sources include SHP files, rational DBMS, like Oracle, SQL Server, ArcIMS Services, Web Map Services, etc. How to handle these data sources will be presented later.

The original single GIS application also can be extended into collaborative GIS application through adding two components separately in the Shared Server tier and the Replicated Client Tier: Collaborative component and collaborative Server.

Following presents how *GeoLink* is working for the first time:
First of all, a special message created as the instance of Data is sent through network to create the first session. The collaborative server then precedes these messages and creates the specific session and channel. Third, these clients may register in the session to communicate with other clients. JSDT which is embedded in collaborative component and server component in *GeoLink* system looks like a bus on which messages are loaded and sent to the other clients.

5.3 Considerations for Collaborative Interactions

Several specific challenges confronted with the complex task of building multi-user, multi-computer systems on top of single-user, single-computer infrastructures. Due to the specific nature, the development of synchronous collaborative GIS involves many issues that introduce

additional technical problems compared to the development of single-user applications and conventional distributed applications.

Several specific problems have to be considered when the detailed designs and developments collaborative GIS begin. These problems include system consistency, latecoming, floorcontrol, Feedback, Feedthrough and Awareness, data transportation and network, and the selection of developing tools.

5.3.1 Latecoming

The term latecoming is used to denote a process which allows latecomers to join and participate in an ongoing session. Two tasks have to be ensured by the collaborative application:

- Current state has to be transferred to the latercomer
- During the transmission of the state, the state of the application has to be in consistency.

The first task related to the problem: how to transfer the current state of the collaborative application to the new comer without disturbing the other users? Two approaches are used in all kinds of collaborative systems: Transparency approach, and No-transparency approach.

The Transparency approach is based on the Model, View and Control (MVC) design pattern [Bushmann, 1996]. There are three kinds of states in the application according to the MVC. If the three states are transferred to a new comer, the new comer will keep synchronous with others. The main purpose of the Transparency approach is to transfer the three states to the new comers.

In order to transfer the model of an application, the current state of the program including program counter and stack frames would have to be captured. Fortunately, if the classes of the model part are serializable, it is sufficient to transmit the member data of all objects automatically according to the serialization mechanism of java.

The view and control parts of the application are realized using UI components and their event processing callbacks (event listeners). The main task is to transmit the state of all currently used UI components. A simple way of realizing this is to transmit the main UI container of the application and in consequence all recursively embedded components. Fortunately, elements of the Swing and AWT object hierarchy are already serializable. But unfortunately, event listeners which are subscribed to UI components are not serializable and therefore get lost or produce undesired exceptions during transmission. The only possibility to fix this problem transparently is to patch the base interface of all event listeners [IIlmann, 2005].

java.util.EventListener

public interface EventListener extends java.io.Serializable {...}

The Transparency approach causes some problems for the limitation of Java, Because people designed Java at the beginning without considering the requirement of collaboration. The big problem is that system can not handle data connection if the system is connected to a database.

Instead of restoring the system state automatically, The Non-transparency approach transfers the system state manually. Any system state, which is necessary for the new comer to restore the system, will be kept in file and transferred to the new comer. Some systems like JASMINE,

record all GUI event to a log file. When the new comer is coming, the new comer loads the log file to restore the system state. The benefit is the new comer can review the session from the beginning. The problem is that the log file may be too big file to transfer.

GeoLink records every necessary state of the system, and transfer it to the new comer. The necessary state includes: data source and data connection, map layers and zoom scales, menu status, windows status, etc.

The second task is related to the consistency of system state. Lock approach is used in this task (see Figure 5.2).

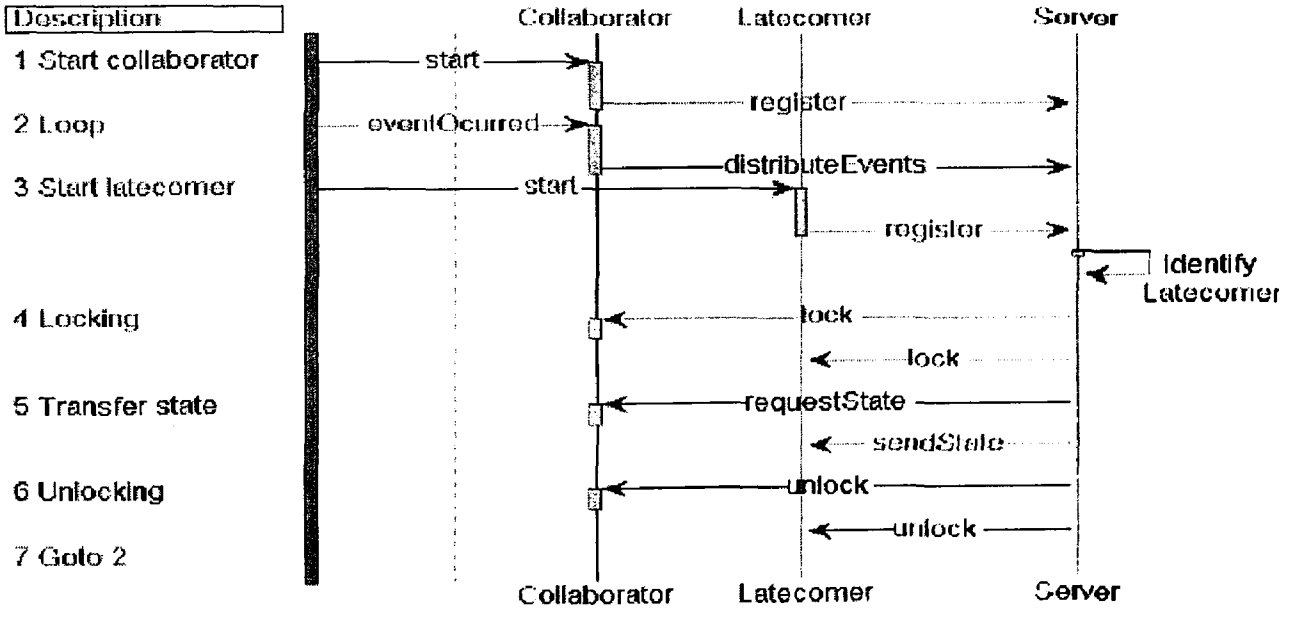


Figure 5.2 Sequence of latercoming

The process of latecoming starts at step 3. A latecomer starts an application while another instance of it is already running (has been executed by a collaborator, step 1 and 2). The application registers at the server and is identified as latecomer. All currently collaborating and

latecoming applications are requested to lock themselves (step 4) as described above. Events that occur during the locking phase are buffered. Next, the server requests the current state of the collaborative application (step 5). The server chooses one collaborator as the one that transmits its current state. Further possible latecomers may integrate in the current latecoming process until the state is not completely transferred to the sender.

As soon as the state is received, the server distributes it to all registered latecomers. In step 6, all applications are unlocked. This includes that they are made visible again and get possible buffered events.

5.3.2 Floor Control

"Floor control" refers to the management of interaction among participants in meetings. This comes from expressions such as "who has the floor" or "yielding in the floor" in formal meetings. For example in a shared white board, only one participant has the floor and draw at one time.

Myers [Myers, 2001] gives a classification to distinguish the different floor control police based on the procedure of obtaining a floor control: assigning control, request control, and releasing control. By combining these release and request mechanisms, all of the existing floor control policies can be constructed. Following is the examples collected by Myers:

Free-floor: Any participant can enter input at any time, and control immediately passes to that user.

Pause detection: the floor is made available automatically when the user is finished, and the next person to do something gets control. Trying to do something while someone else has the floor is ignored.

Preemptive: anyone can grab control of floor at any time, even while someone else is doing something. This is also called "take floor" [Inkpen, 1997].

Fair Dragging: Boyd [1993] classifies floor control mechanisms among a number of dimensions, and introduces "fair dragging," which automatically grabs the floor when dragging starts, and gives pending requests for the floor to users in the order requested [Greenberg, 1990].

Some papers recommend providing users with multiple floor control mechanisms, since different mechanisms might be appropriate for different kinds of meetings and software [Greenberg, 1991, Handley, 1995].

