

**A WEB-BASED APPROACH TO THE INTEGRATION  
OF DIVERSE DATA SOURCES FOR GIS**

**Geoffrey Yu Kai Shea**

**Master of Engineering  
The University of New South Wales  
2001**

## ABSTRACT

The rigorous developments of GIS over the past decades have enabled application developers to create powerful systems that are used to facilitate the management of spatial data. Unfortunately, each one of these systems is specific to a local service, with little or no interconnection with services in other locales. This makes it virtually impossible to perform dynamic and interactive GIS operations across multiple locales which have similar or dissimilar system configurations. The Spatial Data Transfer Standard (SDTS) resolved the problems partially by offering excellent conceptual and logical abstraction model for data exchange. Recent advancements of the Internet enlightened the GIS community as to the realization of an ideal concept of information interchange. A suite of new technologies that embraces Extensible Markup Language (XML), Scalable Vector Graphics (SVG), Portable Network Graphics (PNG) and Java creates a powerful and new perspective that can be applied to all phases of online GIS system development. The online GIS is a Web-based approach to integrating diverse spatial data sources for GIS applications.

To address the spatial data integration options and implications related to the Web-based approach the investigation was undertaken in 5 phases: (1) Determine the mapping requirements of graphic and non-graphic spatial data for online GIS

application; (2) Analyze the requirements of spatial data integration for online environments; (3) Investigate a suitable method for integrating different formats of spatial data; (4) Study the feasibility and applicability of setting up the online GIS; and (5) Develop a prototype for online sharing of teaching resources.

Resulting from the critical review on current Internet technology, a conceptual framework for spatial data integration was proposed. This framework was based on the emerging Internet technology on XML, SVG, PNG, and Java. It was comprised of four loosely coupled modules, namely, Application Interface, Presentation, Integrator, and Data module. This loosely coupled framework provides an environment that will be independent of the underlying GIS data structure and makes it easy to change or update the system as a new task or knowledge is acquired.

A feasibility study was conducted to test the applicability for the proposed conceptual framework. A detailed user requirements and system specification was thus devised from the feasibility study. These user requirements and system specification provided some guidelines for online GIS application development. They were expressed specifically in terms of six aspects: (1) User; (2) Teaching resources management; (3) Data; (4) Cartography; (5) Functions; and (6) Software development configuration.

A prototype system based on some of the devised system specifications was developed. In the prototype software design, the architecture of a Three-Tier Client-Server computing model was adopted. Due to the inadequacy of native support for

SVG and PNG in all currently available Web browsers, the prototype was thus implemented in HTML, Java and vendor specific vector format. The prototype demonstrated how teaching resources from a variety of sources and format (including map data and non-map resources) were integrated and shared. The implementation of the prototype revealed that the Web is still an ideal medium for providing wider accessibility of geographical information to a larger number of users through a corporate intranet or the Internet cost-effectively.

The investigation concluded that current WWW technology is limited in its capability for spatial data integration and delivering online functionality. However, developing of XML-based GIS data model and graphic standards – SVG and PNG – for structuring and transferring spatial data on the Internet appear to be providing solutions to the current limitations. It is believed that the ideal world where everyone retrieving spatial information contextually through a Web browser disregarding the information format and location will eventually become true.

# TABLE OF CONTENTS

	<u>Page</u>
Abstract .....	ii
Table of Contents .....	v
List of Figures .....	ix
List of Tables .....	xi
List of Abbreviations and Acronyms .....	xii
Acknowledgments .....	xiv
<b>1 INTRODUCTION .....</b>	<b>1</b>
1.1 Background .....	1
1.2 The Data Integration Problem .....	2
1.3 Objectives of the Research .....	4
1.4 Overview of the Thesis .....	5
<b>2 TECHNIQUES FOR DATA INTEGRATION .....</b>	<b>7</b>
2.1 Overview of Map Data Sharing among Traditional GIS .....	8
2.1.1 The Integration of Spatial and Non-Spatial Data .....	8
2.1.2 The Integration of Raster and Vector Data .....	9
2.1.3 Multimedia GIS .....	10
2.1.4 Hypermedia GIS .....	12
2.2 Limitations of Current Approaches to Data Integration in GIS .....	13
2.2.1 Proprietary Data Formats .....	13
2.2.2 Lack of Data Information .....	16
2.2.3 Data Transmission Restriction .....	17
2.3 Expanding the Scope of Data Integration .....	18
2.4 A Proposed Framework for Data Integration .....	18
2.4.1 Application Interface Module .....	20
2.4.2 Presentation Module .....	20
2.4.3 Integrator Module .....	20
2.4.4 Data Module .....	21
2.5 The Use of Standards for Data Integration .....	22
2.6 Internet as an Integrating Technology .....	23
2.7 Summary .....	25
<b>3 INTERNET AND WEB PROTOCOLS .....</b>	<b>26</b>
3.1 The Concept of the Internet .....	27
3.2 World Wide Web .....	28
3.2.1 How the Web Works: Concept .....	29
3.2.2 How the Web Works: HTTP .....	30
3.2.3 How the Web Works: HTML .....	31
3.2.4 How the Web Works: URL .....	33
3.2.4.1 The protocol .....	34
3.2.4.2 Domain Name .....	35
3.2.4.3 Path and Filename .....	35

3.3 The Web and Intranets . . . . .	36
3.4 Client-Server Architecture . . . . .	36
3.4.1 The Two-Tier Model . . . . .	39
3.4.2 The Three-Tier Model . . . . .	40
3.5 Summary . . . . .	42
<b>4 ONLINE GIS . . . . .</b>	<b>43</b>
4.1 Online GIS Architecture . . . . .	44
4.1.1 Client-Side Online GIS . . . . .	44
4.1.2 Server-Side Online GIS . . . . .	47
4.1.2.1 Server-Side Programming Technology - Java . . . . .	50
4.1.2.2 Server-Side Scripting Technology - Active Server Pages . . . . .	52
4.1.3 Balance Between Client-Side and Server-Side Solutions . . . . .	55
4.2 Online GIS Data Model and Format . . . . .	56
4.2.1 User-definable Data Modeling on the Web . . . . .	58
4.2.1.1 An Overview of XML . . . . .	59
4.2.1.2 XML-based Applications . . . . .	61
4.2.1.3 Potential Benefits of XML-based Online GIS Applications . . . . .	61
4.2.1.4 Modeling Online GIS Applications with XML . . . . .	63
4.2.1.4.1 Analyze the information requirements . . . . .	67
4.2.1.4.2 Analyze the application requirements . . . . .	69
4.2.1.4.3 Analyze the presentation requirements . . . . .	70
4.2.2 Vector Graphics for the Web: SVG . . . . .	71
4.2.2.1 More Than Just a Vector Graphics Engine . . . . .	74
4.2.2.2 Describing Vector Graphics with SVG for Online GIS . . . . .	74
4.2.3 Raster Graphics for the Web: PNG . . . . .	77
4.2.3.1 Choosing the Right Raster Format for Web Applications . . . . .	77
4.2.3.1.1 Bit depth: grayscale vs. truecolor vs. palette . . . . .	78
4.2.3.1.2 Compression Algorithm: lossy vs. lossless . . . . .	79
4.2.3.1.3 Image format: GIF vs. JPEG vs. PNG . . . . .	80
4.2.3.2 The PNG Features . . . . .	81
4.2.3.3 Describing Raster Graphics with PNG for Online GIS . . . . .	83
4.3 Interactive Problem of Online GIS . . . . .	84
4.3.1 The Current Web-Enabled GIS Packages . . . . .	86
4.3.2 The Future Trend of Online GIS: XML; SVG; and PNG . . . . .	87
4.4 Summary . . . . .	89
<b>5 FEASIBILITY STUDY FOR SETTING UP ONLINE TEACHING RESOURCES OVER LSGI'S INTRANET . . . . .</b>	<b>90</b>
5.1 Scope . . . . .	91
5.2 Analysis . . . . .	92
5.2.1 Types of Educational Resources for Sharing . . . . .	92
5.2.2 Present Situation and Existing Problems . . . . .	93
5.2.2.1 Adaptation to the Current Computing Facilities in LSGI . . . . .	93
5.2.2.2 File Sharing . . . . .	94
5.2.2.2.1 File sharing problem using FTP server . . . . .	95
5.2.2.2.2 File sharing problem using file system of LAN server . . . . .	96
5.2.2.2.3 File sharing problem using local file system with network services . . . . .	97
5.2.2.3 GIS Packages . . . . .	98
5.2.2.4 Map Data Sharing . . . . .	99
5.3 System Requirements . . . . .	100
5.3.1 User Requirement . . . . .	101
5.3.2 Teaching Resources Management Requirement . . . . .	101

5.3.3 Data Requirement . . . . .	103
5.3.3.1 Map Data . . . . .	103
5.3.3.2 Non-map Data . . . . .	103
5.3.4 Cartographic Requirement . . . . .	104
5.3.5 Functional Requirement . . . . .	105
5.3.6 Software Development Configuration . . . . .	106
5.4 Summary of Feasibility Study for Setting up Online Teaching Resources over LSGI's Intranet . . . . .	108
5.5 Prototype Design . . . . .	109
5.5.1 Overall Design . . . . .	109
5.5.2 Interface Design . . . . .	112
5.5.2.1 Interface Design for TRSC . . . . .	115
5.5.2.2 Interface Design for MDRC . . . . .	116
5.5.3 Exploring Design Choices . . . . .	116
<b>6 RESULTS . . . . .</b>	<b>120</b>
6.1 Implementation of Prototype . . . . .	120
6.1.1 Implementation of MDRC . . . . .	121
6.1.2 Implementation of TRSC . . . . .	122
6.2 Discussion of Results . . . . .	123
6.2.1 Results Achieved . . . . .	123
6.2.1.1 Results Achieved in Developing MDRC . . . . .	123
6.2.1.1.1 Integration of different sources of map data . . . . .	124
6.2.1.1.2 Implementation of security on map data . . . . .	126
6.2.1.1.3 Selective(thematic) display of map data . . . . .	126
6.2.1.1.4 Multiple display symbology with the same map data set . . . . .	127
6.2.1.1.5 Basic GIS queries on map data . . . . .	128
6.2.1.1.6 Basic GIS analysis function . . . . .	130
6.2.1.2 Results Achieved in Developing TRSC . . . . .	132
6.2.2 Difficulties Encountered . . . . .	135
6.2.2.1 Failure Implementation of Overlay Analysis . . . . .	135
6.2.2.2 Successful Implementation of Modified FTP Protocol . . . . .	137
6.3 Anticipated Future Enhancement . . . . .	138
6.3.1 MDRC Enhancement . . . . .	138
6.3.2 TRSC Enhancement . . . . .	140
6.3.3 Integrated Information Sharing System . . . . .	142
<b>7 CONCLUSIONS AND RECOMMENDATIONS . . . . .</b>	<b>143</b>
7.1 Conclusions . . . . .	143
7.1.1 Objectives and Strategy . . . . .	143
7.1.2 Mapping Requirements of Graphic and Non-graphic Spatial Data for Online GIS Application . . . . .	144
7.1.3 Requirements of Spatial Data Integration for Online Environments . . . . .	144
7.1.4 Suitable Method for Integrating Different Formats of Spatial Data . . . . .	145
7.1.5 Feasibility and Applicability of Setting Up the Online GIS . . . . .	146
7.1.6 Prototype for Online Sharing of Teaching Resources . . . . .	147
7.2 Recommendations . . . . .	148
<b>REFERENCES . . . . .</b>	<b>150</b>

<b>APPENDICES</b> .....	<b>158</b>
A Functional Requirement Specifications for Teaching Resources Sharing Component .....	158
B Mechanisms to Facilitate the Teaching Resources Sharing Component Functions .....	178
C Database Design and Schemas for Teaching Resources Sharing Component .....	186
D Index File Definitions for Teaching Resources Sharing Component .....	193
E Walkthrough and Screen Shots of the Map Data Retrieval Component .....	200
F Walkthrough and Screen Shots of the SVG Web Mapping Demonstration .....	212
G Technical Description of the SVG Web Mapping Methodology .....	219



## LIST OF FIGURES

	<u>Page</u>
Figure 2.1	Proposed framework for data integration ..... 19
Figure 2.2	Detail arrangement in Data Module ..... 22
Figure 2.3	The use of standards for proposed framework ..... 24
Figure 3.1	A simple HTML document ..... 33
Figure 3.2	Corresponding source file ..... 33
Figure 3.3	An example of a URL ..... 34
Figure 3.4	Basic Client-Server architecture ..... 37
Figure 3.5	A three-tier application ..... 41
Figure 4.1	An example of client-side application ..... 45
Figure 4.2	An example of server-side application ..... 47
Figure 4.3	Architecture of ArcView IMS (adopted from ESRI) ..... 49
Figure 4.4	XML makes data sharing from any device ..... 66
Figure 4.5	A typical three-tier client-server XML-based application ..... 67
Figure 5.1	Overall prototype system architecture ..... 110
Figure 5.2	Overall architecture of Map Data Retrieval Component ..... 113
Figure 5.3	Workflow in the Map Data Retrieval Component ..... 114
Figure 6.1	Main page of MDRC ..... 124
Figure 6.2	Simultaneous display of raster and vector map ..... 125
Figure 6.3	A closer look on the raster and vector map ..... 125
Figure 6.4	Logon window for username/password protected layer ..... 126
Figure 6.5	Using legend to display map layers selectively ..... 127
Figure 6.6	Line style used for Rail layer at 1:20000 ..... 127
Figure 6.7	Line style used for Rail layer at 1:10000 ..... 128
Figure 6.8	A measuring distance example ..... 129
Figure 6.9	Map objects selection by polygon ..... 129
Figure 6.10	Report on the selected map objects ..... 130
Figure 6.11	Zoom buffering properties ..... 131
Figure 6.12	Result of zone buffering ..... 131
Figure 6.13	Prototype graphical user interface of Client Module ..... 133
Figure A-1	System architecture overview ..... 166
Figure A-2	Database Server Interface Module & Database Server System architecture ..... 168
Figure A-3	File System Interface Module & File Server architecture ..... 168
Figure A-4	Proxy Server Module architecture ..... 169
Figure A-5	Client Module architecture ..... 170
Figure A-6	Prototype graphical user interface of Client Module ..... 172
Figure A-7	Interface design for the Login Interface ..... 172
Figure A-8	Interface design for the File Sharing Interface ..... 174
Figure A-8a	Interface design for the File Sharing Interface (Uploading) ..... 174

Figure A-8b	Interface design for the File Sharing Interface (Downloading) . . . . .	175
Figure A-9	Interface design for the Administration Interface . . . . .	176
Figure B-1	New file transfer commands for TRSC (based on FTP RFC959) . . . . .	184
Figure C-1	The TRSC system database . . . . .	191
Figure C-2	TRSC database related to user information . . . . .	191
Figure C-3	TRSC database related to file information . . . . .	192
Figure D-1	Profile for Login . . . . .	195
Figure D-2	Index File for Updating . . . . .	195
Figure D-3	Index File for Searching (Request) . . . . .	195
Figure D-4	Index File for Searching (Reply) . . . . .	195
Figure E-1	Information page of MDRC . . . . .	201
Figure E-2	Main page of MDRC . . . . .	202
Figure E-3	Structure of MDRC main page . . . . .	202
Figure E-4	The legend . . . . .	203
Figure E-5	Using legend to display map layers selectively . . . . .	203
Figure E-6	A typical example of map window popup menu . . . . .	204
Figure E-7	Sub-menu page 1 . . . . .	204
Figure E-8	Sub-menu page 2 . . . . .	204
Figure E-9	List of zoom functions . . . . .	205
Figure E-10	A measuring distance example . . . . .	206
Figure E-11	Map objects selection by polygon . . . . .	207
Figure E-12	Report on the selected map objects . . . . .	207
Figure E-13	Zoom buffering properties . . . . .	208
Figure E-14	Result of zone buffering . . . . .	208
Figure E-15	Hyperlinked map objects example . . . . .	209
Figure E-16	Logon window for username/password protected layer . . . . .	209
Figure E-17	Line style used for Rail layer at 1:20000 . . . . .	210
Figure E-18	Line style used for Rail layer at 1:10000 . . . . .	210
Figure E-19	Simultaneous display of raster and vector map . . . . .	211
Figure E-20	A closer look on the raster and vector map . . . . .	211
Figure F-1	Information page of SVG Web Mapping . . . . .	213
Figure F-2	Main page of SVG web mapping demonstration . . . . .	214
Figure F-3	Structure of SVG web mapping demonstration . . . . .	214
Figure F-4	The legend . . . . .	215
Figure F-5	Using legend to display map layers selectively . . . . .	215
Figure F-6	A typical example of map window popup menu . . . . .	216
Figure F-7	A fly-out information about the map object . . . . .	217
Figure F-8	A hyperlinked object . . . . .	217
Figure F-9	Text annotation on a curved path, layer opacity, and animation . . . . .	218
Figure G-1	Main page of technical description of SVG web mapping . . . . .	220

## LIST OF TABLES

	<u>Page</u>
Table 2.1	Proposed specifications used in data integration ..... 24
Table 5.1	H/W and S/W configurations of Workstation GIS and Desktop GIS ..... 98
Table 5.2	Software development configuration ..... 107
Table 5.3	Comparison among different Web-enabled GIS packages ..... 116
Table D-1a	Profile for login - fields definition ..... 196
Table D-1b	Profile for login - tuple definition ..... 197
Table D-2a	Definition of the index file for uploading - fields definition ..... 197
Table D-2b	Definition of the index file for uploading - tuple definition ..... 198
Table D-3a	Definition of the index file for searching (request) - fields definition ..... 198
Table D-3b	Definition of the index file for searching (request) - tuple definition ..... 199
Table D-4a	Definition of the index file for searching (reply) - fields definition ..... 199
Table D-4b	Definition of the index file for searching (reply) - tuple definition ..... 199