As for the GIS and decision making application, sometimes people, who are involved in the same session, would like to have highly interactive communication on a topic. That means that any person who wants to talk and demonstrate his ideas would like to take the floor control immediately without the grant of a moderator. This is also mentioned as Pause detection. While on the other side, sometimes person who already holds the floor control would like to hold the floor until his presentation is finished. Therefore both two mechanisms are needed in the collaborative GIS and decision making application. I term the two mechanisms as: detection floor and grant floor.

In the detection floor mechanism, there is no moderator and the floor is made available automatically when the user is finished so that every person can take the floor control without grant. The application will know if the user is finished or not through testing the moving of the cursor of the user.

In the grant floor mechanism, on the other hand, there is a moderator who can decide which client has the right to take over the floor control when he receives several floor control requests. While he also have right to take others floor control away and give it to any client. The moderator is always the first client who creates the session.

Following example shows how the floor control helps to solve inconsistent problems. The simple example of inconsistency is when two users simultaneous draw straight lines (see Figure 5.3). Figure 5.3a shows the desired output and Figure 5.3b shows the actual output for the inconsistency. This inconsistency is caused because if events are naively broadcast between collaborators, event streams for nonatomic events such as mouse drags become confused, and can cause conflicts among collaborators.

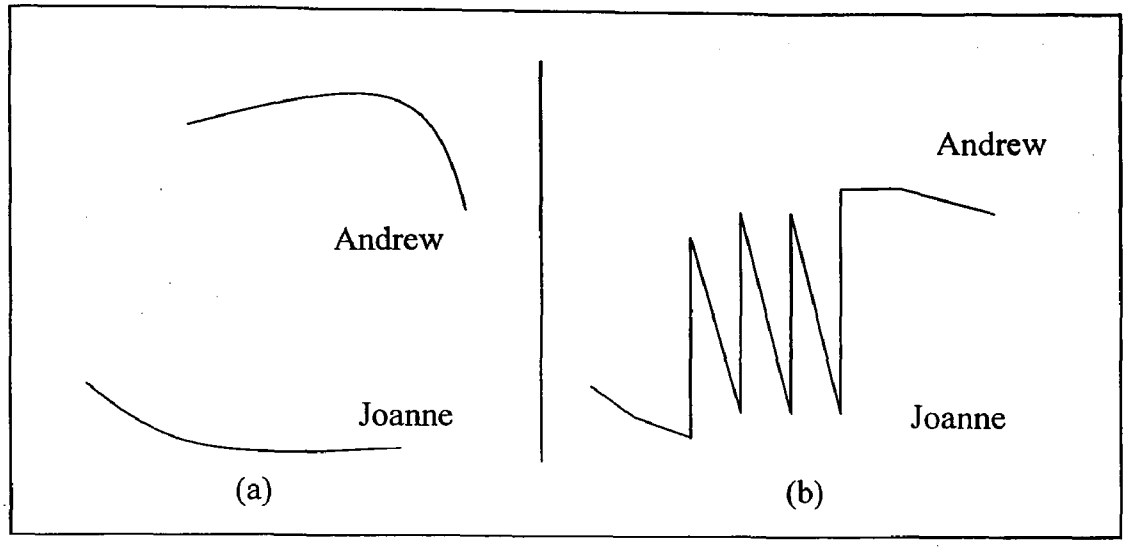


Figure 5.3 Output of inconsistency

In the example of drawing lines, we can see how to solve the consistent problem through detection floor mechanism. Before Andrew can draw a line, he must get the control privilege of the floor. When Andrew gets the control right of the floor, he presses the mouse, and then drags the mouse to a place and at last releases the mouse to finish the line. During this period, the mouse events trigged by Andrew are sent to every client in the session. The constant incoming events will avoid Joanne getting the right of floor control until a timing gap event happens, for example in 5 second no any mouse move event appears. Then, Joanne can have chance to get the control of floor.

5.3.3 Data Transportation and Network

At data transport level, collaborative application requires point-to-multipoint and multipoint-to-multipoint which is named multicast communication. The general idea of multicast is to establish a tree of routers whose root is a router of the sender's LAN and whose leaves are the routers of

the receivers' LANs (see Figure 5.4). This tree is called a multicast tree. A packet transmitted by the sender is propagated along the edges of the multicast tree, with the guarantee that only a single copy of each packet passes over each edge. At each inner node of the multicast tree the packet is copied to all outgoing edges. When a multicast packet reaches a LAN containing one or more receivers, it is broadcast in that LAN.

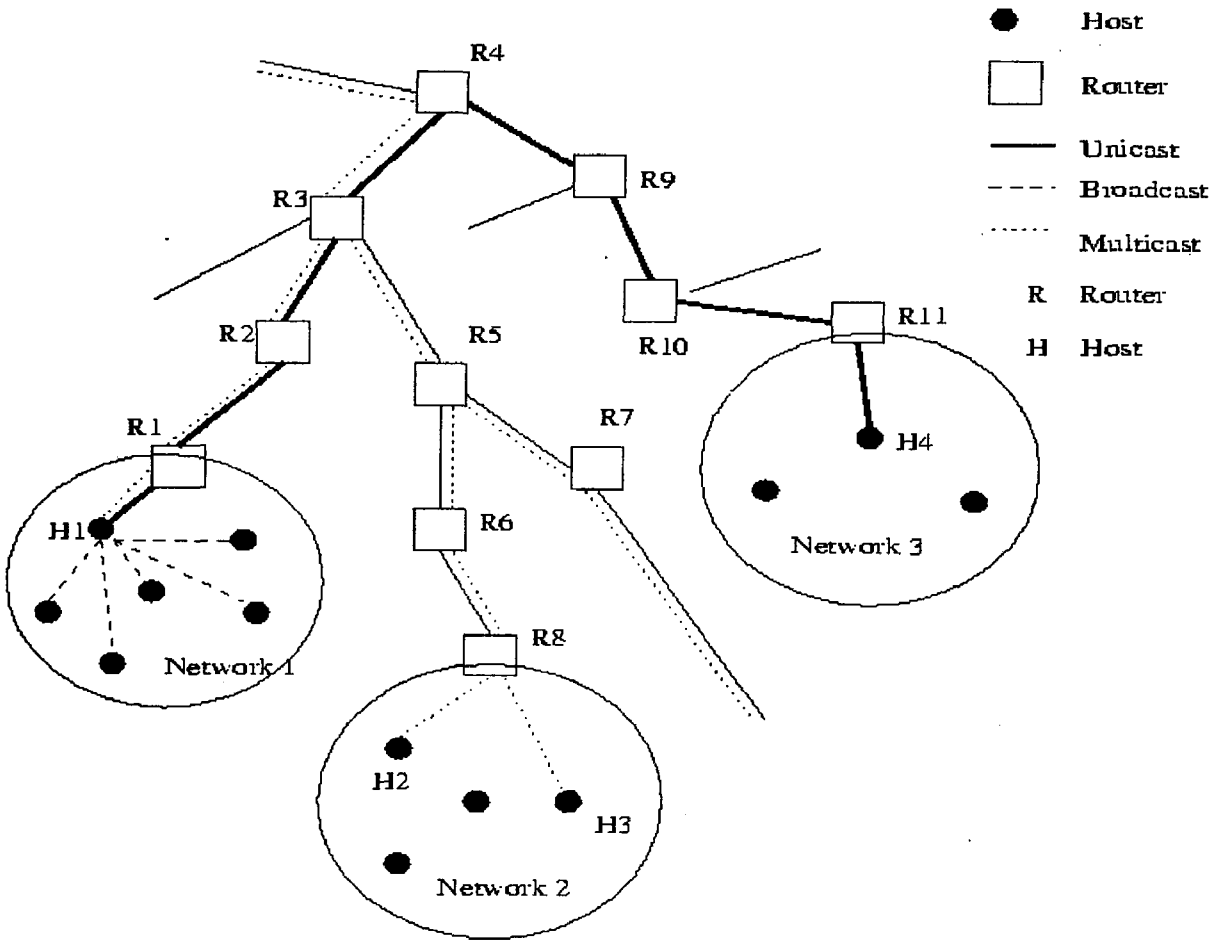


Figure 5.4 Multicast packet deliveries [Deering, 1995]

Unfortunately, there exist diverse multicast protocols which are application dependent. This diversity exists because the best way to achieve reliability for multicast depends heavily on the application.