## LIST OF ABBREVIATIONS AND ACRONYMS

API	Application Programming Interface
ARPANET	Advanced Research Projects Agency Network
ASCII	American Standard Code for Information Interchange
ASP	Active Server Pages (Microsoft)
CGI	Common Gateway Interface
CML	Chemical Markup Language
CORBA	Common Object Requests Broker Architecture
CRC-32	32-bit Cyclic Redundancy Check
CSIRO	Commonwealth Scientific Industrial Research Organisation, Australia
CSS	Cascading Style Sheets
DGN	<i>MicroStation</i> native drawing file format (Bentley)
DOM	Document Object Model
DTD	Document Type Definition
DWG	<i>AutoCAD</i> native drawing file format (Autodesk)
ESRI	Environmental Systems Research Institute, Inc.
FTP	File Transfer Protocol
GIF	Graphics Interchange Format
GIS	Geographic Information System
GUI	Graphical User Interface
HREF	hyper-text reference
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
H/W	hardware
IIS	Internet Information Server (Microsoft)
IMROC	Inner Metropolitan Regional Organisation of Councils, Australia
ISAPI	Internet Server Application Programming Interface
JPEG	Joint Photographic Experts Group
LAN	Local Area Network
LIC	Land Information Centre, Lands Dept., Hong Kong Special Administrative Region
LSGI	Dept. of Land Surveying & Geo-Informatics, The Hong Kong Polytechnic University
MathML	Mathematical Markup Language
MDAC	Microsoft Data Access Component
MDRC	Map Data Retrieval Component
MG	Map Guide (Autodesk)

MS	Microsoft
MWF	map window file (Autodesk)
NSAPI	Netscape Server Application Programming Interface
OCR	Optical Character Recognition
ODBC	Open Database Connectivity (Microsoft)
OGC	Open GIS Consortium
OLE	Object Linking and Embedding
PDA	Personal Data Assistant
PERL	Practical Extraction and Report Language
PGML	Precision Graphics Markup Language
PNG	Portable Network Graphics
RDBMS	Relational Database Management System
RFC	Request For Comments
RMI	Remote Method Invocation
SDE	Spatial Database Engine (ESRI)
SDF	spatial data file (Autodesk)
SDTS	Spatial Data Transfer Standard
SGML	Standard Generalized Markup Language
SMTP	Small Mail Transfer Protocol
SQL	Structured Query Language
SVG	Scalable Vector Graphics
S/W	software
TCP/IP	Transmission Control Protocol/Internet Protocol
TIFF	Tagged Image File Format
TPU	Tertiary Planning Unit (Planning Dept. of Hong Kong Special Administrative Region)
TRSC	Teaching Resources Sharing Component
URL	Uniform Resource Locator
VM	virtual machine
VML	Vector Markup Language
W3C	World Wide Web Consortium
WMT	Web Mapping Testbed
WWW	World Wide Web
XML	eXtensible Markup Language

## ACKNOWLEDGMENTS

Acknowledgment is due to The Hong Kong Polytechnic University for the financial assistance of this study (Staff Development Programme Reference No. GS368/13/1<sup>IV</sup>). The work described in Chapter 5 of this thesis was substantially supported by a grant from the Learning and Teaching Committee of The Hong Kong Polytechnic University (Project No. DLTDC/LSGI01).

Thanks also go to the Lands Department of The Government of the Hong Kong Special Administrative Region for providing the 1:1000 and 1:20000 of Hong Kong digital map data for this project.

I would like to express my sincere thanks to: Dr. Ewan G. Masters, my supervisor, for his invaluable guidance and constructive comments during this project; and Dr. A.H.W. Kearsley, Head of School, for solving my academic enrolment problems.

My colleague, Dr. Lilian S.C. Pun-Cheng, helped me in the final checking of this thesis. I appreciate her for her timely assistance.

Finally, but not the least, I would like to thank my wife Tsay Tsae-jiuan for her endless support throughout my studies. Her patience with me during this time allowed me to concentrate on my job and on finishing the project.

# CHAPTER 1

## INTRODUCTION

### 1.1 Background

A Geographic Information Systems (GIS) is a system for capturing, storing, checking, manipulating, analyzing and displaying data which are spatially referenced to the Earth [DoE, 1987]. A GIS can capture and handle data from various sources, at different scales and in a variety of structures. Implementing a GIS requires a chain of supports ranging from hardware to software and communication tools. It is so designed for managing spatially related resources that a great diversity of applications could be applied. Some of the typical GIS applications are real estate management systems which are used to maintain real estate inventories; and emergency dispatch systems which are used by hospital officials as a supplementary tool to monitor the dispatch of emergency vehicles and plan the shortest route to the scene of emergency.

As a result of the increasing number of GIS applications, there is a growing demand for exchanging spatial data across different platforms, with different formats situated in different locations. In fact, some argue that sharing data will be a key



influence in the continued growth of GIS technology [Antenucci, 1991]. The traditional way of accessing geographic data, however, requires the use of proprietary and expensive GIS packages. This requirement directly limits the use and distribution of geographic data and incurring the problem of data compatibility. It is widely recognized that the full potential of GIS can only be realized when a range of integrated data sets are available for general use in spatial analytical tasks, and thus developments in data integration are key factors towards the wide acceptance of GIS in the information era.

## **1.2 The Data Integration Problem**

In current practice, spatial data from diverse sources have to be made comparable, compatible, and consistent before GIS operations can be performed. Moreover, system connectivity has to be addressed in order to provide an effective and efficient communication means for data interchange. That means data integration problems are always confronting the users when beginning to use a GIS.

Basically, there are three technical problems related to data integration [Flowerdew, 1991]: data modeling; data currency; and data quality problems. In relation to these problems for the exchange of spatial data, the primary difficulty is the lack of a neutral format or syntax for suitably describing the data. In order to implement a successful data exchange mechanism, therefore, the selection of an

appropriate data model is the first hurdle to be overcome in the processes of spatial data integration.

Ideally, users of spatial data should concentrate on their methods to solve particular spatial analytical tasks with instant access to as many different data sets as required without concern for data formats and locations. The users can judge for themselves with reference to the meaning behind the data when it is most appropriate to use each data set. The data manipulation process to convert from one format to another should be handled by the system transparently.

Beside the problems in technical realm, according to the views of some researchers [Epstein, 1991; Carter, 1992; Rhind, 1992], there are financial and legal issues such as charging model for data and right-to-use of data should be handled properly, otherwise the hope for greater sharing of resources – spatial data – will turn to the other end.

With the rapid advancement in computer hardware and the increasing popularity of the Internet technologies, application developers have more flexibility in optimizing functionality, performance and network traffic by distributing the application load between the client and the server. In essence it is reasonable to say that the ability to integrate data is no longer limited by the information technology, but rather by the lack of understanding of the data and the ability to conceptualize problems. The Internet can definitely serve as a catalyst to the development of GIS by

providing an interactive, convenient infrastructure for delivering geographic data and GIS operations in a single unified system.

### **1.3 Objectives of the Research**

The aim of this research was to investigate the mapping implications of setting up an online GIS application for an academic department in the Internet/Intranet environment. The intention of this project is to present a Web-based approach for data integration that will be built upon a framework of ideas on the use of the emerging Internet technologies. The economic and legal implications are beyond the scope of this thesis. This investigation into an online GIS is justified by the increasing popularity of the Internet technology for developing Web applications. A set of emerging technologies, which includes eXtensible Markup Language (XML), Scalable Vector Graphics (SVG), Portable Network Graphics (PNG) and Java technology, for the Internet will be studied thoroughly in this project to examine their capabilities in the development of online GIS applications. The main objectives of this study can be summarized as follows:

- A. Determine the mapping requirements of graphic and non-graphic spatial data for online GIS application.
- B. Analyze the requirements of spatial data integration for online environments.

- C. Investigate a suitable method for integrating different formats of spatial data.
- D. Study the feasibility and applicability of setting up the online GIS.
- E. Develop a prototype for online sharing of teaching resources over an academic departmental (Department of Land Surveying & Geoinformatics, The Hong Kong Polytechnic University) intranet.

## **1.4 Overview of the Thesis**

The findings of this research have been organized into the following chapters. To start with is the introductory and background studies on the problem domain of data integration.

Chapter 3 provides an overview of the Internet and the Web. A close look at the Hyper-Text Transfer Protocol (HTTP) are taken. The similarity between Internet and Intranet is then briefly explained. Two commonly used Internet architectures, Two-Tier and Three-Tier Client-Server models, are reviewed in this chapter to lay the foundation for discussion on online GIS in subsequent chapters.

The principles of the underlying technologies for building Web-based GIS applications are the focus of Chapter 4. This chapter starts the discussion on the two fundamental approaches to online GIS applications, client-side online GIS and

server-side online GIS. A summary of the problems encountered and benefits obtained from the online GIS is given. There is a thorough investigation into the three emerging technologies for developing Web applications, eXtensible Markup Language (XML), Scalable Vector Graphics (SVG) and Portable Network Graphics (PNG). This chapter concludes with improvements to the capabilities and efficiency of Web-based GIS applications. The improvements mainly obtained by employing XML metalanguage for representing structured information along with the SVG and PNG for transferring graphics on the Web.

Chapter 5 is a case study for the prototyping of an Internet/Intranet GIS in the Department of Land Surveying & Geo-Informatics of the Hong Kong Polytechnic University. Several factors are taken into consideration for setting up the prototype system: how to handle different graphics formats, how to incorporate information from different locations, how and to what extent the GIS operations can provide interactive results to client, and how to present the information to client effectively. Other technical issues such as performance and network security are also discussed.

The problems encountered in the implementation and discussion of the prototyping results are provided in Chapter 6.

Finally, the conclusions of this project and recommendations for further research are presented in Chapter 7.

## **CHAPTER 2**

### **TECHNIQUES FOR DATA INTEGRATION**

The development of GIS technology can be depicted as a three-stage process that is in line with other information technologies. GIS technology in its first stage of development is oriented towards data modeling and collection. In the second stage, emphasis shifts to GIS software development/enhancement aimed at providing users with large arrays of functionality and sophisticated analytical operation utilities. With several decades of advances in GIS technology, many applications are now operational and highly productive. As such, a huge collection of diverse thematic spatial data in digital form are becoming generally available. The GIS technology is entering its third stage of development – technology flourishing – when it is truly and widely employed as a spatial decision support system.

It is clearly indicated that the success or failure of GIS technology is largely dependent on the availability of spatial data and on the ability to integrate diverse data sets across multiple systems. Moreover, due to the high cost of data collection and maintenance, it is a financial and reasonable move for a data originator to share/integrate data within an organization and across institutional boundaries. Thus,

the issue is no longer whether or not to use multiple data sets, but how to use it in an efficient and cost-effective manner to successfully realize the data sharing benefits.

This chapter will discuss the prevailing conditions and practices of spatial data integration in an attempt to address new challenges in the GIS technology.

## **2.1 Overview of Map Data Sharing among Traditional GIS**

The proliferation of information technology and GIS has provided an obvious alternative to integrate diverse data for spatial decision making. Thus, it is beneficial to examine and understand, at the outset, the development of the different approaches for a multitude of data integration. The following four major approaches can be summarized from a survey on the GIS literature.

### **2.1.1 The Integration of Spatial and Non-Spatial Data**

One of the long-standing characteristics of GIS since its inception is the capability to integrate graphic and attribute data. With the advancement in GIS technology over the past decades, much of the debate in recent years regarding the relative merits of raster model versus topological vector model has now been nearly settled. The consensus has reached to a point where vector data provide excellent cartographic quality presentation, while raster data provide more advanced spatial

analysis facilities. Regardless of what the internal storage format for spatial data is used, the non-graphic attributes data that are associated with the geographic data are seamlessly integrated within a specific single database called “geo-referenced database.” (See, for example, [Abel, 1989]; [Bracken, 1989])

### **2.1.2 The Integration of Raster and Vector Data**

Not long after the debate about relative merits of raster/vector started, major efforts have been made to integrate satellite imagery (remote sensing data) with topological vector data (see, for example, [Goodenough, 1988], [Zhou, 1989], [Lunetta *et al.*, 1991]). These efforts try to bring out the message that raster and vector models are complementary and that diverse data can be integrated to support spatial decision making. As noted in DoE [1987, p.2] “The benefits of a geographic information system depend on linking different data sets together.”

The pilot study conducted by Morrison and McDonald [1991] provides an example of the way in which both scanned images and vector data can co-exist to help solve the interim requirements for the transition from a traditional manual charting system to a fully computerized Land Information System. The pilot study has investigated several aspects such as extracting vectorized linework from the scanned images, overlaying scanned images with vector data, linking scanned images to textual databases, and browsing continuously the cascading scanned images.



To date GIS technology has reached a level of maturity which has enabled the development of a fully integrated GIS system capable of incorporating raster and vector data. Arc/Info 7 and later versions is a prime example of this category that allows vector information to be overlaid on raster images and provides utilities to convert between the vector and raster data models.

### **2.1.3 Multimedia GIS**

As with all information systems, a primary function of a GIS is efficient and rapid access to multi-facet spatial data. Dangermond [1989, p.1] remarks “A GIS brings information together, it unifies and integrates that information. It makes available information to which no one had access before, and places old information in a new context. It often brings together information which either was not or could not be brought together previously.” By moving beyond a mere display of vector and/or raster data alone, there is a growing concern for multimedia GIS system that can provide a common interface to a wide range of media types. The multimedia GIS appears to offer the best possibility to allow GIS to access all types of data – text, vector graphics, raster images, animation, video, audio, etc. – in a seamless way. McKeown and Lai [1987] suggest that internal multiple data representations should be adopted by all spatial information systems in which a piece of front-end interface is responsible for coordinating the display of these internal representations with appropriate software.

One such application was developed by Abel *et al.* [1991], who describe an object-oriented model<sup>1</sup> of a spatial information system that accomplishes the integration of disparate data sets known as “Environmental Decision Support System (EDSS)” prototype. This system consists of three basic objects. The conceptual structure of each data type and its associative behaviors are encapsulated in a “collection” object. The presentation of collection object is handled by “views” object. All the allowable GIS operations are abstracted under “operations” object. The complex manipulation on and displaying of diverse data sets in a highly-consistent way can be accomplished through the interactions between objects. The extendibility characteristic of object-oriented approach allows new form of data or operations to be added to the system with ease.

Lang [1992] describes a fundamental work that incorporated audio into GIS application to help simulate reality. The system enables users to experience noise levels at a particular location around the Chanute Air Force Base, Rantoul, Illinois by playing a calibrated digital sampling of jet takeoffs.

---

<sup>1</sup> The detail description of object-oriented technology is beyond the scope of this project. Worboys *et al.* [1990] give a concise discussion on the key concepts in object-oriented modeling and demonstrate how these concepts can be applied to the design of GIS.

### 2.1.4 Hypermedia GIS

The terms “hypertext” and “hypermedia” were first used by Ted Nelson in the 1960s as he was writing about his proposed system called *Xanadu*<sup>2</sup>. Originally hypertext was simply applied to text-only applications, whereas hypermedia was used to convey the inclusion of other media, especially audio and video. Actually the distinction between hypertext and hypermedia is so blurry now that they are used interchangeably. In a broader sense hypermedia systems can provide user-friendly interfaces with the capability to make cross-references within and across nodes of information – also called cards, documents, articles, files, pages, frames, screens – easy to create and traverse.

Based upon the hypertext philosophy, it is logical for hypermedia systems to further extend to spatial data. One such fundamental work in this area was demonstrated in the *Community Disk* of the *BBC Domesday* interactive video system [Openshaw and Mounsey, 1987].

The use of hypermedia technology to allow much flexibility for students to learn basic GIS concepts is provided by the *GIST* system [Raper and Green, 1989]. The system is capable of combining text, graphics and sound and its hypertext features enable convenient jumps among nodes of information.

---

<sup>2</sup> More details can be found at “<http://www.xanadu.com.au>”.

The hypertext philosophy was one of the building concepts – Universal Readership, Hypertext, Searching, Client-Server Model, and Format Negotiation – when the World Wide Web was originally conceived [Berners-Lee, 1991]. The hypertext concept has become a mainstream interface paradigm with the emergence of the World Wide Web and it is the Web that expands the hypertext horizon from a standalone environment to a vast network of computers in which millions of users can create and retrieve multimedia materials from around the world.

## **2.2 Limitations of Current Approaches to Data Integration in GIS**

Previous discussions on the common approaches to integrating diverse data in a single system lead to a point that commercially available GIS products do not always provide an entirely satisfactory solution to the integration of spatial data and reveal three technical difficulties in formulating a data integration strategy.

### **2.2.1 Proprietary Data Formats**

The crux of data integration lies in the existence of a variety of proprietary data formats. Since most GIS software packages are functionality oriented and regard their programs as the producer of an end-product that never release the internal structure to public. Guptill [1991] explains that 2 pieces of conversion involved in

direct translation between two formats. The number of conversions grows rapidly as more formats involved (total =  $n \times (n - 1)$ ; where  $n$  is the number of formats). It is, therefore, nearly impossible for GIS software packages to support data abstractions for differently specified solutions. Users must take cumbersome efforts for data reorganization for one system to another to suit the target format.

Guptill suggests two alternatives to direct translation scenario to minimize the number of routines needed for data integration: data “switchyard” approach; and standardized, neutral exchange approach, e.g., Spatial Data Transfer Standard (SDTS). The standardized, neutral exchange approach would provide data structure at a logical level rather than at a physical one. This characteristic enabling separation of the data definition and its associative description. Thus, a recipient receiving data can readily see what a unit of data is supposed to locate and can then construct its local representation of the data according to its own specifications.

Even when the data are exported or converted from proprietary internal format to users readable ASCII format, errors may still exist in the output record sets. There are two traditional method of organizing records of data for exchange:

- A. **Fixed fields location.** This method lays out record into fields where values start at specific locations. A fixed amount of space is taken up by every field regardless of the length of the actual record, i.e., position-dependent.