Currently, the most communication is focused on simulating multipoint communications with many point-to-point messages (e.g., RPC, TCP/IP). But this approach is also inefficient and error-prone.

Fortunately, the usage of transport protocol is simple. The common interface between application and network can be provided so that the application is independent from network protocol and also can be transparent for the application developer.

In the prototype of GeoLink, JSDT is used to provide the communication between users. JSDT provide a common interface for general multiparty communications, beneath which a wide variety of implementation technologies can be employed. In particular, the specific protocol stack used to implement the functionality defined by this toolkit, as well as the negotiation process used to select a specific protocol, are not visible to the user of this interface. Therefore, a range of different protocols can be hidden within the implementation of this interface (including standards-based multi-party communications protocols (e.g., T.12x), custom protocols based on standard networking interfaces (e.g., TCP/IP), and arbitrary proprietary protocols).

5.3.4 Data Source Handling

How to handle the data source for the replicated collaborative GIS application is an important but complex issue. As we have known in Section 3.3, there are all kinds of data sources used in GIS applications. Different data formats may have different programs to deal with. GeoLink considers two kinds of data to handle: file based data, for example, shape files, and connection

Based data, for example Web Map Service. Two approaches handle the two data types respectively: Distributed File approach and Linked approach.

Distributed File approach (see Figure 5.5) is used to transmit single file. When a client wants to load its local file into the system, the file is serialized and transmitted to all the peers who are joined the same session at the same virtue location. This is a simple and effective approach for small size file, for example a shape file format, or any other vector files, while when the size of the file is extremely huge, for example a remote sensing image data which could be several megabits, the time cost of the real time file transmission will not acceptable. The simple solution for the problem is to transmit the file before the session is created. Before the session begins, the plan will be made to identify what files would be used in this session and send the files to all the peers who will attend this session. When a client writes or changes the data, every client will do the same work.

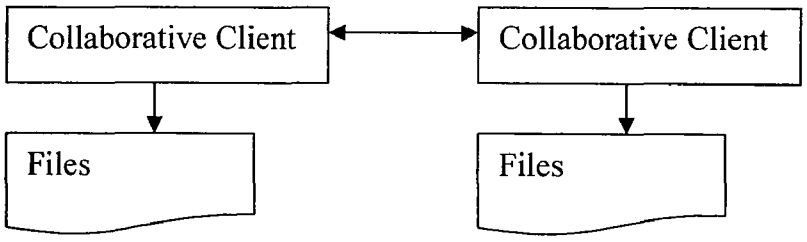


Figure5.5 Distributed file approach

The Linked approach does not distribute data. All the clients in the same session will connect to common data source. This data source is in a special web data server, for instance ArcIMS application server, or any standard web server. When a client need to load data from a web map server, the client send request information to the web map server and also sends the information

to the peers. The peers receive and reconstruct the request information and send them to the web map server. The web server will send the data to every peer. The problem lies in the secure web server in which a user name and password are needed. These kinds of servers are not allowed to be connected via many connections. A proxy server which embedded in replicated peers is design to handle this problem. This proxy server receives every client request, but just sent a request to the data server. The data server responses the request and sends information to the proxy server. The proxy server sends the information to every client. Therefore both read only or write request can be handled with the same function. Figure 5.6 shows how this approach works when Collaborative client1 has a Data Proxy server for the session.

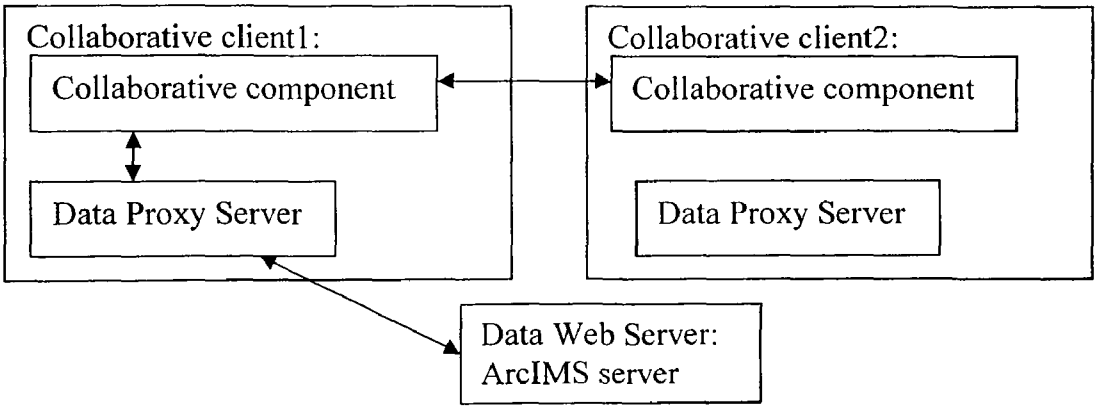


Figure 5.6 The Linked approach

5.4 Deployment of GeoLink

Java Applets, XML/HTML-based clients and Java Web Start are main Java Based Web clients. XML/HTML-based clients are often used to design simple to moderate user interface which is not suitable in highly interactive collaborative GIS. Both Java Applets and Java Web Start can be used to design complicated user interfaces. The two technologies have many familiar features and are even the same for the system developer. While the difference lies in two characteristics:

offline support and subsequent use response. An application can be launched offline. Since the application is downloaded from the Internet, it can be run without through the Internet. The subsequent use response is just seconds if there is no any change happens in the server, because the application launched with Java Web Start are cached locally.

Table 5.1 compares different technologies used for designing Web clients. These different factors influence the design of these Web clients.

Table 5.1 Web-client factor comparison

Factors	**Applets**	**XML/HTML-based clients**	**Java Web Start**
User interface	Moderate to sophisticated	Simple to moderate	Moderate to sophisticated
Offline support?	No	No	Yes
UI response	Network independent	Network dependent	Network independent
Interactivity	Browser limited	Browser/markup limited	Open
First use response	Minutes	Seconds	Minutes
Subsequent use response	Minutes	Seconds	Seconds
Bandwidth	Variable	Fixed	Flexible

usage			
Lightweight client support	Limited	Open	Limited

Since *GeoLink* will be deployed over Internet, Applets and Java Web Start are potential options. Applets option requires that the application have to be downloaded every time when it is launched, while Java Web Start option need not. Applications with Web Java Start will be downloaded once and will not be downloaded again until the application is updated. This feature is more important to *GeoLink* because the replicated client is a little bit fat client which will cost much time if it is downloaded from the server every time.

Java web Start is used in *GeoLink*. A server Web site with a link of *GeoLink* application is built. When a user accesses the website and hits the link of *GeoLink* application, the *GeoLink* application will be downloaded in the use's desktop. The user can collaboratively explore this application with the others through the *GeoLink* Server runing on the web server. The user also can tell his peers its own server information for instance IP Address, Session Name, and Port Number and create a server by itself. These peers can change the register information to join this session.

Figure 5.7 shows an overview picture of GeoLink with five replicas distributed over the Internet and local network.