**B. Character delimited fields.** This method uses one of the special characters, for example comma “,” or slash “/”, to delimit the fields of record. The position of the start of a record/field is not fixed and the order of the fields is important, i.e., order-dependent.

These two methods of record layout are the potential sources of inconsistency between data sets. Since data structure or requirements change over time, redundant or obsolete fields need to be deleted and new fields need to be added. A series of modifications will be invoked in relation to the changes of record layout. For example, the translator commands have to be modified to cope with the format change, meaning that a synchronized software upgrade is required for all parties involved.

To address these types of problems, the technique employed in Hyper-Text Transfer Protocol (HTTP) to mark the presentation style of an entity by a pair of tags can be considered. For instance, for the following case,

`<B>This is bold</B>`

the text enclosed by the tag pair will be displayed in bold in a Web browser. This technique benefits the organization of records in two ways. First, the order or position of the fields is not important because the fields and their values are tagged. Second, the addition and/or removal of fields are now easily handled by translators by just ignoring those tags which are not understood. Third, users of spatial data are

provided with flexibility by just extracting those interested fields within a record while ignoring those not interested in.

The following example proposes the layout of a control point using different tags to delimit the attributes of the point:

```
<Point>
  <Name>TRIG237</Name>
  <X>801234.567</X>
  <Y>839876.123</Y>
  <Z>127.321</Z>
  <Control>Primary</Control>
  <DateCreated>2000-3-28</DateCreated>
  <Owner>Department of Land Surveying & Geo-Informatics</Owner>
  <SketchLocation>http://www.lsgi.polyu.edu.hk/
    TrigSketch/TRIG237.JPG</SketchLocation>
</Point>
```

More storage space, longer processing time and more complicated parser are the obvious demerits of this approach. However, current hardware advancement with a reduction in price should minimize the implications. After the detailed discussions in Chapter 3 and 4 on the advancement of the Internet technology, we can see the improvements in data structuring with the application of the Extensible Markup Language (XML) technology.

### **2.2.2 Lack of Data Information**

Many organizations or individuals wish to exchange more than just a graphical image representing a spatial entity. They also want to know the meaning behind the graphic data – also called metadata, lineage – such as the accuracy of the data, the

method used to collect the data, date of creation, the currency of the data, the intended use of the spatial data and so on. Lanter and Veregin [1990] noted that some geographic data are inherently inaccurate or intentionally uncertain. Irrespective of the data, users of spatial data should have an indication of the quality of the data resident in the spatial data set being shared/integrated. With such information provided in the data set, the misuse, misinterpretation, and abuse with spatial data would be reduced and users could evaluate if the quality is appropriate to the task at hand. However, nearly most of the commercially available GIS software packages do not provide any means to transfer or store this kind of meta data information.

### **2.2.3 Data Transmission Restriction**

The data integration process has to inevitably involve data exchange between different hardware platforms. Most of the GIS software packages are using hardware-dependent binary data format for internal storage in a sense to improve the software performance. These binary data format are subjected to specific platform differences such as byte order, word order and floating point representation. An exchange in binary format may result in a compact data transmission, but at the expense of complicated and error-prone transformation of the data. However, character representation of data would avoid these problems with a moderate increase in storage space, which can be compensated by the rapid hardware price drop.



## **2.3 Expanding the Scope of Data Integration**

In the evolution of GIS, the functions performed tend to be the most volatile part. Most GIS software packages are functionality oriented. Since GIS technology has been expanding so fast, software developers will be asked to develop a more realistic GIS software environment in which users could focus on real-world problem solving. Using a group of data conversion routines, user customized menus or user-generated scripts/macros within the context of a particular vendor environment does not make data integration easier. The main reasons are due to the complexity of spatial data and the diversity of data sources to be manipulated. Thus, there are some areas that seem particularly desirable for inclusion into the design of a data integration solution:

- front-end user interface;
- transmission protocol;
- data presentation; and
- data modeling.

## **2.4 A Proposed Framework for Data Integration**

To cater for the expended scopes, a conceptual framework for integration of diverse data sources is suggested in Figure 2.1. The framework provides an

environment that will be independent of the underlying GIS data structure. The proposed system is not just one program, but a set of communicating tasks that are loosely coupled with minimal interdependency. This loosely coupled approach makes it easy to change or update the system as a new task or knowledge is acquired. The proposed system is basically following the three-tier client-server model. There are altogether there are four modules in this approach and they are grouped into three tiers. The application tier takes two roles in this scenario. On one hand it interacts and serves the requests coming from client, on the other hand the application tier interprets and distributes the requests accordingly to data tier.

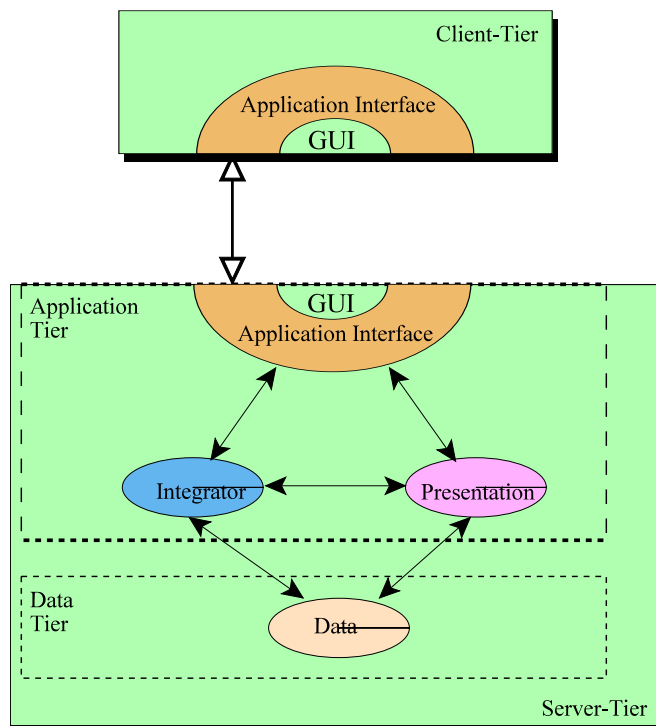


Figure 2.1 Proposed framework for data integration.

### **2.4.1 Application Interface Module**

Although the functions provided in the Application Interface Module is split into two parts in the operational point of view, these two parts are working as a single module contextually. The physical location of the client is not important in the sense that the client and server are virtually working as if in the same system. In this way the users will be presented with a consistent set of functions and graphical user interface across dissimilar platforms. The Application Interface Module is a gateway to Presentation Module and Integrator Module.

### **2.4.2 Presentation Module**

Presentation Module contains a set of actions to be performed on Data Module so that different format of data can be represented to the user. Because the system is targeted to handle multimedia data. The module should provide flexibility for further expansion to incorporate new forms of data.

### **2.4.3 Integrator Module**

The Integrator Module is the core module of the proposed integration system. The sole purpose of this module is to provide functions for users to perform data integration. The module should be able to recognize and convert data from a wide

variety of sources with minimal user interaction. The translation program inside the module will employ the XML and the SDTS. After the detailed discussions on the Web and XML in chapters 3 and 4, it will be made clear that a Web browser possess the capability to interpret the content of the information being received and to invoke the appropriate application program. This means that the Integrator Module would determine the format of each requested file by interpreting the filename extension and/or by reading the information provided by the Web page.

#### **2.4.4 Data Module**

A detailed arrangement within the Data Module is illustrated in Figure 2.2. As the Data Model is proposed to host data from a diversity of formats and sources. The data are stored in native format within the Data Module but split into two categories for easy classification. There is a slight different in handling the data within the data module according to its internal format. Those data formats recognized by the Web server will be directly handled by XML, while those proprietary GIS data formats will be converted in the first instant to a neutral interchange format such as SDTS and then handled by XML.

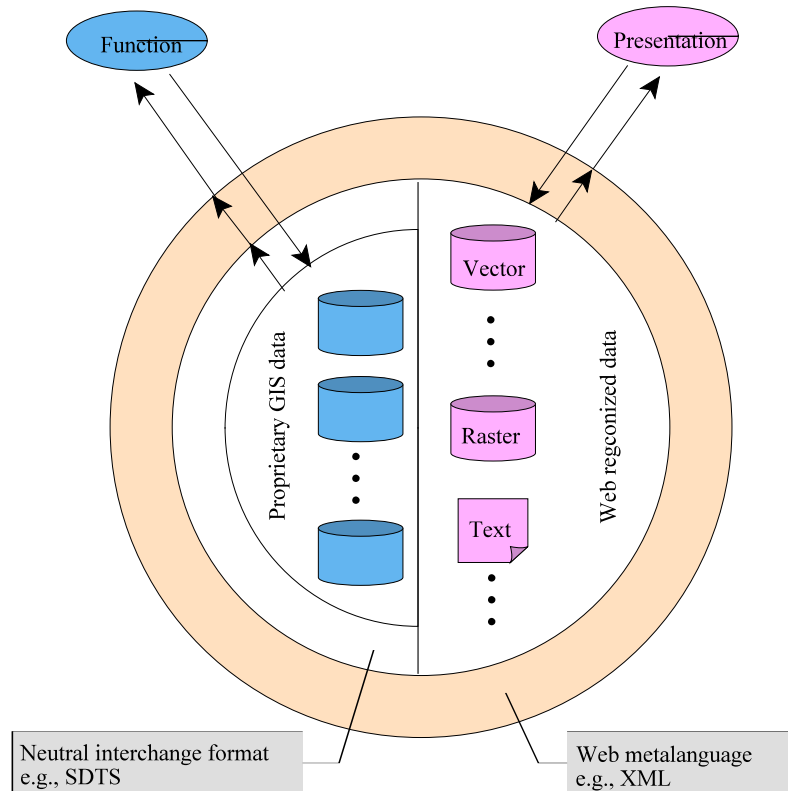


Figure 2.2 Detail arrangement in Data Module.

## 2.5 The Use of Standards for Data Integration

Clearly, a lot of data communication will be involved during the data integration processes. One solution is to adopt standards in every aspect of the process. This is easy to develop and maintain, and flexible for expansion. The reasons for standardization are summarized as follows:

- A. The knowledge gained in using one product is not readily transferable or applicable to another.

- B. Insufficient documentation on the non-standard or proprietary data format and geo-referencing (applied both to vector and raster data).
- C. Lack of skill or training for using a particular vendor data format.
- D. It is difficult and time-consuming to customize proprietary format.
- E. Commercially available GIS products are difficult to learn and use.

## **2.6 Internet as an Integrating Technology**

To address the issues mentioned in sections 2.3 and 2.4 above from a new perspective, perhaps it may be necessary to look to developments in the Internet technology. This has led to the establishment of specifications that define the framework for information interchange across dissimilar systems over the connected network. The proposed specifications used for the development of a data integration solution are listed in Table 2.1 and illustrated in Figure 2.3. Detailed discussions on the rationale behind the suggestions will be given in chapters 3 and 4. Ideally, such techniques will come not only from those who design these tools in the world of Internet technology, but also from those who must ultimately put them to use in that other world just outside – GIS community.

Front-End User Interface	—	Web/Java/JavaScript
Transmission Protocol	—	Hyper-Text Transfer Protocol
Graphics Presentation		
vector graphics	—	Scalable Vector Graphics
raster graphics	—	Portable Network Graphics
Data Modeling	—	Extensible Markup Language + OpenGIS Specification – Web Map Server Interfaces Implementation Specification + Spatial Data Transfer Standard

Table 2.1 Proposed specifications used in data integration

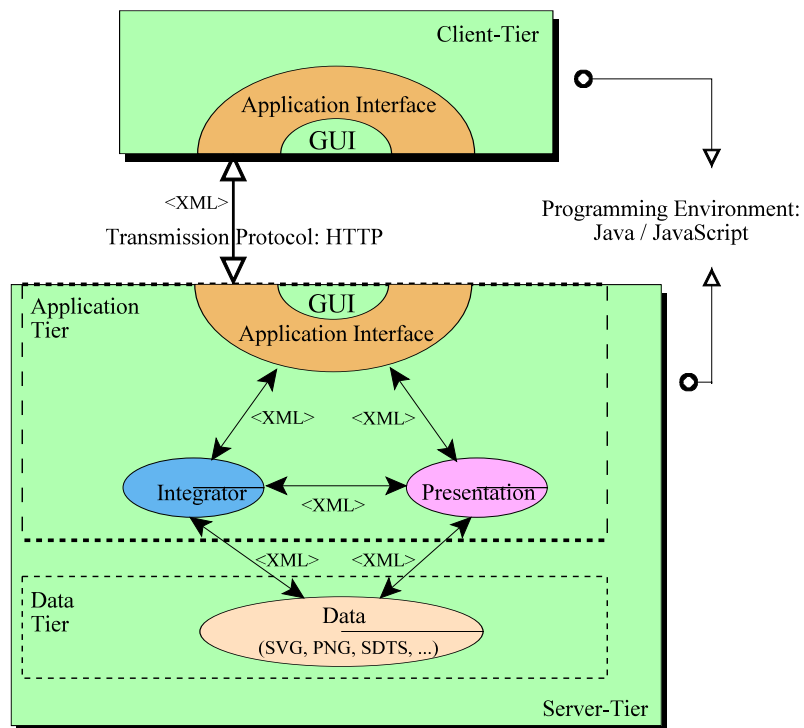


Figure 2.3 The use of standards for proposed framework.

## **2.7 Summary**

In this chapter, an introduction to some techniques for data integration have been given. The basic concepts and operations in the integration of non-spatial and spatial (including raster, vector, and multimedia data) map data were discussed and illustrated. The limitations of current approaches to data integration in GIS have also discussed in terms of data formats, lack of data information, and data transmission restriction.

A three-tier client-server conceptual framework for integration of diverse data sources is suggested. The four components that will be substantially using standards such as XML, SVG and PNG to make up this conceptual framework are described.



## **CHAPTER 3**

### **INTERNET AND WEB PROTOCOLS**

It has taken 30 years for the Internet to get popular and to provide a huge, rich source of information that is globally accessible. The ARPANET (Advanced Research Projects Agency Network) was the most early form of the Internet developed in the late 1960s [Shelly, 1996] as a method for the US government's computers to communicate with each other for scientific and military purposes.

The early development of Internet required the sharing of information via a series of not-so-friendly interfaces and services. In the late 1980s, the World Wide Web project [Berners-Lee, 1994] has made possible the idea of distributing information on the Internet with attractive user friendly interfaces. The rapid growth of the Internet's user base in the past decade endorses the World Wide Web to be a crucial technological entity for the next century.

Today, the World Wide Web uses hypertext to link tens of millions of documents together, and innovations are being added rapidly. This chapter is an overview of the current technologies used and provides a basic understanding of the Internet and the distributed online GIS.

### 3.1 The Concept of the Internet

The Internet is a global, distributed network of computers that use a common protocol to communicate - TCP/IP (Transmission Control Protocol/Internet Protocol). With this communication protocol properly setup in each computer system, a network of what many people refer to as the “Information Superhighway” is made possible. No one owns the Internet and it is a public domain that can be accessed by everyone disregarding the physical location and the operating system of the computer to be connected.

TCP/IP enables any two computers on the Internet to talk to each other at the hardware level and provides the framework for application development. In order to supplement the functionality of the Internet, some other protocols that run on top of the TCP/IP protocol and on the software level are required. The following lists some of the typical services provided:

- A. To communicate between users over the Internet, computers use the Small Mail Transfer Protocol (SMTP). This service provides an electronic message (E-mail) to be delivered from one host to another host on the Internet.
- B. To exchange files through the Internet, computers use the File Transfer Protocol (FTP). This service enables file(s) to be uploaded or downloaded from one computer to another computer.

- C. To establish a terminal session with a remote server to access the computing resources on that server, computer client use Telnet service.
  
- D. To allow collaborative presentation of information with text, graphics, illustrations, sound, video, and any other means via Internet, computers must conform with the Hyper Text Transfer Protocol (HTTP). The World Wide Web (WWW) makes it possible and plentiful for users to traverse Internet documents that contain graphics and highlighted items (hypertext links or just hyperlinks) . The Web browsers let users navigate the Internet not by entering commands, but rather by moving the mouse pointer to the desired hyperlink. With a simple mouse click, users can jump to related web pages, which may be delivered from millions of server computers. Instantly, the World Wide Web service establishes contact with the remote computer and transfers the requested file to a client computer, displaying it in the browser as another formatted, hyperlinked document.

### **3.2 World Wide Web**

The emergence of World Wide Web has pushed the Internet into a new era. As most of the online GIS packages are developed for the WWW, the following section will give a detailed description of the WWW.

### 3.2.1 How the Web Works: Concept

The World Wide Web, or simply the Web, is a by-product of the Internet. It is created because of the Internet's overwhelming amount of information provided. It is important to realize that the Web is a concept and not a software, although a special software (web browser) is required to access the Web. It is also not a hardware although hardware equipment is required to connect to the Internet in order to access the Web. It is not even a specific protocol although HTTP is required to conform with the communication between computers. In the simplest term, the Web is a standard and a specific set of requirements for developing information bases to be distributed over the Internet [Erickson, 1996].

In March 1989, Tim Berners-Lee of Geneva's European Particle Physics Laboratory (which is abbreviated as CERN, based on the laboratory's French name) circulated a proposal to develop a "hypertext system" for the purpose of enabling efficient and easy information sharing among geographically separated teams of researchers in the High Energy Physics community. There were three important components of the proposed system:

- A. A consistent user interface.
- B. The ability to incorporate a wide range of technologies and document types.

- C. “universal readership”; that is, anyone connected to the network can read the same document as anyone else disregarding the physical location and operating system of the computer used.

After the rapid development in the past decade, the Web now contains the technology necessary to give the Internet a flexible and versatile tool to access huge amount of information with less effort. The Web is not just providing links from information source to information source, but also links that are contextually related.

### **3.2.2 How the Web Works: HTTP**

To facilitate the access and display of information between any two computers over the Internet, the Hyper Text Transfer Protocol (HTTP) is established as the standard communication protocol for a distributed collaborative hypermedia information system on the Internet. The Web document presented by World Wide Web Consortium (<http://www.w3.org/Protocols/HTTP/HTTP2.html>) indicates that the transaction has to undergone four basic phases: (1) connection; (2) request; (3) response; and (4) close.

In the connection phase, the Web client attempts to establish a connection with the HTTP server. If the client cannot establish the connection, nothing further can happen and possibly an error message will be issued to inform the client.

As soon as the connection to the server is established, the client sends a request to the server. The request specifies which protocol is being used, and it tells the server what object it is retrieving and the method it wants the server to respond.

Once the request is received by the server, the server will either send error messages to the client if it cannot fulfill or execute the request and will send back the client the appropriate status code and response data.

Finally, the connection is closed by either party. Soon after the successful connection is closed, the web browser invokes a series of actions to complete the user request. The browser loads and displays the requested object in a recognizable format, saves the data to a file, or launches a registered application to display the requested object.

### **3.2.3 How the Web Works: HTML**

The simplest component of the Web is HTML (Hyper Text Markup Language). HTML is a simplified derivative of Standard Generalized Markup Language (SGML<sup>3</sup>, ISO 8879:1986), for formatting documents that are displayed in a Web

---

<sup>3</sup> SGML – The first significant standardized structured information technology, which was created as a result of work done at IBM to provide a means of formatting and maintaining structured documents, such as legal documents and technical documentation.

browser. The primary task of the browser is to render documents according to the HTML tags that are contained and displayed on the screen.

A Web page is basically a text file that contains the text to be displayed and references to elements such as images, sounds, and hyperlinks to other documents. HTML tag is made up of text-formatting element that is placed usually in a pair of angle brackets. The first tag turns on a formatting feature, and the matching tag turns it off. The end tag looks like the start tag except a forward slash (/) character precedes the text within the angle brackets. For example, the tag `<I>` turns on the italic feature and the following will be displayed in italics, until the `</I>` tag is encountered.

Some elements may include attributes, which are keywords with special meanings within a specific tag. The `<A>` tag, which is used to insert a hyperlink in the document, recognizes the HREF attribute. The syntax of a hyperlink to the home page of the Department of Land Surveying & Geo-Informatics on the Web would be something like:

Click `<A HREF="http://www.lsgi.polyu.edu.hk">here</A>` will link to  
LSGI's home page.

The text between the `<A>` and `</A>` tags is marked as a hyperlink. The HREF attribute in the `<A>` tag tells the browser which URL to jump to when a user clicks this hyperlink. A typical example of a HTML page displayed in a Web browser and the corresponding HTML page are shown in Figures 3.1 and 3.2.

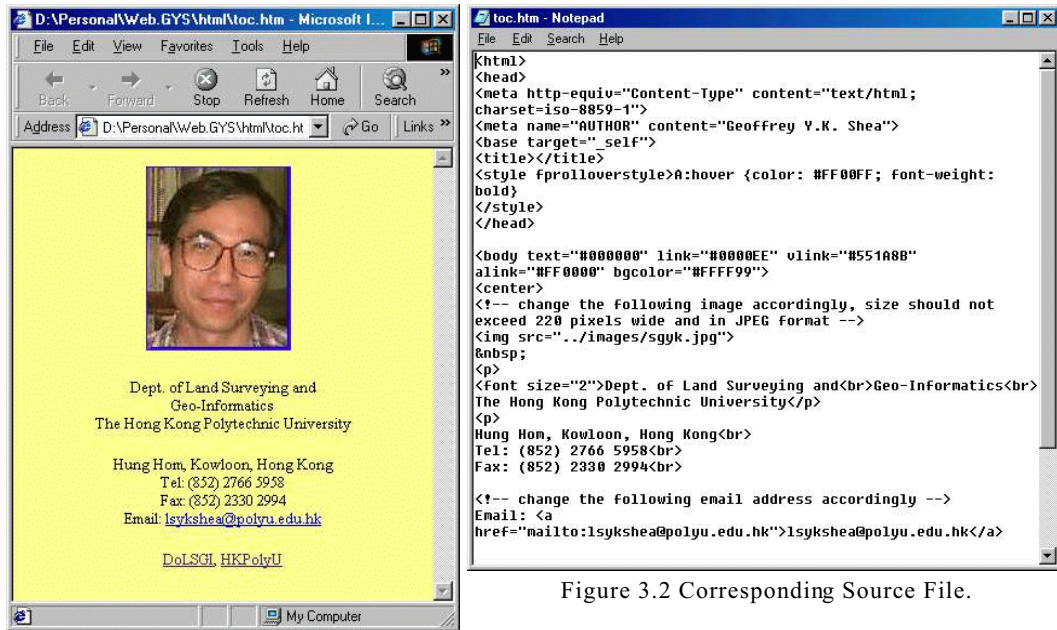


Figure 3.1 A simple HTML document.

Figure 3.2 Corresponding Source File.

### 3.2.4 How the Web Works: URL

To make the Web a reality, a new type of addressing system needs to be developed that can describe not only the location of a file or server, but also its type. The Web uses an addressing system known as a Uniform Resource Locator (URL) to achieve this. For most users, the URL means to the address of a particular web page. In fact, URL contains information of what protocol to use for handling a file and the location of the resource. A URL consists of three separate parts that, when combined, completely define the location of any file or server located anywhere on the Internet. These parts are the protocol, domain name, path and filename. An example of URL



for the Department of Land Surveying & Geo-Informatics of the Hong Kong Polytechnic University is shown in Figure 3.3.

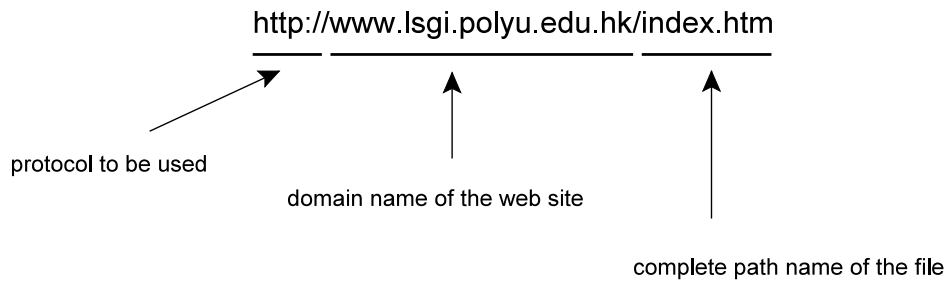


Figure 3.3 An example of a URL.

#### 3.2.4.1 The protocol

The most important part in the URL is addressing definition. This piece of information defines the method for transporting/handling a file over the Internet. Having specified the protocol in the URL, the web browser will understand which port and server it is talking to in order to obtain the information pointed to by the selected hyperlink. HTTP is not the only allowable protocol to be used, other protocols such as FTP, Telnet, mailto and news can also be specified in URL to access other kinds of resources on the Internet.

### **3.2.4.2 Domain Name**

The next item to be defined by a URL is the location of the server hosting the file that the hyperlink pointed to. This item can be either a meaningful name or an IP address. For example, both “www.lsgi.polyu.edu.hk” and “158.132.74.218” are referring to the same web server. It is quite obvious for an infrequent user to use the name of a web site when navigate the Internet because it provides a more easier understandable form to remember than the IP address.

### **3.2.4.3 Path and Filename**

The last piece of information to be defined in the URL may or may not exist depending on the resource the hyperlink is pointed to. For most of the protocols used there is a default file or home directory will be referred if the last part is not entered. For example, the following URL will be pointed to the “index.htm” file resided on the root directory of the web server: “http://www.lsgi.polyu.edu.hk/”. The root directory here does not mean the absolute root in the remote host but can be anywhere in the remote host file system. When specifying the path to a file, the standard UNIX method for path definitions is used by separating each directory by a forward slash (/). Again the directory here is referred to the directory relative to the root directory of the web server defined in the remote host file system.

### **3.3 The Web and Intranets**

The previous sections have addressed those technology applied to the Internet. The descriptions are also equally applicable to intranets. In fact, an intranet is a network of computers that uses Internet technology to operate on the TCP/IP protocol, except that it is not global. Intranets are restricted to the users of a corporation, a university, or certain organization, meaning that they cannot be accessed by the outside world. Many corporations use intranets to provide information to their employees. For external uses, they would run another Web site for security reasons.

### **3.4 Client-Server Architecture**

Client-Server computing model is the key to successful information sharing on the Internet/Intranet and this Client-Server architecture is based on a simple idea: different computers perform different tasks, and each computer can be optimized for a particular task. In the Internet environment, the Web browser acts as the client and the Web server serving the application acts as the server. Figure 3.4 depicts the basic client-server architecture. Basically, the Client-Server model for single Request-and-Reply cycle operates as follows:

- A. A Web browser, such as Netscape or Internet Explorer, makes a HTTP request to the Web server.

- B. Having received the request, the Web server responds to the request and re-directs part of the request to other appropriate resource(s) such as a map server or dedicated GIS.
- C. Once the server has obtained the required information from the re-directed server(s), it then makes a HTTP reply in the form of HTML documents to the client.
- D. The client browser displays the received HTML documents.

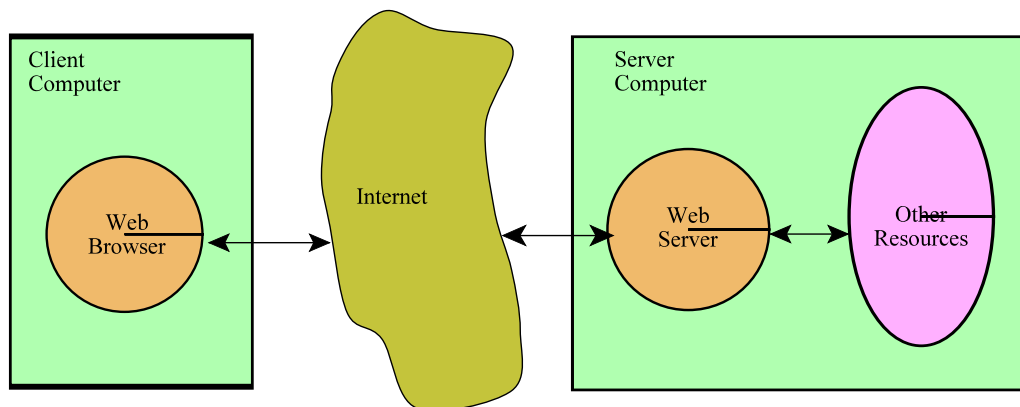


Figure 3.4 Basic client-server architecture.

The client-server model became very popular because much of the processing is done on the client computer, which can be an inexpensive desktop computer. The Web and the Internet applications are all built around this model.

A Web site consisting of HTML pages is interactive only in the sense that it allows the user to jump from page to page through hyperlinks. The client requests

documents from the server, and the server supplies them. Obviously, updating the information entails editing the HTML documents and it is no wonder most sites cannot provide up-to-date information.

Initially, the role of the browser was to render Web pages on the client computer. Since Web pages are simple text files, and contain only information as to how the text should appear in the browser window, neither simple nor complicated calculations have to be carried out on the client side. The hardware requirement for clients to render the Web pages do not have to be powerful. In fact, based on the hardware configuration there are two types of clients:

- A. **Thin Clients.** Thin clients are less-powerful computers that do very little processing on their own. The benefits of thin clients are: (a) low cost of ownership; (b) simple hardware configuration; (c) simple software installation; (d) standardized presentation capabilities; and (e) easy to maintain.
  
- B. **Fat Clients.** A fat client is a desktop computer with sophisticated hardware configuration that provides rich presentation features and better performance.

### 3.4.1 The Two-Tier Model

In the client-server architecture, the application is broken into two distinct components, which work together for a common goal. These components are called tiers, and each tier implements a different functionality. The client-server architecture involves two tiers.

The first tier of a client-server application is the client tier, or presentation tier, which runs on the client. The second tier is the server tier that manipulates a complicated request and response with a simplified view of the data through HTTP protocol. The server tier receives many requests from the clients and it must serve them all. It means that the server is the central point of contact on all requests.

The two-tier model is a very efficient architecture for some applications, but not always the best choice. For example, Web-based GIS applications require large transmissions of data between client and server tiers, and demand for advanced spatial analytical capabilities such as overlay analysis and online map data editing. All these requirements indicate that a heavy burden will be placed on the server tier which will substantially degrade the overall performance of the two-tier model and naturally leads to the introduction of a three-tier model.

### 3.4.2 The Three-Tier Model

A new tier called the middle tier is introduced in between the Client and Server tiers yielding a new three-tier model. The main advantage of this model is that the client can engage in a conversation with the server so that information can flow in both directions or in other words dynamic web content can be generated on-the-fly. A server-side scripting language such as PERL<sup>4</sup> through CGI<sup>5</sup> makes it possible for Web developers to develop dynamic content for their Web applications. The server-side scripting language works together with the Web server as a middle tier to respond to client requests.

Figure 3.5 shows a Web application that runs in a browser and contacts a Web server and a database server to interact with the user. The first tier – the presentation layer – is the browser, which interacts with the user through HTML documents. A Web page may contain scripts where the user can enter information and submit it to the server. The values of the scripts on a Web page must be passed to the server before they can be processed. All requests are channeled by the browser to the Web

---

<sup>4</sup> PERL is an acronym for Practical Extraction and Report Language and is a server-side scripting language with which powerful data and text manipulation routines can be created. It is originally designed for UNIX systems but now is extended to Microsoft's *Windows* system. However, PERL is a cryptic language that *Windows* programmers or *Visual Basic* developers are unfamiliar.

<sup>5</sup> CGI is an acronym for Common Gateway Interface and is a server-side interface for initiating software services. A set of interfaces that describe how a Web server communicates with software on the same computer. Any software can be a CGI program if it handles input and output according to the CGI standard.

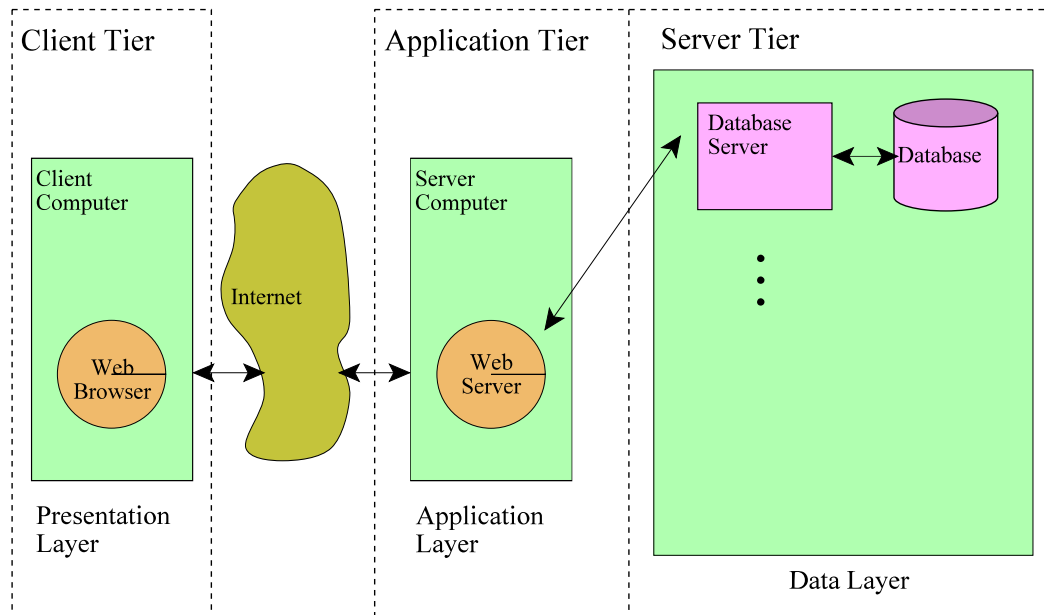


Figure 3.5 A three-tier application.

server, which is the middle tier (the application layer). The Web server's role is to generate HTML documents and send them to the client. If the Web server needs to access a database (data layer), it must contact a database server through a server-side script.

The middle and server tiers of a Web application are normally installed on the same machine for easy maintenance but they run as two separate processes. In order to fully utilize the advantages of a three-tier model, it is a good practice to separate the middle and server tiers apart by installing and executing them on different machines. This arrangement improves the overall performance of a Web application in three ways:



- A. **System throughput.** The system throughput is increased by adding more servers on the server tier. The system loading can also be evenly distributed among the servers.
  
- B. **System reliability.** The system reliability is increased. If one of the servers on the server tier was crashed and is unable to provide the service, the system is only partially affected.
  
- C. **System functionality.** The system functionality is enhanced. A dedicated server on the server tier can be added to provide more functions whenever any special need arises.

### 3.5 Summary

In this chapter, the fundamentals of the Internet and the concepts of the Web have been discussed. The relationship between the Web and Intranet is also described. The basic client-server computing architecture and the variants – the two-tier model and three-tier model – of such architecture are discussed. Their relative advantages and disadvantages are also given. The thorough understanding of these concepts in three-tier computing architecture is important to the success of building an efficient web-based application such as in this project.

## **CHAPTER 4**

### **ONLINE GIS**

If there is one term that caught up literally overnight and has affected more users than any other, it is the Web. Along the same line the term “Online GIS” is becoming more and more important. Online GIS is a methodology for building distributed GIS applications on the Internet.

A distributed GIS application is made up of multiple components that run on different machines. These machines can be interconnected through a local area network (LAN or Intranet). This can be a few machines on an Intranet or a few more machines on the Internet.

The idea of Online GIS starts with the realization that not all information are stored in one single place or in one single format. A lot of information are stored in text documents, spreadsheets, proprietary data format or even audio and video files. The ultimate and ideal data access technology is one that can access any information of any data format from anywhere in a uniform way.