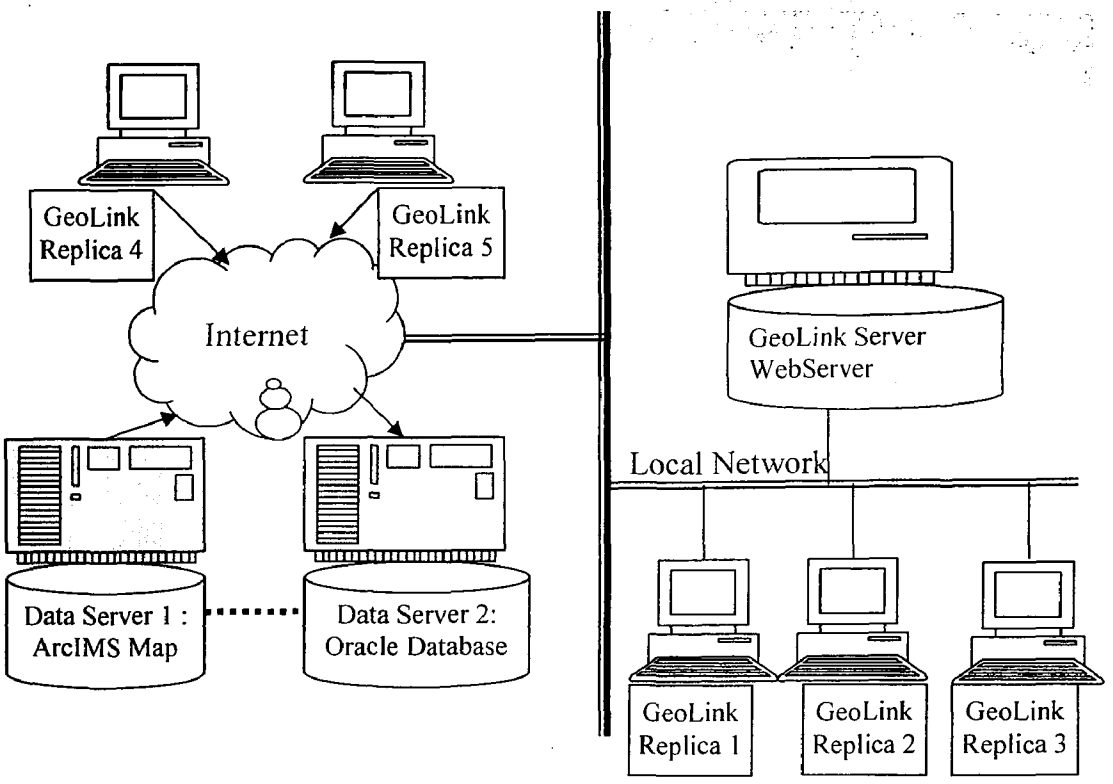


Figure 5.7 Structure of *GeoLink* with 5 replicas and several data servers

5.5 Experience and Evaluation of GeoLink

The prototype of *GeoLink* possesses both collaborative functions and GIS functions. Both kinds of functions are described as follows:

5.5.1 Basic Collaborative Functions

GeoLink Client

1. ***Shared View***

   GeoLink client possess shared view, control and object selection of geographical information. Figure 5.8 shows two users/clients launched in one machine sharing the same map view and observing other operations of the application.

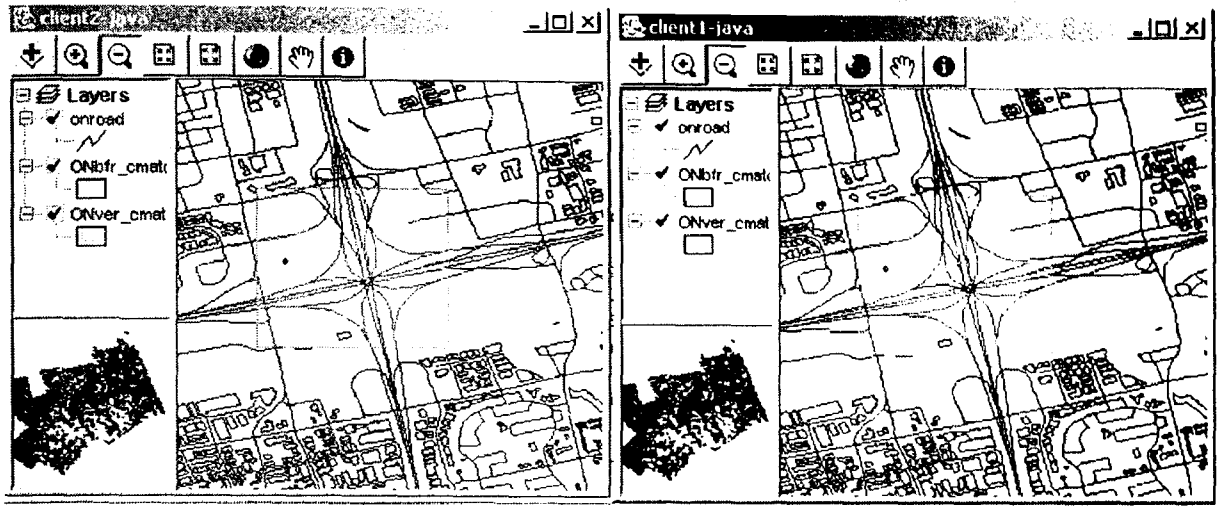


Figure 5.8 Snapshot of shared view

2. *Awareness and Telepoints*

GeoLink client is a collaborative GIS tools with transparent WYSIWIS functions. Any users who join the same session can observe other users' operations to the application. Figure 5.9 and Figure 5.10 show how telepointers work when two clients are launched (User/client 2 on the left side and User/client 3 on the right side) in one machine. Figure 5.9 shows that when the client 2 is operating the application, the client 3 shows the GUI's operation of client 2 through telepointer. Figure 5.10 shows that when client 3 is operating the application and the client 2 observe the client 3' operation.

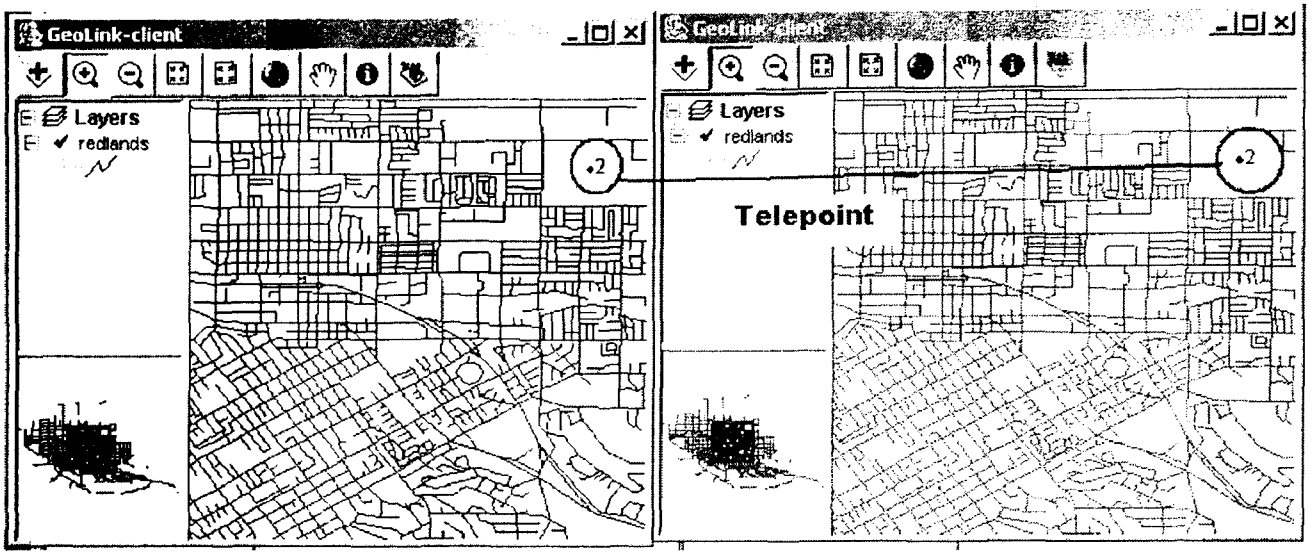


Figure 5.9 Snapshot of telepoints in *GeoLink* when client 2 (left side) is active

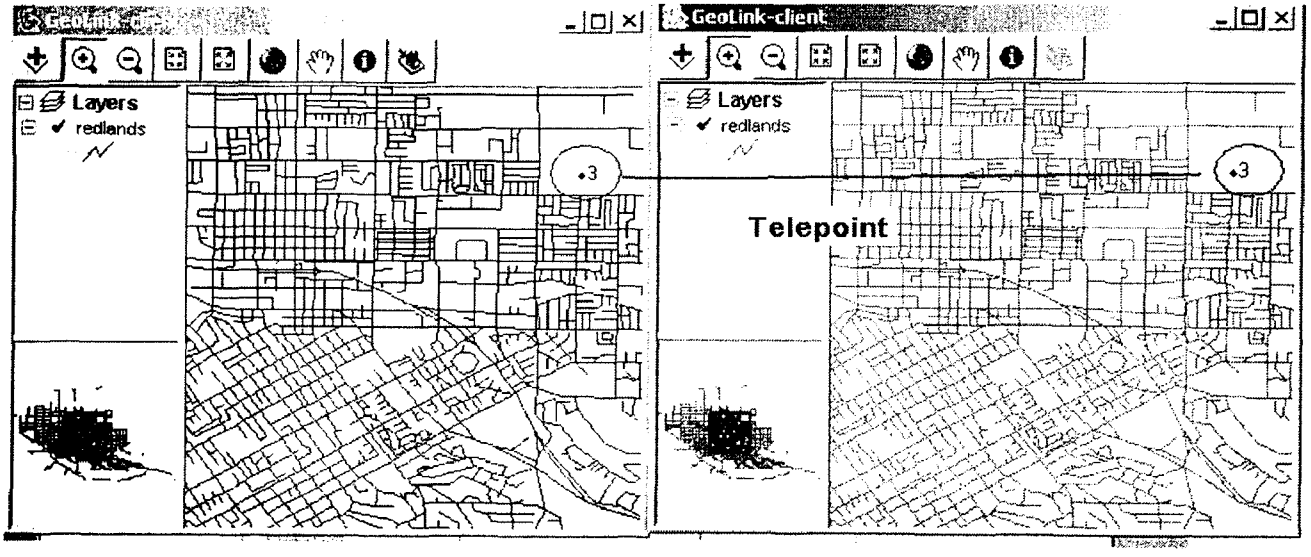


Figure 5.10 Telepoint of *GeoLink* when client 3 (right side) is active

3. Floor Control

The floor control of *GeoLink* (see Figure 5.11) is obtained by the user who is the first one to enter and create the session. The user who is holding the floor control can set a new server,

change the state of telepointers, and view other user's state. Cursor type means that if this user would like to see others' telepointers, it should be set true for it will cause confusion if there are too many telepointers shown in the screen. Operation Status is set true if this user has the privilege to operate the GUI interface.

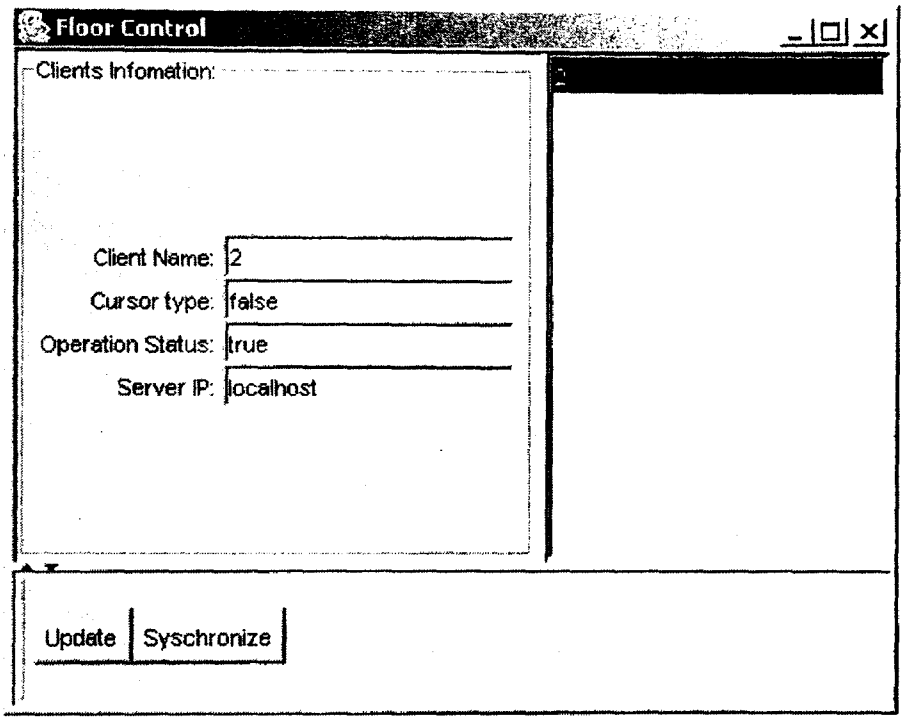


Figure 5.11 Floor control of GeoLink

4. Latecoming

GeoLink support latecoming function. After the session is set up, the latercomer can join the session through request the state of the application.

GeoLink Server

The server of the *GeoLink* can manage the sessions and Servers (see Figure 5.12) in a machine or in different machines. The session is composed with several parts: Server name, Host Name, Host Port, Session Name and Channel Name. These five parts identify the unique session in a machine. The sessions even can be distinguished in one machine through Session name.

GeoLink Server can create a new session through inputting these five parameters. *GeoLink* Server also deletes a launched server, disconnects and connects the server as well.

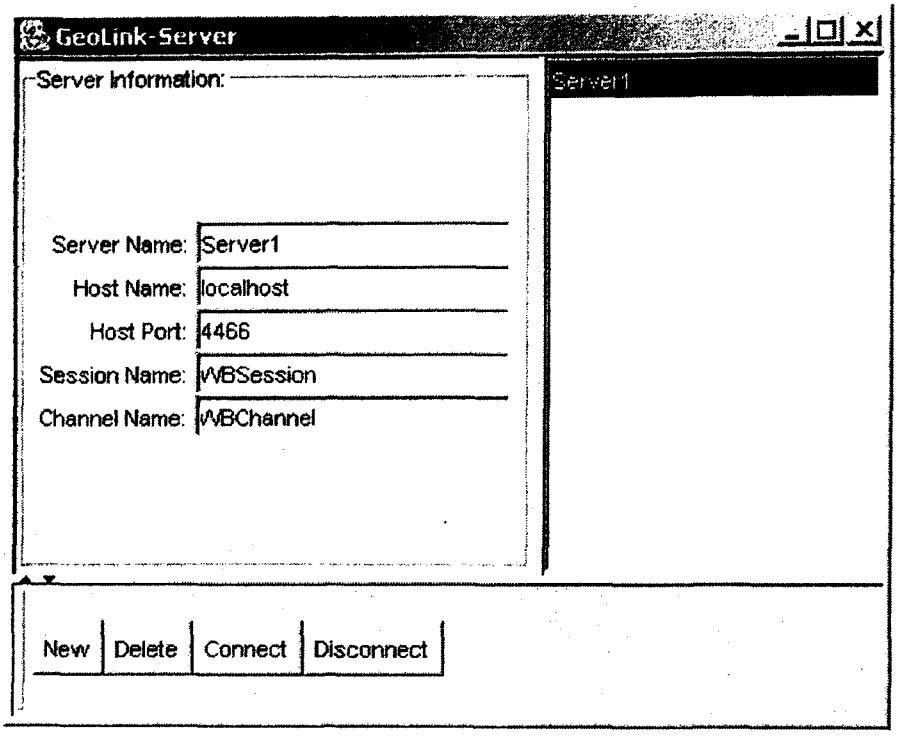


Figure 5.12 *GeoLink* server

5.5.2 Basic GIS function

Zoom, Pan

GeoLink possesses basic GIS map browse functions including zoom in, zoom out, pan, zoom to full extent, zoom to identified rubber windows, etc (see Figure 5.13).

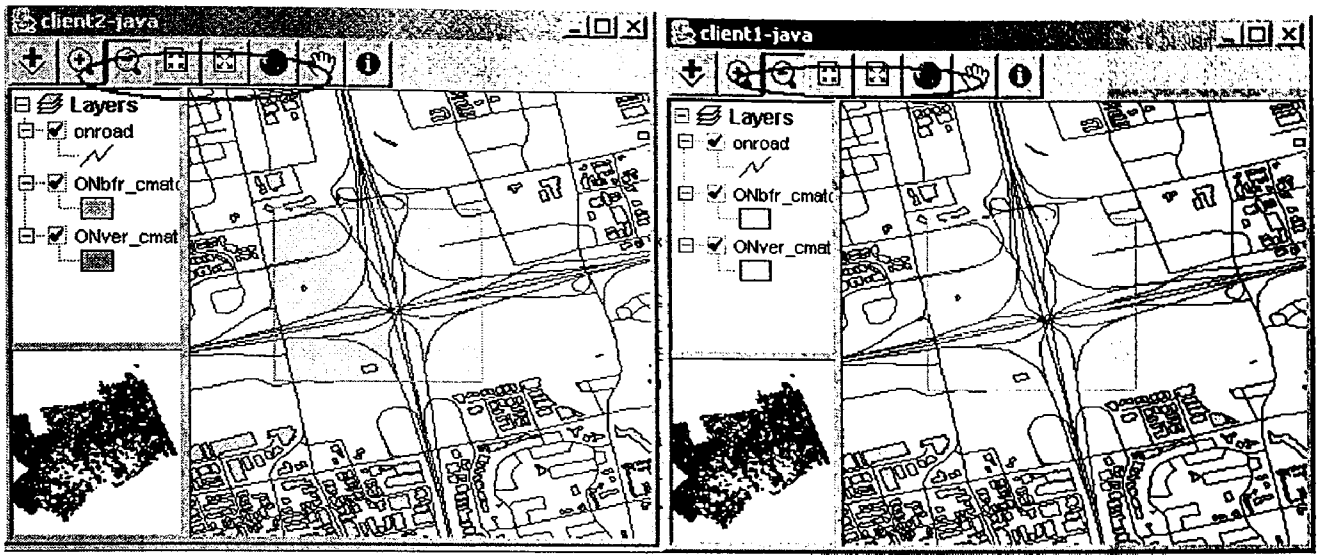


Figure 5.13 Zoom and Pan functions of GeoLink

Identify

GeoLink clients can identify the attributes of the map through clicking on the map features collaboratively. Figure 5.14 shows that one client is viewing the attributes of the features the other client is identifying.