Users of Web applications are now expecting more functionality provided, such as more attractive user interface, secure enough to protect personal privacy, and

better performance over the Internet. This chapter will give a brief discussion on the design issues for Online GIS taking into consideration of its interactive, computational-intensive characteristics and requirement on high-resolution graphical user interface.

## **4.1 Online GIS Architecture**

An online GIS application (sometime referred to as *Web-enabled GIS* or *Internet GIS*) is a Web application fully charged with GIS functionality. Usually the applications are emerging by extending the traditional GIS packages to support the Web technology.

Basically, online GIS follows the Client-Server Model with Web browsers as the clients and the Web site serving the application as the server. There are two variations to the basic online GIS application: (a) the Client-side; and (b) the Server-side applications [Gifford, 1999].

### **4.1.1 Client-Side Online GIS**

The client-side online GIS architecture (Figure 4.1) is a highly client-dependent platform configuration which requires the client machine to take up all the responsibilities of processing GIS operations. Since all the core GIS operations such

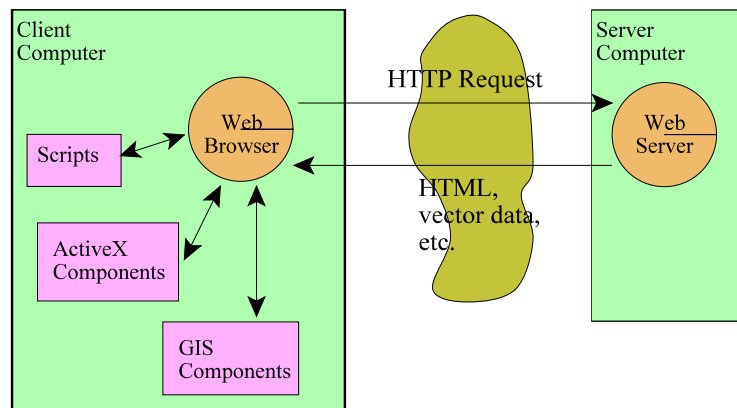


Figure 4.1 An example of Client-side application.

as spatial query, buffering, overlay analysis and network analysis are heavily loaded in the client machine, the hardware configuration in the client machine is critical to the success of this approach. The server's job is to extract the required data from the database and furnish them to the client. Once the data are on the client computer, the application is quite capable of manipulating the data locally. The more powerful the client is, the more processing it can do. Two clients may receive the same data from the server but different outputs of the GIS operations in terms of processing speed and graphical presentation may be expected. This client-server model involves two or more tiers.

As mentioned in Chapter 3, HTML is a document-formatting language that cannot be used to carry out even simple calculations such as additions or to display the data in the browser window. To perform the GIS operations in the client machine,

the Web browser must be assisted with client-side scripting languages<sup>6</sup>, Web-enabled programming languages such as Java (to be discussed in later section), and Web browser plug-ins (e.g. *ActiveX* components<sup>7</sup>) whenever necessary.

Another feature of client-side online GIS, which makes it suitable for Web-based GIS applications, is data awareness. Each data-aware page contains not only the HTML information, but also proprietary vector data that support GIS operations in the client computer. Developers can create complicated GIS operations in the client computer to manipulate the proprietary vector data that usually offer more efficient and flexible data processing than the generic Web-recognized data format.

Despite the good features mentioned above, there are several disadvantages exposed by the client-side approach that need to be remedied in order to provide a wider acceptance by the GIS community. First, the security issues related to the use of vector data in the client computer must be addressed so that the ownership and the copyright of the vector data will not be abused. Second, the vector data to be

---

<sup>6</sup> A client-side script is a kind of program that consists of a set of instructions for a Web browser to interact with the user. A script can be embedded in a Web page. *JavaScript* and *VBScript* are two most popular scripting languages to be used.

<sup>7</sup> As Java made a great success in the network market, Microsoft responded to this market opportunity by modifying its Object Linking and Embedding (OLE) controls to be network enabled and renamed it as the *ActiveX* controls [Strand, 1997]. An *ActiveX* component is a general term that encompasses a compiled software component that encapsulates a set of user interface functions. The *ActiveX* component can be executed either on a client computer or on a server computer and be driven by a scripting language such as *VBScript*.

transmitted between server and client must follow some kind of standard such as the Scalable Vector Graphics (SVG) proposed by the World Wide Web Consortium (W3C) ensuring the transmitted vector data can be manipulated across multiple platforms. Third, the frequency of vector data transmission between server and client must be optimized in order to reduce the high usage of bandwidth in the Internet.

#### 4.1.2 Server-Side Online GIS

The server-side online GIS architecture, shown in Figure 4.2, requires sophisticated hardware configuration on the server platform to handle the highly process-demanding requests from the clients. The Web browser on the client side plays a

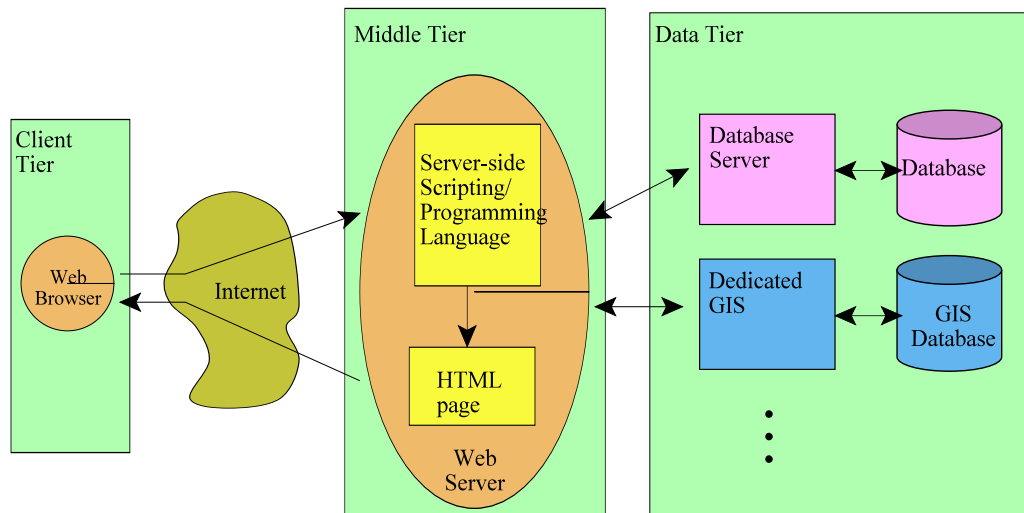


Figure 4.2 An example of Server-side application.

passive role in this approach – it only generates HTTP requests and waits for the results from the server and displays immediately on the client computer without taking any responsibility for processing GIS operations. The Web server, however, plays an active role to handle HTTP requests. The dedicated servers on the server-side work together as a group extracting “live” data from the database, producing the required GIS data, formatting the data as an HTML document, and furnishing them to the client with the help of Web server. In this manner, the client sees up-to-date, accurate information.

The server-side approach can eliminate the problems related to data incompatibility, data inconsistency and data unreliability. This is because all the GIS operations are channeled by the Web server and processed by dedicated data servers. All clients use the same graphical user interface to perform the same set of GIS functions on the same set of GIS data stored on the database(s) that provided by the servers.

The server-side approach reveals, however, a major drawback that needs to be tackled immediately. Since servers must transmit GIS data to clients for every GIS operation over the Internet (e.g., users reset a display window by panning or zooming, turn a layer on or off, make a spatial analysis), a heavy loading in the network is unavoidable. The situation is even worse if many clients are connecting to the server with slow modems.

The server-side online GIS is by nature a three-tier client-server environment.

Figure 4.2 shows the three tiers for a typical Web-based GIS application. At least one map server, which is a dedicated server running with the Web-enabled GIS package, is involved in the data tier to handle map related requests from clients via the Web server. Figure 4.3 shows an example of a Web-enabled GIS package from the Environment Systems Research Institute (ESRI) [ESRI, 1998] that adopts the server-side approach.

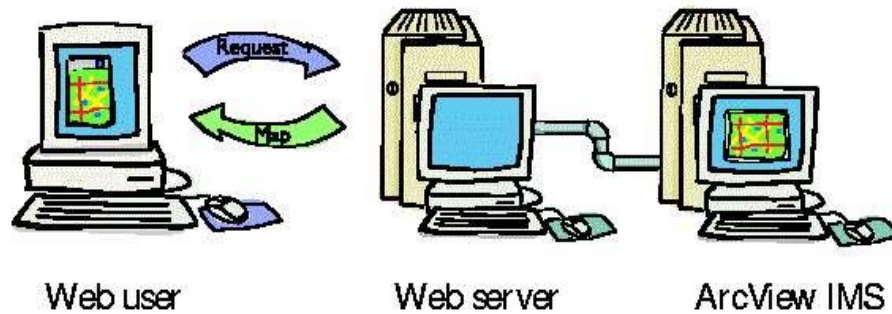


Figure 4.3 Architecture of *ArcView IMS* (adapted from ESRI[n.d., p.2]).

HTML is a great language for displaying information on the client. However it was not designed to be a programming language, and no matter how many extensions are introduced, HTML will never become a proper programming language. In addition, the Web browser takes a passive role in the server-side online GIS application that does not take up any programming responsibility. Therefore, a programming tool which acts as an agent on the server-side is required to channel messages between Web server and dedicated servers. The following sections will



describe the current programming technology employed in developing interactive Web-based GIS applications.

#### **4.1.2.1 Server-Side Programming Technology - Java**

Java is an object-oriented programming language developed by Sun Microsystems and it is best known for its capability to work in conjunction with the Internet. The Java-enabled Web browsers such as *Netscape Navigator* and *Microsoft Internet Explorer* can download a Java program from a Web page and run it locally on the client. These programs, which are called *applets*, appear in a Web page in a similar fashion as images but can be interactive – taking user input, responding to it, and presenting dynamic content.

One of the significant advantages that Java has over other programming languages in developing Internet applications is its platform independence – the capability of the same program to run on different platforms and operating systems. Java programs achieve this independence by the use of a *virtual machine* (VM). A virtual machine is a sort of computer-within-a-computer. Different platforms have different VMs. The virtual machine takes compiled Java programs and converts their instructions into commands that an operating system can handle. First of all, the Java source programs have to be compiled into bytecodes which are platform independent codes, and then the compiled program can run on any platform and operating system that has a Java virtual machine.

Because of the platform independent characteristic, Java applet or application is a suitable tool for Web-enabled GIS application development especially in designing those Graphical User Interface components. A typical client-server interactions [Strand, 1997] involved in a Web-enabled GIS application using Java applet will be as follows:

- A. User in the client-side requests to retrieve a set of geographic data by invoking a client-side's Java component through a Web browser.
- B. Web server receives the request and sends to a CGI to execute.
- C. The CGI script interfaces with the Common Object Requests Broker Architecture<sup>8</sup> (CORBA) services using the standard Interface Definition Language statements to retrieve the geographic data stored in a database.
- D. The results are furnished to the Web browser for display.

Another significant characteristic of Java that benefits the development of Web-enabled GIS applications is its intelligent handling of multithreading – a technique in which a process, executing a program, is divided into threads that can run simultaneously; a thread is a part of a program that is set up to run on its own while the rest of the program does something else. Multithreading is useful for applications that perform a number of essentially independent tasks that do not need to be

---

<sup>8</sup> An Object Management Group (OMG) specification for the interface definition between OMG-compliant objects. OMG is a vendor alliance formed to define and promote CORBA object specifications.

serialized. An example is a GIS package running on a server that listens for and processes numerous Web client requests. While one thread might be processing a spatial query, another thread could be handling spatial data editing. With multiple threads running within the same process, switching back and forth among threads involves less processor overhead than a major process switch between different processes – it means an overall performance improvement is gained.

To summarize, a scalable and efficient online GIS application can be achieved by employing the two major characteristics of Java – platform independence and intelligent multithreading – in designing stage.

#### **4.1.2.2 Server-Side Scripting Technology - Active Server Pages**

The disadvantage of gateway programs, e.g., programs created by PERL, is that they are difficult to implement and maintain. To simplify CGI programming on the server-side especially for Visual Basic developers, Microsoft introduced Active Server Pages (ASP) technology in 1996. Active Server Pages provides a framework on the server that allows developers to create interactive and dynamic Web-based applications using *VBScript* (it is possible to use *JavaScript*) commands and *ActiveX* components.

Active Server Pages are basically HTML pages that contain *VBScript* codes, which is executed on the server, and *ActiveX* components. With ASP, Web

applications can contact the objects<sup>9</sup> and *ActiveX* components exposed on the server to carry out complicated data processing, to access a database to store and retrieve information, and so on. A typical client-server interaction with ASP supported server will be as follows:

- A. A Web browser requests an .asp file residing on an Active Server – A server that supports ASP.
- B. The Web server calls ASP that reads data submitted by the client and the requested .asp file from top to bottom and executes any recognizable script commands.
- C. The ASP produces, where necessary, text and HTML tags on-the-fly.
- D. The Web server sends the result in the form of a Web page to the client.
- E. The Web browser receives the result and renders it on the screen.

The major limitation of ASP scripting is compatibility issue. Currently two Web servers from Microsoft support ASP: (a) the *Internet Information Server*; and (b) the *Personal Web Server*. This situation restricts developers to designing Web-enabled

---

<sup>9</sup> In object-oriented programming, an object is a variable comprising both procedures and data that is treated as a distinct entity. The object's procedures and data cannot be directly accessible by other software or object. Instead, objects interact by means of its well-defined interfaces (methods). The client can then call these methods to perform operations.

GIS applications solely on Microsoft products and provides less flexibility in application development stage.

However, there are four favorable characteristics [Microsoft, 1997] that must be considered when developing Web-based applications using ASP:

- A. **Security.** With ASP, server-side scripts are stored and run on the server rather than on the client, only the results of the scripts are sent back to the client. This means users can only view the Web pages created by the scripts but not the script commands that created the page.
- B. **Component-based flexibility.** All the standard functionality to build an interactive Web page comes in the form of objects. Microsoft provides a whole suite of objects to handle the frequently used situations. Developers can build their own objects to address specific requirements. This flexible arrangement allows a complicated Web application to be developed.
- C. **ASP applications isolation.** ASP applications can be isolated to run in a separate memory space from the Web server in order to protect other applications running on the server. This means an ASP application can be stopped and unloaded from memory whenever necessary without stopping the Web server.
- D. **Database accessibility.** ASP works together with *Microsoft Data Access Components (MDAC)* to provide easy-to-use, flexible programmatic

access to all types of database, both relational and non-relational, under a wide variety of application scenarios: (a) Intranet Client-Server; (b) Internet Client-Server; (c) Microsoft Windows LAN Client-Server; and (d) Stand-alone. Evangelos Petroustos [Petroustos, 2000] demonstrates a complicated but easy to implement Web application which involves database searching and updating via ASP and *MDAC*.

#### **4.1.3 Balance Between Client-Side and Server-Side Solutions**

In general, the client-side solution provides a better working environment with powerful analysis functions. This approach is favorable for a smaller group of sophisticated users who are looking for complicated GIS functions. To achieve the best performance for this approach, additional add-on components (or *plug-ins*) for the Web browser are required to be developed and extra effort on software and plug-ins maintenance is also required.

Obviously, the server-side solution offers a standardized and economic GIS solution to a wider group of infrequent users who are not eagerly looking for a highly-responsive GIS server. As all the processing is done by the server and all the processed information are returned back to the client in the form of HTML documents, the server-side solution demands a better and powerful hardware configuration for the server.

Employing either one of these approaches in online GIS is a matter of choice between performance and functionality. With the advancement in the development of Internet technologies, the classification of the server-side and client-side approaches is no longer easily distinguishable as before. Currently, mixing the two approaches to provide an optimized online GIS application is the mainstream.

Achieving a highly-responsive dedicated online GIS system that can provide complicated GIS query and analysis functions for the users to manipulate remotely is one of the main challenges facing the developers. The burden of achieving such a system is shared by the server-side approach and the client-side approach.

## **4.2 Online GIS Data Model and Format**

Much of the Web-enabled GIS applications are currently using either native web-recognized bitmap format such as GIF or proprietary vector format in transferring data between client and server and then rendering the results on the client side with HTML. There are several drawbacks when using these bitmap data formats and HTML presentation format in online GIS applications development.

First, neither GIF nor JPEG provides intelligent information about the data being transferred, it means that the major characteristic of GIS data – providing intelligence between graphic elements and textual information – will not be preserved in the process of data communication over the Web. If the GIS data intelligence is lost

during the transferring process, much of the interactive and sophisticated functions cannot be performed.

Second, it is difficult to get high-quality results when scaling or transforming a bitmap. The Web community is still awaiting a standard vector format for exchanging vector data. In the transitional stage, a diverse proprietary vector format has been adopted by the online GIS developers. This means users have to download/install a lot of plug-ins to display the data appropriately. The users also have to switch between different plug-ins when manipulating different GIS applications. This heterogeneity makes it difficult for users to combine data from multiple sources. Obviously this is a cumbersome design and undesirable from the users' point of view.

Third, it should be kept in mind that HTML is just a data formatting language designed as a means of presenting static information for display purposes only. HTML is now heavily burdened with lots of expectations such as presenting dynamic information with interactivity, serving as a means of storing specific types of data and used as a database interface that it really is not intended to. All these involve changing its original nature. Even more significant than this is that HTML does not provide extensibility, structure, and data validation needed for large-scale development of data-centric Web applications [Bosak, 1997]. All these data-centric applications are now seeking a development infrastructure that could provide a flexible and extensible data model on one hand, and could provide a standard and



scalable graphic format to handle all kinds of graphic data including raster, vector and text on the other hand.

The development of eXtensible Markup Language (XML) and Scalable Vector Graphics (SVG) from World Wide Web Consortium (W3C) are addressing the above problems. The establishment of such an extensible data model and open vector format is critical to the successful development of online GIS applications. The following sections exploit the potential benefits of online GIS applications which adopt these new technology.

#### **4.2.1 User-definable Data Modeling on the Web**

HTML has proved to be widely successful. However, members of the W3C realized that a much more extensible alternative to HTML would be necessary to cope with the increasing demands of data-centric Web applications that require more functionality beyond the current HTML. One of the main problems of using HTML in these applications is that information cannot be precisely defined with the limited tags set in HTML. For example, traveling agents like to define their own sets of elements for maps, flight schedules, hotel information and so on. The GIS community is more concerned with maps, scale, land use and road network. It is obviously understood that a huge amount of elements have to be derived within HTML in order to cater the needs of all trades and users. This is an ineffective task

that will involve substantial modifications to the HTML specification and is hard to augment HTML to express highly structured information efficiently.

#### **4.2.1.1 An Overview of XML**

In 1996, the W3C has set up a working group to deal with those problems related to HTML as being the vehicle for data-centric applications. The Extensible Markup Language (XML) Specification v1.0 then produced by the working group was accepted by the W3C as Recommendation in 1998.

Basically, XML is a specification for designing platform-independent markup languages in text format that is unambiguous and that avoids HTML common pitfalls. XML only defines a framework upon which solutions can be created. The premise behind XML is the creation of custom tag sets to encode specific types of information. Tag is used to provide the description of a document storage structure and logical structure. XML allows the separation of the meaning or semantics of the data from the way it is used by an application or rendered on the screen or output device.

XML is a simplified subset of the SGML. While XML is being simplified to deliver structured content over the Web, the following design goals stated in the XML Specification Section 1.1 [W3C, 1998] are fully satisfied:

- A. XML shall be straightforwardly usable over the Internet.

- B. XML shall support a wide variety of applications.
- C. XML shall be compatible with SGML.
- D. Programs shall be easily written to process XML documents.
- E. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
- F. XML documents should be legible and reasonably clear.
- G. The XML design should be prepared quickly.
- H. The design of XML shall be formal and concise.
- I. XML documents shall be easy to create.
- J. Terseness in XML markup is of minimal importance.

As a meta-language for describing structured information, XML does not do much by itself. In order to develop interactive, dynamic data-driven XML applications, developers must use XML in the context of an enabling technology. This enabling technology can come in a variety of different forms: (a) programming languages; (b) scripting languages; and (c) style sheet languages.

#### **4.2.1.2 XML-based Applications**

The applications that will drive the acceptance of XML are those that cannot be accomplished within the limitations of HTML. Although possible XML-based applications are almost endless and the full extent of XML's impact on the Web still is not clear, Jon Bosak describes four application scenarios [Bosak, 1997] that would be enriched by employing the new XML technology: (a) heterogeneous database applications; (b) client-server applications with more proportion of client-side processing load; (c) applications requiring multiple presentations of the same data set; and (d) Artificial Intelligent (AI) applications providing tailor-made information according to clients' preferences.

#### **4.2.1.3 Potential Benefits of XML-based Online GIS Applications**

There is next to none XML-based GIS application available on the market as of this writing. But based on the facts that XML is in the unique position of being embraced by all of the leaders in the computer industry and strongly supported by the W3C, we could be confident that XML will flourish in the next few years. The followings are the potential benefits for a XML-based online GIS applications:

- A. **Meaningful data definition.** XML makes it possible to create user-specific tags for defining their own data domain, to handle large and complex data, to manage large information repositories. This is the key

motivation for developing XML-based online GIS applications and is the basis of its usefulness.

- B. **Model/view separation.** The main point of XML is that GIS developers, by defining their own markup language, can describe their highly structured information much more precisely with a rigid hierarchy than is possible with HTML. This means that programs processing these information can understand them much better and therefore process the information in ways that are impossible with HTML. Ironically, the possibilities for error arising from data exchange will be reduced by avoiding formatting tags in the data, but marking the meaning of the data itself with user-defined tags.
  
- C. **Platform independence.** XML makes data accessible in a more meaningful manner and greatly enable data sharing among any device from the PC with a browser to the cellular mobile phone and small PDA. The result can be tailored to the output platform and rendered differently on the client device.
  
- D. **Application simplification.** XML derives from a philosophy that data belong to its creators and that data are best served by a standardized open data format which allows applications to freely compete based on their feature sets [Bosak, 1997]. The prevalence of XML data will drive Web browsers and applications to be XML aware.

- E. **Database accessibility.** One of the principal uses of structured information in GIS community is to enable data interchange from multiple heterogeneous databases. However, users are often complaining that they are forced to read and write many different data formats stored in different databases. With XML extensions such as XML Query (a query language similar to SQL for specifying queries within XML and, at the time of writing, XML Query is still a W3C Working Draft for review) XML-based Web servers can access data from multiple databases and conventional file system and then return the results in HTML or XML.
- F. **GUI flexibility.** XML promises a better approach to graphical user interface (GUI) transmittal to Web browsers across the Internet. The dynamic interface elements will be transferred to Web browser in a way similar to that map data are sent in HTML. The XML would include attributes to generate various interface similar to those in HTML forms, but more complete [Plewe, 1997]. This is the most important marketing area for software vendors showing off their development capability to attract potential users.

#### **4.2.1.4 Modeling Online GIS Applications with XML**

The guaranteed platform-independent inter-application data interchange solution from XML technology (which is one of the principal XML design goals for the

Internet) and the potential benefits as mentioned in previous sections reveal that online GIS applications fit nicely in the evolving XML framework.

Different industries create consortia to specify the content model – DTD<sup>10</sup> in the jargon of SGML – to better suited the needs of their information, such as CML<sup>11</sup> for chemists and MathML<sup>12</sup> for mathematicians. This means that they can share their information easily and efficiently with each other. Surely GIS community will benefit immediately from the XML phenomenon. There is already an application-oriented standard – Spatial Data Transfer Standard (SDTS) – in place for exchanging geographic and cartographic information in the GIS community and this is a good opportunity to dissolve SDTS into the larger XML technology to produce a kind of GIS markup language similar to MathML or CML. The GIS developers could envision a broad range of XML-based applications like a central GIS data warehouse supplying XML encoded data for all walks of spatial data customers. Customers can find their services and search for spatial data precisely as if they were in an ordinary relational database. They can also purchase it regardless of the platform that serves

---

<sup>10</sup> DTD is an acronym for Document Type Definition and is the definition of a markup language. DTD forms the basis of XML documents because it establishes the grammar of an XML vocabulary.

<sup>11</sup> CML is an acronym for Chemical Markup Language and is an XML vocabulary used to describe chemical information. More information on CML can be found at <http://www.xml-cml.org/>

<sup>12</sup> MathML is an acronym for Mathematical Markup Language and is an XML vocabulary used to model mathematical equation. More information on MathML can be found at <http://www.w3.org/TR/REC-MathML2>

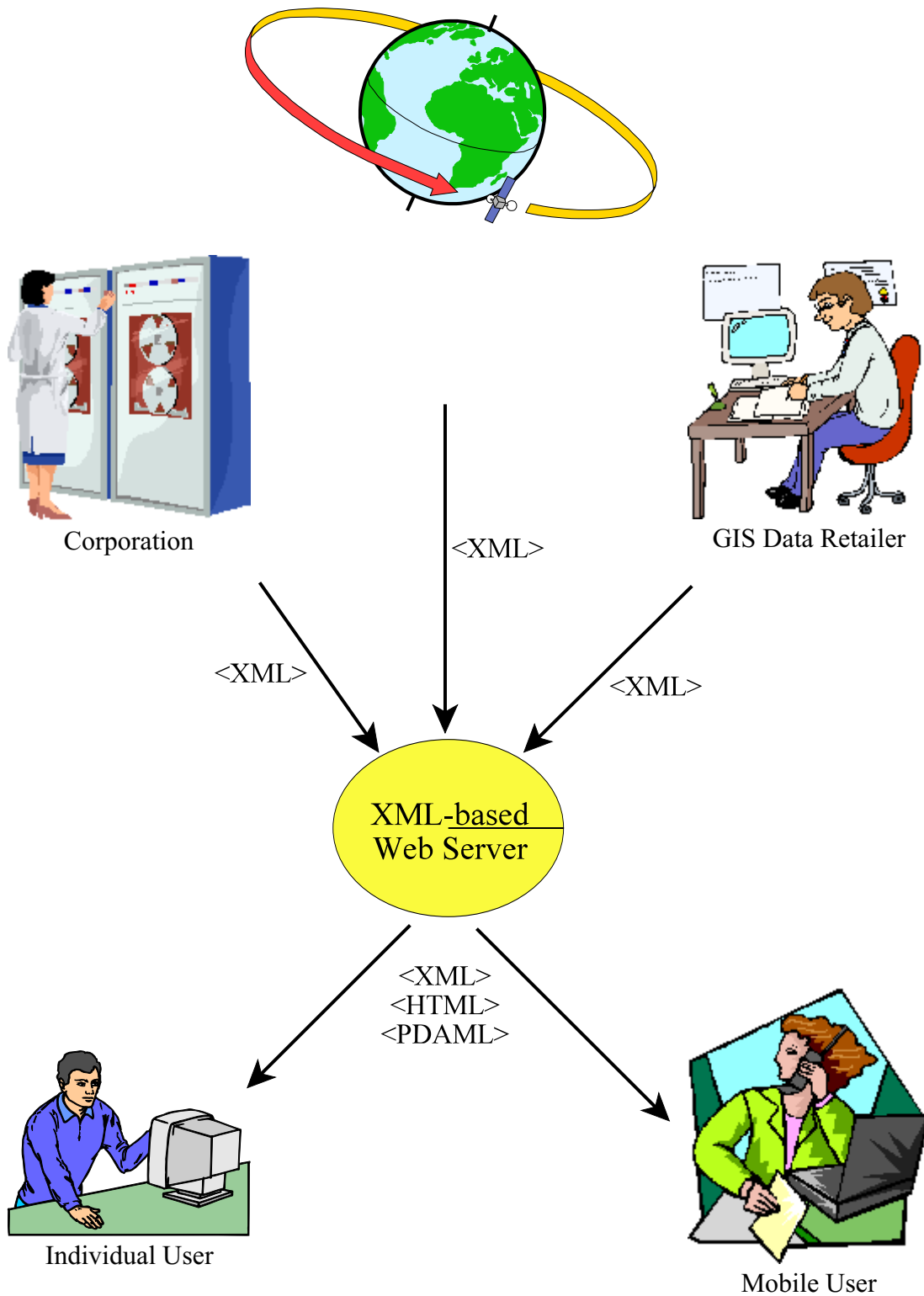
the data, or the platform that the client is using. Figure 4.4 depicts how meaningful data sharing across the Internet can be applied to XML data in the GIS community.

In order to take the full advantage of XML phenomenon and its promises, the transition from a legacy data model to XML object-oriented model must be addressed technically and strategically. Strategic issues related to the transition process will not be covered in this thesis due to their complexity and diversity, which are worth a separate topic. From the technical point of view, information conversion involves a three-tier client-server architecture: a database back-end, an application tier where the application logic acts on the data; and the client where the data are displayed and further processed. The database can receive information from multiple databases either locally or remotely. The application tier can then pull together the data and push it to the presentation tier. Figure 4.5 shows the various components and actions within the XML framework.

Opting for XML is a bit like choosing SQL for databases: developers still have to build their own database and their own programs/procedures that manipulate it. This depends on what approaches and facilities the developers implement. It can simply be said that XML is a family of technologies. The following sections suggest a generalized approach to implementing online GIS applications with this family of technology.



Figure 4.4 XML makes data sharing from any device.



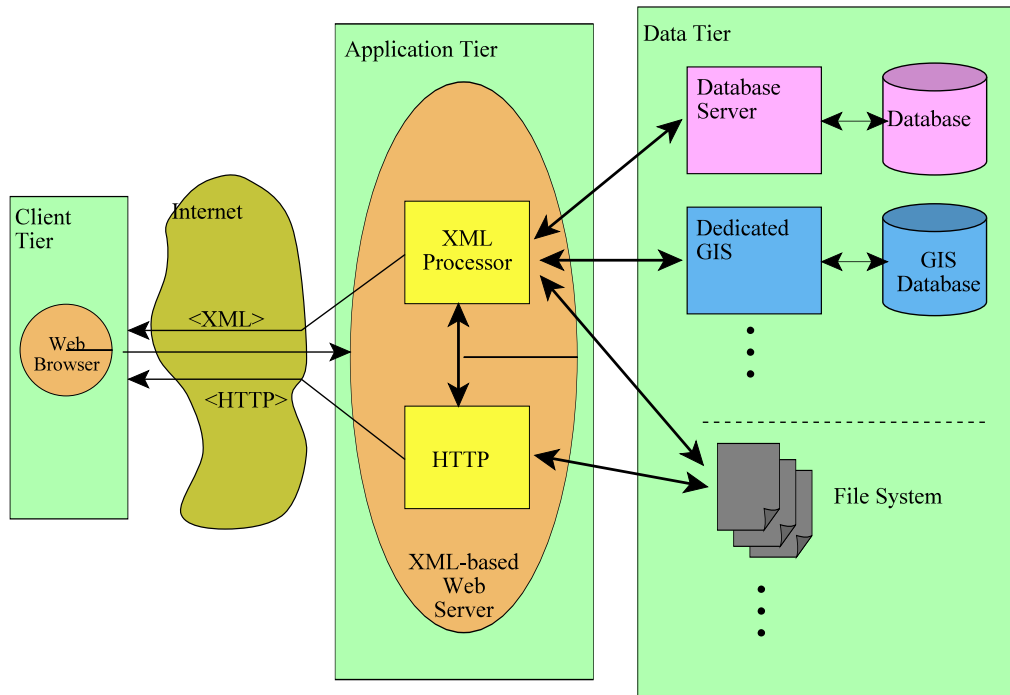


Figure 4.5 A typical three-tier client-server XML-based application.

#### 4.2.1.4.1 Analyze the information requirements

The first step in migrating a data-centric application to XML is the data conversion process. The conversion process definitely will come across data that are stored in any number of ways, including text files, databases, and Web pages. With regard to the sheer size of GIS databases and the diversity of the database definitions, converting the data manually and individually is not a feasible option. Therefore, an automated approach to XML data conversion with the help of qualified consultants is necessary both from the information provider stand point and end-user stand point.

Because of the underlying principles that XML is a metalanguage for expressing *structure* in data and data belongs to its creators, XML does not do much by itself. In

order to ensure the information to be maintained is meaningful, efficient and effective, a critical analysis of the information semantics is important. The expressive power of a specific XML data model is highly related to the degree of understanding in the specific data domain. To view XML data in a meaningful context, it is important to have a solid grip on the data structure itself. The basic building block of an XML document is the entity, which contains either parsed or unparsed data. Parsed data consists of characters that are considered either character data or markup and that are processed by an XML processor<sup>13</sup>. Unparsed character data is handled as raw text and is not processed by XML processor.

There are two fundamental approaches to modeling XML data: (a) modeling data with DTD; and (b) modeling data with XML Schema. A schema, a term borrowed from the database world, describes the arrangement of markup and character data within a XML document. Although technically DTDs and XML Schemas are both considered document schemas and either approach is sufficient for modeling the vast majority of XML documents. However, in situations where data exchange among applications, objects, or databases, the XML Schema provides significant benefits in terms of power and flexibility over the traditional DTD approach to describing the information structure. There are two major benefits worth mentioning: (a) XML Schemas are based on XML, which means that creating schemas using XML Schema is very much like creating any other XML document; and (b) XML Schemas present

---

<sup>13</sup> XML processor – A software module that reads an XML document and provides access to its content and structure. XML processors typically process XML documents on behalf of applications.

an open-ended data model, which allows schema extensibility by incorporating the elements and attributes of another schema and establishment of inheritance relationships between elements.

There are three scenarios when XML is implemented to tackle data management:

(a) use XML as a data store in its own right; (b) use XML as a data store in association with external databases; and (c) use external databases only.

#### 4.2.1.4.2 Analyze the application requirements

Once the relevant information models and their expressions in XML are constructed, the effort to evaluate the applications and tools can proceed. The application requirements is varied from case to case, but XML provides an open data format for applications to compete based on their feature sets, and this open data format scenario exposes the full value of the information and, more importantly, enable hassle free data sharing. This means programmer can thus concentrate on the expertise and value-add of the application domain.

XML is purely a content-based metalanguage that by itself is no more than a storage system for data, but many of the more interesting XML applications require the sorting and searching of an XML document in order to present certain views to client. This is done by external applications to interact with objects exposed by Document Object Model (DOM). This W3C DOM's API (Application Programming

Interface) is the means by which applications can dynamically access and modify the content and structure of XML documents. Besides applications support DOM which can communicate with each other even if they are written using different programming languages. No matter which database scenario is implemented, XML Query and DOM work together to provide a consistent syntax to query data in all types of applications across several different platforms and maintain the promise of XML as a truly interoperable language.

Clearly, hyperlinking will play a critical role in the success of online GIS applications. XML provides a good opportunity to redesign hyperlinking on the Web and augment it to a powerful mechanism for linking XML content in a variety of new ways such as compound links, bidirectional links, and out of line links. The XLink technology, which is the second component of the XML family of technology envisioned by the W3C and is still in the working draft phase of development as of this writing, accomplishes this excellent linking task.

#### 4.2.1.4.3 Analyze the presentation requirements

By separating structure and content from presentation, the same XML source data can be written once, then displayed in a variety of meaningful ways. For example, a closed polygon (this might be a resulting polygon after performing a certain spatial operation on data from different sources) might turn green and a small table is popped up, listing the associated information when a user mouses over it. The

displaying table can also be sorted by any column. Because of this separation, developers need a new way to control design, display, and output issues. A general mechanism for applying formatting styles to XML documents is used. Among the three approaches to styling XML documents currently adopted – Cascading Style Sheets (CSS); Extensible Style Language (XSL); and Document Style Semantics and Specification Language (DSSSL), XSL offers developers and users more presentation flexibility in conjunction with a programming language such as Java or *JavaScript* in handling XML documents. Users could, for example, embed a rendering algorithm such as “shade all XML graphic elements in green if an attribute of an element has a value of `Open_Area`.”