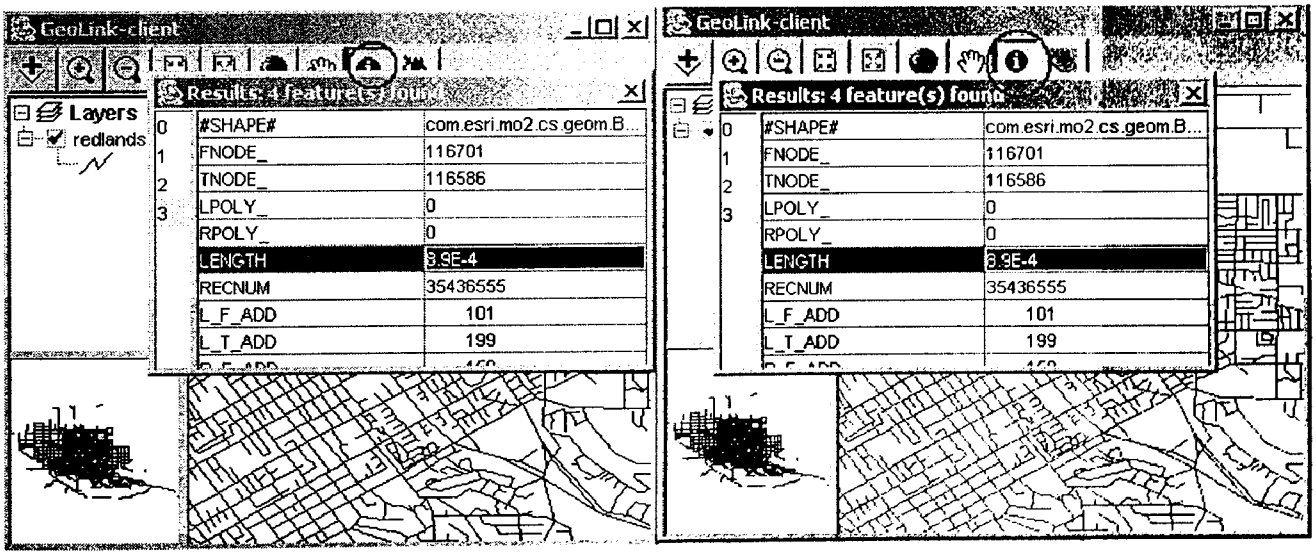
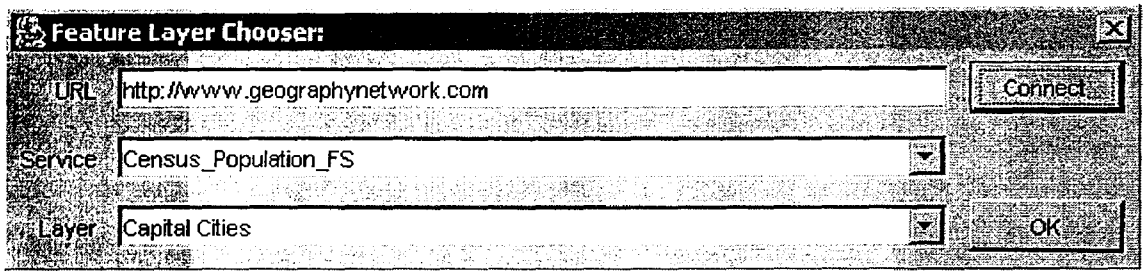


Figure 5.14 Identify map features

Datasource

GeoLink can access two kinds of datasoures: SHAPE format file and public web map service. Figure 5.13 shows that three layers with shape format were loaded into the application. Figure 5.15 shows that different layers in USA were loaded from public map service: *Geographynetwork.COM.*



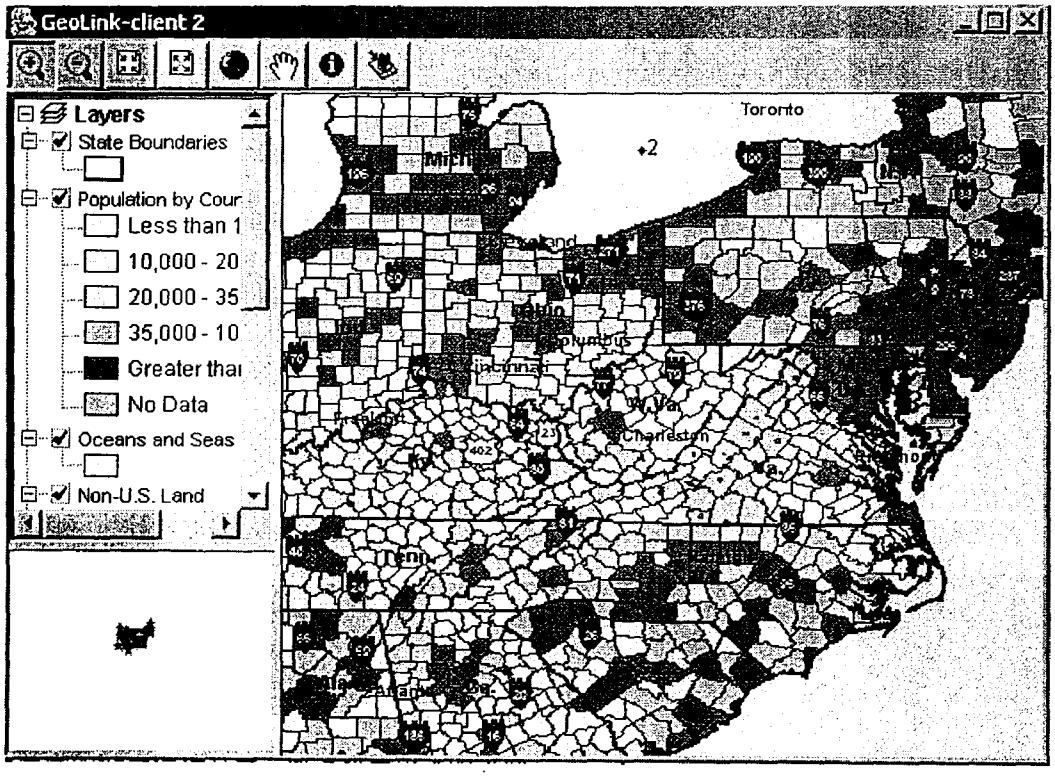


Figure 5.15 Public Web Map Service

5.5.3 Software Testing Issues

Software Testing involves operation of a system or application under controlled conditions and evaluating the results. For example, if the user is in interface A of the application while using hardware B, and does C, then D should happen. The controlled conditions should include both normal and abnormal conditions. Testing should intentionally attempt to make things go wrong to determine if things happen when they shouldn't or things don't happen when they should. Software QA involves the entire software development process - monitoring and improving the process, making sure that any agreed-upon standards and procedures are followed, and ensuring that problems are found and dealt with. It is oriented to "prevention" [Hower, 1995].

Because *GeoLink* is a concept-proof prototype, the standard software development process is not followed. The main test approach is software testing approach instead of software QA approach. The main testing involved two aspects: usability and performance.

The usability testing mainly tests if the functions are implemented correctly. The detailed testing approach is that testers attempt to experience every function to determine whether these functions meet design requirements with different data sets, network environments, operating systems, and hardwares. Table 5.2 shows the detailed testing environments used in *GeoLink*. The results met the design requirements. Future test work will involve more complex environments, for example, more data set, Unix and Linux, PCA, etc.

Table 5.2 *GeoLink* testing environment

Testing catalogue	Testing contents
Data Sets	SHP files, ArcIMS Web Map Services

Networks	Local Area Network, Internet
Operating Systems	Windows 2000/XP/2003
Hardware	Workstation, PC, laptop

The performance testing is mainly based on two network environments mentioned above. Since the local area network (LAN) speed is 10Mbps and 100Mbps or more, *GeoLink* can easily meet the design requirement with excellent performance. When *GeoLink* is deployed over Internet, the network speed may change from time to time. The performance is a challenging issue. Because the limitation of the time and cost, the detailed performance evaluation will be available in the future work. Some testers' experiences show that when the network speed is above 200Kbps, *GeoLink* can work smoothly. When the network speed is below 56Kbps and the testers use ArcIMS Web Service as the data set, the testers will cost intolerant time to wait the incoming messages.

Chapter 6 Conclusion and Future Work

6.1 Significance and Contributions

With the development and grow-up of CSCW and SDSS, a great innovation for the GIS will go to collaborative GIS in which users can collaboratively operate the GIS applications with others through the Internet. Unfortunately, neither most current main frame GIS systems support collaborative functions, nor do theirs open APIs and the applications based on the APIs.

The outputs of the research could potentially have a significant impact on collaborative GIS especially on the applications in Spatial Decision Support Systems (SDSS), Spatial Decision Making Systems, etc. This research makes the following contributions to the overall field of knowledge in this area:

- The thesis analyzed and summarized the current solutions for system design and development in CSCW and related applications which will be an important reference to the design and development of collaborative GIS.
- The system requirements based on a case study is conducted through a case in Emergency Operating Centre. Through the case study, some core functions requirements in collaboration with GIS environments are obtained. These functions requirements are not only applied to EOC's but also many collaborative GIS applications.
- A hybrid (or semi-replicated) system architecture is designed according to the special requirements of collaborative GIS.
- A prototype for collaborative GIS is designed, developed, tested and deployed.

- The modules used in the prototype are extendable and scalable, which make it possible to change a single-user interface GIS application to a collaborative GIS application just through adding some collaborative component or plug-ins. Since GIS modules and Collaborative modules are highly independent, the collaborative modules can, therefore, be applied to other GIS applications with minor changes.

6.2 Limitation of the Research

Because *GeoLink* is a concept-proof prototype, the research reported in this thesis is subject to several possible constraints and limitations listed as follows:

- The system requirement analysis in this case study is an approximate one and not so well detailed that every piece of work is involved. The main requirements are focusing on GIS and collaborations. Others like workflows which are also very important in designing EOC system are ignored on purpose because that is out of my the research scope.

- Data source share component is not finished in the prototype although the logical design is discussed in the thesis. The users can just load and browse the maps from open ArcIMS Web Service, for example www. geographynetwork.com. Any special operation like editing map or adding new features is not allowed. The research on data source share component is underway.

- Based on the system design, any GIS tools should be available to develop the prototype, but just MapObject was tested for the development of GIS components. In the future, some open source GIS tools should be tested in proposed framework.

- The usability and functionality of prototype was not tested using rigorous software engineering testing methods such as alpha and beta testing.

6.3 Future Work

Some future work will be done next. The work includes:

- The problems in data source handling will be solved in collaborative GIS application. In this prototype, the clients can just load maps from open ArcIMS Web service without changing the map. A data server proxy will be further developed to handle two kinds of data sources, File source and Database source, with full access privileges.
- The feasibility and usability study of extending the system to include more required GIS functions such as :
  1) Some interfaces of the prototype will be changed to fit the multi-session, multi-task requirements.
  2) Annotation and mark-up of geographic (map) features with multimedia data in the form of text, graphics, photos, and audio/video clips will be added in the prototype.
  3) GIS analysis functions will be added in the prototype.
- As part of the future work, collaborative 3D and virtual reality environment are important aspects to be integrated in this prototype.

References

Armstrong, M. P., (1994) "Requirement for the development of GIS-based group decision support systems", journal of the American society for information science 45:667-677.

Begole, J., C.A. Struble and C.A. Shaffer, (1997) "Leveraging Java a pplets : toward collaboration transparency in Java". IEEE Internet Computing, 1 (1997), 2, p. 57-64, http://pdf.computer.org/ic/books/ic1997/pdf/w2057.pdf.

Begole, J. and Struble, C.A. and Shaffer, C. and Smith, R.B. (1999) "System Resource Sharing for Synchronous Collaboration". Technical Report TR-99-11, Computer Science, Virginia Tech. http://eprints.cs.vt.edu:8000/archive/00000524/01/BetEtAl99b.pdf

Begole, et. al. (1998) "Supporting Worker Independence in Collaboration Transparency". *Pages 133{142 of: Proceedings of the ACM Symposium on User Interface Software and Technology (UIST-98)*. New York: ACM Press.

Birman, K., Cooper, R., Joseph, T., Marzullo, K., Makpangou, M., Kane, K., Schmuck, F., and Wood, M. (1990) "The ISIS System Manual", Version 2.1. Department of Computer Science, Cornell University, 1990. Available at ftp://gogol.cenatls.cena.dgac.fr/pub/languages/Isis/ISISV21-DOC.TAR.Z.

Boyd, J. "Floor control policies in multi-user applications," (1993) in INTERACT '93 and CHI '93 conference companion on Human factors in computing systems. 1993. Amsterdam, The Netherlands:pp. 107 - 108.

Burridge, R. (1997) "Sharing Data in Highly Interactive Multimedia Conferencing Applications". Sun Microsystems, Inc., Mountain View, CA, September, 1997

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M. (1996) "Pattern-Oriented Software Architecture, A System of Patterns", John Wiley \& Sons.

Chabert et al., "NCSA Habanero-Synchronous collaborative framework and environment," http://havefun.ncsa.uiuc.edu/habanero/Whitepapers/ecscw-habanero.html (accessed on July, 2004)

Chabert, A., Grossman, E., Jackson, L., Pietrowicz, S., and Seguin, C. (1998) "Java Object-Sharing in Habanero". *Communications of the ACM*, Vol. 41 # 6, June 1998, pp 69-76. 13. JCE, 2004. Java Collaboration Environment website, http://snad.ncsl.nist.gov/madvtg/Java/overview.html (accessed on May 20, 2004)

Cameron, B., DePalma, D.A., O'Herron, R. and Smith, N. (1995) "Where does groupware fit? "*The Forrester Report: Software Strategies*, 6(3), June 1995.

Chang, Z. and Li, S. (2005) "A Framework design for collaborative GIS application: Based on Replicated Architecture", 98th Canadian Institute of Geomatics, Ottawa, June 13~15, 2005.

Chung, G., Jeffay, K., & Abdel-Wahab, H., (1994) "Dynamic participation in computer-based conferencing system". Journal of Computer Communications, 17(1), 7–16. (1994).

Churcher, N. & Churcher, C. (1999) "RealTime Conferencing in GIS", Transactions in GIS, 1999, 3(1): 23-30

Churcher, N., Churcher, C. (1996) "GroupARC – A Collaborative Approach to GIS", the 8th Colloquium of the spatial Information Research Center, University of Otago, New Zerland, July 9-11

Collabworx inc. Website, CollabWorx: a Web-based Real-Time Collaborative Platform http://www.collabworx.com/Technology/CWPlatform/architecture.html (accessed on July, 2004)

Coulouris, G., Dollimore, J., Kindberg, T., (1994) "*Distributed Systems: Concepts and Design*". Second edn. Reading, MA, USA: Addison-Wesley.

Deering, S. E., (1995) "Multicast routing in internetworks and extended lans". ACM SIGCOMM, 25(1), Jan 1995.

Dewan, P. (1999) "Architectures for Collaborative Applications". Computer Supported Cooperative Work. M. Beaudouin-Lafon. Chichester, UK, John Wiley & Sons: 169-194.