#### **4.2.2 Vector Graphics for the Web: SVG**

Most of the graphics on the Web today are in bitmap format, such as GIF or JPEG. It is difficult to get high-quality results when scaling or transforming a bitmap. Although bitmap graphics are great for scanned images, such as photographs and satellite images, many data-centric Web applications, in which online GIS is one of them, require high-quality, interactive vector graphics that take up less space than bitmaps, and that can be transformed and manipulated with ease. Vector graphics on the Web is not a new idea, and basically most of the GIS vendors have already provided a means of using vector graphics on the Web. However, all these vector supports are based on their own binary file format that require a special plug-in and

are not built upon XML. Not only are plug-ins somewhat of a hassle, but the proprietary binary file format makes it tough to share files between applications and they cannot provide any of the XML benefits.

XML provides a means of defining a Web standard for vector graphics that can be shared between multiple applications and directly inserted into Web pages. XML offers so many benefits as a vector graphics markup language in that no less than three XML vocabularies have been submitted to the W3C to address vector graphics – PGML<sup>14</sup> (Precision Graphics Markup Language), VML<sup>15</sup> (Vector Markup Language) and SVG<sup>16</sup> (Scalable Vector Graphics).

SVG represents a collaborative effort by some of the key players in the computer world – in alphabetical order: Adobe; Apple; Autodesk; Corel; HP; IBM; Inso; Macromedia; Microsoft; Netscape; Quark; RAL; Sun; and Visio [W3C, 1999]. Hence, to find a workable vendor-neutral, cross-platform solution to Web graphics application, every Web browser will eventually get native SVG support<sup>17</sup> and SVG

---

<sup>14</sup> Further information on a note submitted to the W3C on PGML can be found at <http://www.w3.org/Submission/1998/06>

<sup>15</sup> Further information on a note submitted to the W3C on VML can be found at <http://www.w3.org/Submission/1998/08>

<sup>16</sup> As a result of the 2 notes (PGML and VML) submitted to the W3C came the SVG working group, which published a set of requirements for SVG in October 98. The very first draft of SVG then came out in February 99. SVG is currently in its eighth Working Draft as of April 2000. This is unprecedented in the entire history of W3C producing this many versions of a specification in so short a time.

<sup>17</sup> At the time of this writing (early April 2000), native SVG support is unavailable for all Web browsers, but native SVG support is expected

will likely be an important catalyst for a wide range of graphics applications. This collaborative effort is the momentum pushing SVG toward becoming the vector graphics standard for the Internet replacing the previous proposals PGML, VML and WebCGM<sup>18</sup>.

The CSIRO (Commonwealth Scientific Industrial Research Organisation Australia) SVG Viewer page gives a brief and precise summary of SVG:

SVG or Scalable Vector Graphics is a language for describing two-dimensional graphics in XML. SVG is currently being developed as a standard for web-based display of vector data such lines and polygons as well as images and text. Graphical objects can be grouped, styled, transformed and composited into previously rendered objects. The feature set includes nested transformations, clipping paths, alpha masks, filter effects, template objects and extensibility. The Document Object Model (DOM) for SVG, which includes the full XML DOM, allows for straightforward and efficient vector graphics animation via scripting with event handlers for any SVG graphical object. Because of its compatibility and leveraging of other Web standards, features like scripting can be done on SVG elements and other XML elements from different namespaces simultaneously within the same Web page. [CSIRO, 2000]

---

for 6.x releases of Internet Explorer and Netscape Navigator.

<sup>18</sup> WebCGM is a profile of the ISO Computer Graphics Metafile standard (ISO/IEC 8632:1992), tailored to the requirements for scalable 2D vector graphics in electronic documents on the World Wide Web. W3C issued a WebCGM Profile in January 1999 as a Recommendation. The major differences between WebCGM and SVG are: (a) WebCGM is not written in XML syntax; and (b) WebCGM is not a text-based format.



#### 4.2.2.1 More Than Just a Vector Graphics Engine

Being XML-based, SVG will seamlessly integrate with all other emerging XML-based standards such as DOM, CSS, XLink, and so on. That means SVG is not merely a graphics engine but is a fully scriptable, stylable, linkable, interchangeable and Unicode-ready document standard with all the power of XML. To summarize, the key advantages of SVG [W3C, 1996] include: (a) smaller file size; (b) resolution and platform independence; (c) dynamic control and greater interactivity; (d) powerful client-side and server-side graphics authoring capabilities with *JavaScript*; (e) richer graphic design; (f) better printing capabilities; (g) more accurate color profile; (h) searchable text and graphic elements; (i) flexible hyperlinking; and (j) versatile graphics presentation with CSS style.

#### 4.2.2.2 Describing Vector Graphics with SVG for Online GIS

Among those advantages of SVG applicable to all Web, several of them are particularly beneficial to online GIS applications:

- A. **Dynamic control and greater interactivity.** SVG conforms to the DOM, which means every element inside an SVG document is reachable from *JavaScript* and is changeable at runtime in response to mouse events or whatever functions exposed by DOM. Being text-based, content generation is also possible, so that server-side scripts such as CGI can

create custom graphics tailored to the user and situation. All these have far-reaching implications for online GIS applications. For example an engineer of a Water Supply Department is trying to update the information of a valve that has just been replaced. The update is being carried out on-site with a notebook computer connected to the Web via mobile phone. An initial index map pops up when the engineer logs successfully on to his departmental intranet. The engineer would then be able to dive into the map and find out the workplace. When zooming into the area of interest, the engineer turns on to display the facility layers belonging to the Department and the basic mapping layers provided by another department. The valve symbol is highlighted and descriptive information about the valve symbol is popping up when the mouse is putting over it. A popup window asking for action will be displayed when the valve symbol is double clicked. Finally the information is updated.

- B. **Versatile elements grouping/layering.** Because SVG uses XML syntax to describe information in terms of its data type and SVG files are not proprietary binary data files, therefore, developers can easily create scripts that help all stages of data process – data capturing, data editing, data analysis, and data interchange. SVG provides a greater flexibility for organizing data such that graphics elements can be displayed in a variety of ways such as group by group or one layer at a time or all together.

- C. **Flexible graphics display.** Because vector graphics are drawn on demand, at runtime, these can be displayed in any size without quality loss. This quality also means that it is relatively easy to substitute colors, modify line widths, change symbology etc. on-the-fly. SVG also includes the ability to place text along the shape of a path element. It could be a curving path or any path defined by the designer. This type of flexibility goes far beyond what bitmap image can offer. All these make SVG a very flexible and powerful graphics information standard.
- D. **Better element searching.** SVG keeps text in the graphics as text, rather than a “picture of text”. The text remains searchable by search engines even though it is a stroked and filled component of a piece of vector art (e.g. search for a street name on a map, or a landuse type of “Open\_Area” on a landuse layer).

However, no technology comes without at least a few drawbacks and is so true with the new comer SVG. The problem is that SVG is still a vector format designed for the support of line map. Since it is lacking of a high quality image presentation approaching the realism of a photograph, SVG will never be used to store the original satellite imageries. It will only be used to store the extracted data in vector format. Thus a better bitmap format supplementing the vector graphic for representing the real world in online GIS applications is profoundly important. These are what developers and programmers have dreamed of for years.

### **4.2.3 Raster Graphics for the Web: PNG**

For online GIS applications, both vector and raster data have an important role to play. One immediate benefit to online GIS applications is that it will provide unlimited spatial information to the user in a variety of presentation formats. Raster format will be in a better position when such information is in highly color-coded format or in photo-realistic style.

Simplicity, portability and interchangeability are the three major factors to be considered in designing online GIS applications. Therefore, this section will take a look at some fundamental concepts related to raster graphics first, followed by the exciting new raster PNG format and its inclusion in online GIS applications.

#### **4.2.3.1 Choosing the Right Raster Format for Web Applications**

“A picture worth a thousand of words!” Virtually every Web application will involve graphics to supplement the presentation of information and manipulation of data. Therefore, selecting the right raster format for presenting impressive graphics in an efficient manner and manipulating data in a flexible manner are the first and important processes. The following sections describe briefly some fundamental concepts that will enhance the processes.

#### 4.2.3.1.1 Bit depth: grayscale vs. truecolor vs. palette

Raster images are pixel based. The color (bit depth) of each pixel is represented by an integer value of either one-, two-, or three-byte long depending on the representation method used. All color values range from zero (representing black) to most intense at the maximum value. Usually three types of pixel representations are used:

- A. Grayscale pixels are represented by a single value that is a grayscale level, where zero is black and the largest value for the bit depth is white.
- B. Truecolor pixels are represented by three-byte samples (Red Green Blue). The bit depth for each byte specifies the size of each sample while the minimum value for each sample is black. The resultant color for a pixel is the combination of these three sample values and therefore this representation method is capable of displaying  $2^{24}$  or 16 million colors. Since the eye has trouble to distinguish between similar colors, these 16 million colors are often called truecolor.
- C. Palette-based pixels are represented by a single value that is an index of a supplied palette. The bit depth determines the maximum number of palette entries but not the color precision within the palette. If the image has fewer colors than the supplied palette, the image can be rendered exactly. However, when the image contains more colors than the supplied palette,

the image will be rendered with approximate colors available in the palette. The color information of the original image is lost in this situation.

#### 4.2.3.1.2 Compression Algorithm: lossy vs. lossless

Normally raster images are larger than vector images in file size, and the bandwidth in the Internet is limited, resulting in the transfer of a large file across the Internet being unacceptably slow. Therefore, some kind of built-in compression algorithms are often used in the raster image format in order to achieve smaller file size. There are two approaches for compression:

- A. Lossy compression. A lossy compression algorithm means that some image data are lost when the image is compressed by applying the compression algorithm, reducing the quality of the image.
- B. Lossless compression. A lossless compression algorithm means that no image quality is lost when the image is compressed. The compression algorithm simply looks for more efficient ways to represent an image but discards no data information.

#### 4.2.3.1.3 Image format: GIF vs. JPEG vs. PNG

Currently, GIF and JPEG are the formats used for nearly all Web images. The claim from Unisys on GIF as a proprietary standard<sup>19</sup> gave birth to PNG format. PNG is now supported by most of the latest generation browsers and graphics drawing software such as *CorelDraw*.

GIF (Graphics Interchange Format) is a variable 8-bit format that compresses bitmap files into half of their original size, and supports transparency (either transparent or opaque). GIF is a lossless compression format only for images with 256 colors or less. For a true color image GIF may lose most of the color information. GIF is best for creating icons and animated cartoon-like images.

JPEG (Joint Photographic Experts Group) is another raster format developed specifically for photographic images and similar continuous tone images of many colors. JPEG supports millions of colors (24-bit). It is a variable lossy format, which works by analyzing images and discarding those information that the eye is unlikely to notice. The degree of compression is adjustable. JPEG is not a favorable format for the storage of intermediate stages of editing because the associated lossy compression algorithm makes it lose data every time the image is compressed and the quality degradation will be accumulated if it is re-editing, restoring and re-saving

---

<sup>19</sup> The claim arose from the use of LZW as the compression engine in GIF. In 1983, Terry Welch of Sperry (which later merged with Burroughs to form Unisys) developed a fast variant of lossless data-compression algorithm LZ78 called LZW and filed for a patent on LZW.

over the same file [Matthews, 2000]. However, JPEG offers a much more efficient compression algorithm that can produce a smaller file size image with quite minimal loss of quality for photographic images. JPEG has no support at all for transparency – one of the major graphics features on the Web for stacking multiple images.

PNG (Portable Network Graphics) is the most versatile of the current Web graphic formats and it became the W3C's first Recommendation<sup>20</sup> in 1996. PNG is also a lossless format and can compress more efficiently than a GIF or JPEG of the same color depth and quality in most cases, allowing for smaller files that are downloaded more quickly. One of the most commonly heard criticisms of PNG is its lack of support for animation.

#### **4.2.3.2 The PNG Features**

At the outset of the PNG development, it was targeted for a flexible, portable and royalty-free Web raster format replacing the older and simpler GIF format. The following features [Roelofs, 1999] clearly show that PNG is suitable for online applications:

---

<sup>20</sup> The PNG development was not instigated by either CompuServe or the World Wide Web Consortium, nor was it led by them. Individuals from both organizations contributed to the effort, but the PNG development group exists as a separate, Internet-based entity.



- A. **Two-dimensional progressive display.** The 2-D progressive display creates a sense of “fading in” rather than “wiping down” of the image as in 1-D progressive display. In fact, the distinction is purely aesthetic that both approaches will not speed up the downloading of the whole image but will result in a slightly longer loading time than in a similar non-interlaced image. The advantage of interlacing lies in the time required to get an initially low-resolution image onto the screen.
- B. **Multiple layers of transparency.** The variable transparency allows web designers to create special effects that can easily move images from one background to another or different portions of an image to have varying levels of transparency. PNG is using an extra byte to store the additional alpha information instead of storing three bytes for every pixel (red, green, blue). This additional storage approach is to avoid an irretrievable loss of information by pre-multiplying the pixels by the alpha fraction thus increasing the file size of the image.
- C. **Built-in gamma correction.** PNG can store gamma and chromaticity data to improve color matching on heterogeneous platforms.
- D. **Supports three types of pixels: truecolor; grayscale; and palette-based.** PNG supports up to 48-bit truecolor or 16-bit grayscale images, whereas GIF is limited to palette images and JPEG to grayscale and 24-bit true color images.

- E. **File integrity checks.** PNG supports three types of integrity checking to help avoid problems with file transfers: (a) eight-byte signature at the beginning of every PNG image; (b) 32-bit cyclic redundancy check or CRC-32; and (c) Adler-32 checksum.
  
- F. **Metadata for searching and indexing.** PNG files can contain both compressed and uncompressed text chunks for copyright and other info that can be extracted and indexed by Web search tools.

#### **4.2.3.3 Describing Raster Graphics with PNG for Online GIS**

All the above-mentioned PNG features are equally applicable to online GIS applications. As mentioned, JPEG is a lossy format that data degradation is unavoidable and is not the favorable format for handling spatial data that requires accurate storage and transmission of information. Whereas, PNG is a lossless format and is a good choice for those high resolution truecolor images which are unsuited to GIF limited color depths and JPEG lossy compression, such as satellite images. Therefore, for the online GIS applications, it is still worth switching from the traditional JPEG and GIF to a versatile PNG format.

Specifically, for online GIS applications, PNG really has three distinct advantages over JPEG and GIF:

- A. **Integrity.** PNG provides a useful and rigid infrastructure for the storage of intermediate stages of editing – data accuracy is preserved no matter the numbers of saving, editing or restoring are taken on the same file. The three types of file integrity checking provided by PNG will help avoid problems with file transmission. The lossless compression algorithm ensures that results from certain spatial analysis functionality such as raster overlays are free from data error introduced by file manipulation.
- B. **Flexibility.** Unlike JPEG, the variable level of transparency support in PNG is an impressive and convenience feature for the viewing and manipulating of a stack of multiple images. For example, the change patterns can be identified by viewing a stack of two satellite images that are taken on different dates over the same investigation area.
- C. **Interoperability.** Since SVG is also supporting the inclusion of raster graphics, the superimposition of vector and raster graphics for viewing and manipulating is becoming much more easier with a Web browser.

### 4.3 Interactive Problem of Online GIS

The existing and emerging technologies for the development of online GIS applications have been thoroughly investigated in the previous sections and the corresponding implementation issues and problems have also been addressed. As the

Internet and Intranet are becoming the key component for information interchange. The primitive GIS functions provided by the existing Web-enabled GIS applications seem inadequate and users are demanding for more advanced spatial analytical capabilities such as overlays analysis and online map editing. This great demand for advanced GIS functions on the Web is a strong driving force exerted on the developers.

It has been identified in the previous sections that most of the problems related to Web-based GIS applications have occurred because of the use of heterogeneous technologies and spatial data formats. If there is a set of standardized specifications for the multi-facets of online GIS development, most of the existing compatibility problems and implementation issues will be largely resolved.

It is also observed that the infrastructure for building sophisticated online GIS applications have already undergone vigorous evolution. As of this writing, XML and SVG are still relatively new technology that has yet to be adopted by any of the major GIS developers. However, they offer so many benefits that it is probably just a matter of time before GIS designers are modeling their spatial data in XML and performing spatial query and analysis on SVG elements. There is a reason to be so optimistic since XML coupled with SVG and PNG would be the technology that moves traditional GIS applications to a new era of online GIS.

This section will start the discussion on current online GIS development first, followed by a discussion on the future trend of online GIS development.

### 4.3.1 The Current Web-Enabled GIS Packages

The wide acceptance of the World Wide Web occurred in less than a decade. The emergence of such a user-friendly technology has triggered a new wave of development for Web-based GIS over the past few years. In the early stage of Web-enabled GIS (or the first generation of online GIS), nearly all GIS vendors are just moving the host-based architecture in mainframe or minicomputer towards a Client-Server architecture employed by the Web. Since a standardized vector format is not yet established and the raster-based maps are still inadequate to support complex GIS operations, all GIS vendors employ their own proprietary vector formats for the internal vector data interchange. This heterogeneity scenario prohibits data interchanges among multiple systems. Also these initial Web-based GIS applications commonly offer just a limited range of GIS operations.

In order to provide more GIS functions to the Web users, GIS vendors usually employ Java technology for developing the server-side components of Web-enabled GIS software packages. For example, the *ArcIMS version 3* from ESRI is composed of Java and C++ components allowing users to pan, zoom, identify map feature attributes, and link to geographic hot-spots. The *MapXtreme* from MapInfo is also written in 100% Java [MapInfo, 1999].