Dix, A., (1996) "Challenges and Perspectives for Cooperative Work on the Web", Proceedings of the ERCIM workshop on CSCW and the Web, Sankt Augustin, Germany, February 7-9, 1996

Ellis, C., Gibbs, S., and Rein, G (1991) "Groupware: Some issues and experiences" Communications of ACM, 34(1), 38-58

Faber, B. G, Wallace, W., Croteau, K., Thomas, V., & Small, L. (1997) "Active Response GIS: An Architecture for Interactive Resource Modeling", Proceedings of the GIS'97 Annual Symposium on Geographic Information Systems, Vancouver, B.C, March 1997, pp. 296-301

Gallo, S. K. (1996) "GIS Within Electronic Meetings: The Effective Use of Spatial Data in a Joint Intellectual Effort", Proceedings of 1996 ESRI User Conference, Palm Springs, California, May 20-24, 1996

Garfinkel, D., Wleti, B., & Yip, T. Hp sharedx (1994) "A tool for real-time collaboration". HP Journal, 45(2), 23–36. (1994).

Greenberg, S., Roseman, M., (1999) "Groupware Toolkits for Synchronous Work". *Chap. 6, pages 135{168 of:* Beaudouin-Lafon, Michel (ed), *Computer-Supported Cooperative Work, Trends in Software Series*. John Wiley & Sons.

Greenberg, S., Hayne, S., Rada, R. (1995) "Designing Groupware for Real-Time Drawing". McGraw Hill.

Greenberg, S. (1990) "Sharing views and interactions with single-user applications," in Proceedings of the ACM/IEEE Conference on Office Information Systems. 1990. Cambridge, MA: pp. 227-237.

Greenberg, S. (1991) "Personalizable groupware: Accommodating individual roles and group differences," in Proceedings of the ECSCW '91 European Conference of Computer Supported Cooperative Work. 1991. Amsterdam: Kluwer Academic Press. pp. 17-32.

Handley, M., Wakeman, I., and Crowcroft, J. (1995) "The conference control channel protocol (CCCP): a scalable base for building conference control applications," in Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication. 1995. Cambridge, MA: pp. 275- 287.

Hill, R.D., T. Brinck, S.L. Rohall, J.F. Patterson and W. Wilner, (1994) "the rendezvous architecture and language for constructing multiuser applications". *ACM Transactions on Computer-Human Interaction*, 1 (June 1994), 2, p. 81-125, http://www.acm.org/pubs/articles/journals/tochi/1994-1-2/p81-hill/p81-hill.pdf.

Hinssen, P.J.H., (1998) "What difference does it make? The use of groupware in small groups". Elematica Instituut Fundamental Research Series, vol. 002. Telematica Instituut, Enschede, the Netherlands, in press, http://www.telin.nl/publicaties/1998/scout/scout.htm.

Hofte, G. H., (1998) "Working Apart Together: Foundations for component groupware", Telematica Instituut Fundamental Research Series

Inkpen, K., et al. (1997) "Turn-Taking Protocols for Mouse-Driven Collaborative Environments," in Proceedings of Graphics Interface '97. 1997. Kelowna, BC, Canada: pp. 138-145.

Illmann, T., Thol, R., Weber, M., "Transparent Latecomer Support for Web-Based, Collaborative Learning Environments", http://citeseer.csail.mit.edu/509281.html (accessed on May 6, 2004)

JAMM, (2004) Flexible Java Applets Made Multiuser website, http://simon.cs.vt.edu/JAMM/ (accessed on May 20, 2004)

JCE, Java Collaboration Environment website, http://snad.ncsl.nist.gov/madvtg/Java/overview.html (accessed on May 20, 2004)

Johansen, R., (1998) "Current user approaches to groupware". In R. Johansen (ed.), *Groupware : Computersupport for business teams*. Free Press, New York, 1988, p. 12-44.

Jones, R. M., Copas, C., & Edmonds, E. A. (1997) "GIS support for distributed group-work in regional planning", International Journal of Geographical Information Systems, 1997, Vol. 11, No. 1, pp. 53 – 71

Knister, M. and Prakash, A. (1990) "DistEdit: A Distributed Toolkit for Supporting Multiple Group Editors". In Proceedings of the ACM 1990 Conference on Computer Supported Cooperative Work (CSCW '90), 343-355, Los Angeles, CA, October, 1990.

KPB, 2004, Kenai Peninsula Borough Emergency Operations Center Guide, http://www.borough.kenai.ak.us/emergency/, (accessed on July, 2004)

Lantz, K. (1986) "An Experiment in Integrated Multimedia Conferencing". *In: Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW '86).* Austin, Texas: ACM Press. Reprinted in I. Greif (editor), *Computer-Supported Cooperative Work:A Book of Readings,* pp. 533{552, Morgan Kaufmann, 1988.

Lauwers, J. and Lantz, K., (1990) "Collaboration Awareness in Support of Collaboration Transparency: Requirements for the Next Generation of Shared-Window Systems," Proc. Conf. Human Factors in Computing Systems, ACM Press, New York, 1990, pp. 303-311.

Li, S. & Coleman, D. J. (2005) Modeling distributed GIS data production workflow. Computers, Environment and Urban Systems, 29(4), pp. 401-424

Li, S. and Z. Chang (2005) "Developing Component-based Generic Tools for Group Visualizing, Manipulating and Exploring Spatial Information", Proceedings of 2005 GeoTec Event, Vancouver, B.C., February 13-16, 2005

Li, W., Wang, W., Marsic, I., (1999). "Collaboration Transparency in the DISCIPLE Framework", Proceedings of the ACM International Conference on Supporting Group Work (GROUP'99), November 14-17, 1999, Phoenix, AZ.

MacEachren, A. M., Brewer, I., & Steiner, E. (2001) "GEOVISUALIZATION TO MEDIATE COLLABORATIVE WORK: Tools to Support Different-Place Knowledge Construction and Decision-making, Conference Proceedings of the 20th International Cartographic Conference, Beijing, China, August 6-10

Myers et. al., "Floor Control in a Highly Collaborative Co-located Task" http://www-2.cs.cmu.edu/~pebbles/papers/pebblesfloorcontrol.pdf (accessed on July, 2004)

Netmeeting, Microsoft Technet (2000), NetMeeting 2.1 OverView http://www.microsoft.com/technet/prodtechnol/netmting/reskit/netmtg2/chpt1.mspx (accessed on July, 2004)

Nyerges, T. L., Montejano, R., Oshiro, C., & Dadswell, M. (1997) "Group-based Geographic Information Systems for Transportation Improvement Site Selection", Transportation Records C: Engineering Technologies, Vol. 5, No. 6, pp. 349 – 369

Ousterhout, J. (1994) "*Tcl and the Tk Toolkit*". Addison Wesley.

Roth, J. and C. Unger (2000). "An extensible classification model for distribution architectures of synchronous groupware". Fourth International Conference on the Design of Cooperative Systems (COOP2000), Sophia Antipolis (France), May 2000, IOS Press.

Roseman, M. and Greenberg, S. (1992) "Groupkit: A Groupware Toolkit for Building Real-Time Conferencing Applications". In *Proceedings of the ACM 1992 Conference on Computer Supported Cooperative Work (CSCW '92)*, 43-50, Toronto, Canada, November 1992

Shaw, M. and Garlan, D.,(1996) "Software Architecture: Perspectives on an Emerging Discipline". Prentice Hall, New Jersey, 1996.

Smith,R., B., (1996) "Kansas: A Large, Flat, Multiuser Virtual World for Interactive Simulations," Virginia Tech Computer Science Colloquium Series, Apr. 1996.

Sun Microsystems, inc. Website, Java tutorial, http://java.sun.com/docs/books/tutorial/uiswing/components/rootpane.html, (accessed on July, 2005)

Suthers, D., (2001). "Architectures for Computer Supported Collaborative Learning", IEEE International Conference on Advanced Learning Technologies (ICALT 2001).

Wang, W., Dorohonceanu, B. and Marsic, I., (1999). "Design of the DISCIPLE Synchronous Collaboration Framework", Proceedings of the IASTED International Conference Internet and Multimedia Systems and Applications October 18-21, 1999 - Nassau, Grand Bahamas.