Another measure that is commonly employed in the online GIS software to improve the performance on geographic data serving is the incorporation of Spatial Engine in the architecture. Spatial Engine is a data blade or data cartridge that can be

directly plugged into relational databases and it provides the following capabilities [Zhuang, 1997]:

- A. Storing geographic data centrally in native relational database tables which can be managed by the Relational Database Management System (RDBMS).
- B. Providing spatial index in the geographic data to improve the access performance.
- C. Generating real time or dynamic topology to support spatial queries in Structured Query Language (SQL) or other high-level languages.

Currently, several Spatial Engine products are available in the market, including *Spatial Database Engine (SDE)* from ESRI, *SpatialWare* from MapInfo Corporation and *Spatial Data Option* from Oracle Corporation. Although each product has its own strengths and limitations, they all follow the three-tier model with the Spatial Engine acting as the middleware to serve geographic data out of a relational database.

#### **4.3.2 The Future Trend of Online GIS: XML; SVG; and PNG**

Official Internet organizations, GIS organizations, GIS vendors and spatial data providers are all working together for a common goal to establish new standardized data formats, protocols and architectures that could facilitate the development of

sophisticated online GIS applications. The Web Mapping Testbed (WMT) project initiated by the Open GIS Consortium (OGC) was the forerunner targeted for this common goal. Specifically, the main objectives of the WMT project are to produce, as quickly as possible, open geo-processing Web technology and to generate the associated OpenGIS specifications for the technology [OGC, 1998]. The GIS community is eagerly awaiting the result of this project and expecting to see a standardized specification for geographical information interchange on the Web which is similar to SDTS but offers much more flexibility.

In the mean time, the Web and GIS communities are anxiously waiting for the release of a SVG- and PNG-enabled Web browser from Microsoft and Netscape. Full compliance with PNG and SVG standards by the major browser vendors is the major hurdle to overcome before widespread use of SVG and PNG images in Web applications can be realized.

The next generation of online GIS application promises to provide developers and users with more interactivity, more functionality, and perhaps best of all, more manageable data structure. The XML and the two new graphics formats – SVG and PNG – are likely to have a significant impact on the future online GIS applications.

A demonstration of web mapping based on SVG and JavaScript was developed to illustrate the potential application of SVG. Detailed descriptions of the demonstration were provided in Appendix F. The technical descriptions of the technology employed were given in Appendix G.

## **4.4 Summary**

In this chapter, the architecture for developing Online GIS application has been discussed and the software developing environment using Java and Active Server Pages have been thoroughly reviewed. Detail review of XML, SVG and PNG and their implications for GIS have also been investigated. A web mapping demonstration site employing SVG, HTML and JavaScript have been developed.



## **CHAPTER 5**

### **FEASIBILITY STUDY FOR SETTING UP ONLINE TEACHING RESOURCES OVER LSGI'S INTRANET**

The discussions on the underlying concepts and technology related to the development of Internet-based GIS have been fully reviewed in previous chapters. The second part of the thesis is to further investigate into which problem domains are appropriate for using Internet technology.

Considering the diversity of technology to be involved for a full implementation of a Web-based approach, the integration of diverse data sources for GIS will be a substantial effort. The main aims of this feasibility study are therefore to find out the key components of a Web-based approach and to test out such components by partial implementation.

The Department of Land Surveying & Geo-Informatics (LSGI) of the Hong Kong Polytechnic University is chosen for the study because of its similarity in working environment with other GIS organizations. Being similar, LSGI holds a lot of spatial data in various formats with different themes. Furthermore, all of these spatial data have been used frequently for teaching and learning purposes. Other than spatial data,

academic staff members have developed and generated a lot of supplementary educational resources to enhance the teaching and learning activities. All these simulate a situation to maximize the benefits of sharing both spatial and non-spatial information. Realizing that sharing/integration of information is an important goal, would an online GIS development be more appropriate for this situation?

## **5.1 Scope**

Specifically, the objectives of setting up online teaching resources over LSGI intranet are:

- A. To provide a convenient way for users in the department – including teaching staff and students – to manage and share electronic educational resources in a multi-user environment over various platforms.
- B. To provide methods or services for online access of spatial data such as the provision of interactive map query and analysis functionality.

## 5.2 Analysis

At the beginning of the feasibility study, interviews with the teaching staff were conducted to identify the problem domains. These interviews also help to devise and formalize specific system requirements.

### 5.2.1 Types of Educational Resources for Sharing

In general, the educational resources to be shared among users in LSGI department can be divided into four types:

- A. **Document type resources in electronic format.** For example, files stored in the computer storage media such as hard-disk.
- B. **Document type resources in non-electronic format.** For example, documents in hard-copy.
- C. **Non-document type resources in electronic format.** For example, numerical data from government departments such as census data and digital map data.
- D. **Non-document type resources in non-electronic format.** For example, a paper map sheet, an aerial photograph, etc.

The first step for any information or document to be shareable and accessible via computer networks is that all the educational resources have to be transformed into electronic format. These can be classified into two types of electronic materials, namely files and data. It is clear that resources of category 2 and 4 are not in electronic format and require transformation.

Document type resources in non-electronic format can be transformed into electronic by using optical character recognition (OCR) scanners or by reproduction using some suitable application software and then save as a file in appropriate format such as spread sheets, plain text, and databases.

Those non-document type resources in non-electronic format will involve a complex transformation that requires data analysis and modeling before any form of presentation. Data analysis refers to what data to be stored whereas data modeling describes the structure and process for storing the data. A much simpler method to convert into electronic format is to scan the information and store the softcopy in image format such as JPEG or PNG.

## **5.2.2 Present Situation and Existing Problems**

### **5.2.2.1 Adaptation to the Current Computing Facilities in LSGI**

There are two types of computing machines currently used in the LSGI department. One is based on *IBM PC* compatible with *Microsoft Windows* operating

systems and the other is based on Sun Sparc workstations with Unix-like operating system. All of these computers are connected to the campus backbone with fast connection speed (100 Base-TX fast Ethernet connection). This would imply that the proposed system should be portable to these platforms in operation and could use the current network infrastructure in communication.

#### **5.2.2.2 File Sharing**

Teaching staff in the department have developed their files of notes, diagrams, presentation, articles and so on, for a given subject of a given programme in a particular year. Both staff members and/or students need to access these resources according to the programme, the subject, and the year responsible for taken in the department. As a result, there exists relationships among the entities of staff members, students, programmes, subjects and academic years. Thus, file sharing leads to the requirement of the storage of relationships with some meaningful entities and these entities form the indices. For example, if one teaching staff wants to share a file related to GIS with all his students and colleagues, the teaching staff has to identify all the entities having relationships with GIS, such as all the subjects related to GIS, all the students who take GIS subjects, and all the colleagues having GIS as research interests.

To allow efficient file sharing, more information such as the relationship between the owner and the files uploaded will be required to associate a given shareable file

with its location. These information, to be supplied by the owners during uploading, will be useful for indexing. The following sections will describe and analyse the three file sharing techniques that are currently used in the Department.

#### 5.2.2.2.1 File sharing problem using FTP server

For sharing files using FTP server, an administrator who is knowledgeable to the shared files is required to design the indices for users to search files effectively and efficiently. However, just depending on an administrator's knowledge to design indices is not a practical solution because only file owners are familiar with their file contents. An administrator, in general, has no knowledge in advance. Thus, the administrator has to ask the file owner about some file information so that a suitable index can be built to facilitate searching and/or subsequent file manipulations such as "update" and "delete" operations. If the amount of the files to be shared is large, the amount of these indices will also be large. Management control turns out to be difficult because in the point of view of the administrator, the indices can be unrelated. The administrator in general has no means to figure out all of these relationships and thus storage redundancy may occur.

Another problem of using FTP server for file sharing is that operations for proper file and index management often require the administrator to work on behalf of the users for each file and index. Users have no way to participate and thus the physical communication overhead is great for file sharing.

Furthermore, the nature of directory structure does not allow a file with multiple indices as well as without file and index duplication. Although shortcuts or symbolic links are possible but the process to manipulate them have to be manual and often requires an administrator's involvement. Using file and index duplication to achieve multiple indices, we have to pay extra storage and overhead in file and index manipulation.

As a summary, using FTP server to share files requires an administrator to execute management policy and to manipulate files and indices. This technique also introduces tremendous communication overhead and creates inflexibility for file owners to upload and update files.

#### 5.2.2.2.2 File sharing problem using file system of LAN server

There are two methods to allow file sharing using a centralized file system of LAN server in a multi-user environment. One is to use publicly accessible directories (both read-enabled and write-enabled) and the other is to assign write-enabled directories to individual users and read-enabled directories to all users. For publicly accessible directories, it will be impossible to enforce security control because everyone has entire read and write access rights. Maintenance on the file system and indices for these publicly accessible directories is hard to achieve.

In addition, each file owner has to either conform to standard directory naming or publicize the shareable directories to other users. For the former practice, the effect of file sharing is the same as that using FTP server. It is less optimal as each user has to index files using the standardized directory naming. For the latter practice, large communication overhead will inevitably be introduced.

In conclusion, the technique of using the file system of LAN server to facilitate file sharing produces with similar problems as the technique using FTP server. It is due to the nature of the directory structure of the file system. Files cannot be multiple indexed without file and index duplication.

#### 5.2.2.2.3 File sharing problem using local file system with network services

Using local file system with network services of LAN server can reduce the storage problem, since the security control of the file system of individual users is done locally by the service of their local operating system. However, this may result in many technical difficulties in terms of mutual communication across various platforms for different machines and operating systems. Consequently a higher level of centralized security control could not be done. Furthermore, all the shared files could not be indexed using multiple indices without file and index duplication.



### 5.2.2.3 GIS Packages

At present, there are several GIS packages installed in the Department for teaching and learning purposes. Basically, they can be generalized into two different categories, namely the Workstation GIS and Desktop GIS. The hardware and software configurations for these two categories are shown in Table 5.1.

Configuration	Category	Workstation GIS	Desktop GIS
Hardware		Sun Sparc Workstation	Intel-based PC
Operating System		<i>Solaris 2.5</i>	<i>MS Windows NT Workstation 4.0</i>
GIS Package		<i>ESRI Arc/Info Ver. 7.0.3</i>	<i>ESRI ArcView 3.0</i> <i>AutoCAD Map 3.0</i> <i>MicroStation Geo-graphics</i>
DBMS		<i>Oracle 7</i>	<i>MS SQL Server 7.0</i>
Number of Installations		5	33

Table 5.1 H/W and S/W configurations of Workstation and Desktop GIS

The Workstation GIS can be considered as a high-end GIS which is mainly used to carry out sophisticated and computing-intensive GIS functions such as overlay and network analysis. These Workstation GIS are fully occupied by senior students and research personnel. On the other hand, the Desktop GIS are heavily used by undergraduates for practical assignments and by teaching staff for demonstration of simple GIS functions such as buffering and spatial query.

The exposure to a diversity of GIS packages is good for students' learning. However, the technical supports for such a diversity of GIS packages are very limited. It is because finding experienced personnel who are competent in both the general operation of the software and programming the proprietary development tools is more difficult than finding a programmer knowledgeable in common programming languages such as Visual Basic or Java.

#### **5.2.2.4 Map Data Sharing**

The Workstation GIS and the Desktop GIS together provide a wide range of GIS functions. However, there are a number of limitations or problems arising from data sharing due to the diversity of the GIS packages used in the Department:

- A. All the installed GIS packages are using proprietary data format for improving storage and better processing performance. It means that the data prepared by these packages cannot be directly and readily used in other competitive packages. Normally, data conversion from one system to another will introduce data compatibility problems.
- B. There is a lack of mechanism to manage and store the information of spatial data (metadata). The information such as the method of acquisition, the scale of the original source, the intended usage, etc., are important and crucial to the success of data sharing. It is dangerous to arrive at a result

from integrating spatial data without knowing the background information of the data. Currently, the spatial data and its associated information are managed by individual staff and the communication between staff on this aspect is weak. Thus the availability of the spatial data are not fully known by other colleagues in the Department.

### **5.3 System Requirements**

After the investigation of the present situations and the existing problems confronting the file and data sharing in the Department, the target system requirements are devised and summarized as follows:

- A. To provide an automatic centralized file management mechanism – this involves a centralized file system to store document type educational resources with user interfaces for file manipulation and with different levels of accessibility according to the predefined privileges assigned by the system administrator.
  
- B. To provide a non-document type information management mechanism – this involves a centralized database to store non-document type educational resources with user interfaces for data manipulation and presentation with different levels of accessibility according to the predefined privileges assigned by the system administrator.

- C. To disseminate spatial data in a non-proprietary format such that commercial Web browsers are able to recognize and manipulate the received data.

To achieve the above mentioned general system requirements, the specific requirement for different aspects, which will be described in the subsequent sections, must be taken into consideration as a whole.

### **5.3.1 User Requirement**

After conducting interviews with the teaching staff in the Department, it was found that all the teaching resources could be shared among two groups of users: staff and student. According to the interview results, the system should provide different levels of access control to distinguish different types of users. Such distinction is found to be reasonable as the access from the students should be carefully controlled in some restricted circumstances.

### **5.3.2 Teaching Resources Management Requirement**

As discussed in sections 5.2.2.2.1 through 5.2.2.2.3, there are pros and cons associated with each file sharing scenario. Basically, a centralized file system could simplify the policy in file input and output and could reduce the implementation

efforts. By extracting the physical storage of the indices and their associated relationships with the files from the directory name outside the file system could provide complicated and flexible file and index manipulation such as multiple indexing. A database system is used to cater for the external storage of indices and their relationships outside the file system. In fact, the extracting process and storing of indices are transparent to users. Thus a proposed system for the teaching resources management is a combination of a centralized file sharing system and a centralized database system.

The advantages in this scenario by integrating a database system with a file system are:

- A. A central file system being embedded into the system for file storage in the server.
- B. A central database management system being embedded into the system for index storage in the server.
- C. File and data being accessed and processed in a multi-user environment.
- D. File and data being stored in a central management environment.
- E. Scalable amount of users using the client to communicate with the server and retrieve the shared files.

- F. Access and security policy could be done on the server and could be monitored and controlled in the server process.

### **5.3.3 Data Requirement**

#### **5.3.3.1 Map Data**

The digital map data provided by the Land Information Centre (LIC) of Lands Department of The Government of the Hong Kong Special Administrative Region is selected as the target data set for developing the prototype. Specifically, several layers of the 1:1000 and 1:20000 basemap digital data are used, such as street boundary features, building block features, survey control points, planning information layers, etc. In addition, an ortho-photograph in TIFF raster format is used. These spatial data formed the basis for spatial queries and simple spatial analysis functions.

#### **5.3.3.2 Non-map Data**

As the target system will be in some form of computer-assisted system, all the pre-existing non-map data such as overhead transparency, hard-copy document etc. must be converted to digital version. PNG is selected to be the storing format for

scanned document/image. These non-map data form the basis for sharing among teaching staff and/or students via departmental Intranet.

#### **5.3.4 Cartographic Requirement**

Cartographic requirements specify what and how features should be presented in a map at different map scales. After several discussions with teaching staff, a number of requirements related to the detailed presentation of map features are addressed and summarized as follows:

- A. Selective display of map features should be provided in the prototype to enable a better presentation of the map on the screen. For example, the building outline can be turned on and off at will by the user with ease.
- B. Display symbology for map features is another key cartographic issue that should be addressed. In the prototype, different sets of symbols will be used to represent features at different scales in order to achieve a better presentation of the map. For example, the railroads are presented in double parallel lines at large scales, but by a single dashed line at small-to-medium scales.
- C. Display descriptive information associated with a map feature as fly-out annotation when the cursor is moved over a map feature would help to optimize the information to be displayed on a limited screen area. For

example, the building name will be displayed when the cursor is moved over the building polygon on the building layer.

- D. The issue of generalization of map features should be addressed to avoid too much details to be displayed on the screen. For example, road centerlines will be shown on the screen for a range of small scales, while road alignments will be shown at large scale. Small map symbols such as gas stations should only be shown at small-to-medium scale.

### **5.3.5 Functional Requirement**

In order to satisfy those three general system requirements as stipulated in Section 5.3 above, the following mechanisms have to be devised to provide functionality for the anticipated system:

- A. Mechanism to input and output the resources.
- B. Mechanism to store and manage the resources.
- C. Mechanism to administer the system and provide security measure.
- D. Mechanism to access the system.
- E. Mechanism to facilitate network communication between users and system.



It is anticipated that users can access both the document and non-document type teaching resources via the prototype system based on the criteria set by the users. For example, users should provide functions to search and download lecture notes related to their studies and allow them to upload completed assignments to appropriate subject lecturer. Appendix A contains a detailed description of functional requirement specifications for sharing document type teaching resources.

In accessing map data, easy-to-use map manipulation tools such as “Zoom In”, “Zoom Out” and “Pan” should be provided. The prototype system should also allow users to look for their required map data easily by providing flexible spatial search. In addition, users should be able to perform some simple spatial query functions such as “Buffering”, “Total distance traveled” and “Current location”. All these anticipated functionality would help to demonstrate the fundamental GIS concepts and techniques to students effectively.

### **5.3.6 Software Development Configuration**

Basically, there is no single tool or software package available in the market that can be used for building such a specific online system. Different software technologies and tools are employed in collaborations for building different components of the prototype. This situation presents a difficulty in developing the prototype system as the integration of different software technologies and tools requires substantial understanding of each of the evolving software tools.

Java is selected as the principle development language for the server and client components in the prototype system. The selection criteria for development tools are:

- A. Ease of installation and upgrade.
- B. Minimization of platform dependency in system operation.
- C. Provide mechanism to facilitate communications across different computer networks.
- D. Favor future enhancements and maintenance.

The detailed software configuration to be adopted for development is listed in Table 5.2.

Operating System	<i>Microsoft Windows NT Server 4.0 Microsoft Windows NT Workstation 4.0 Microsoft Windows 98</i>
Web Server	<i>Microsoft Internet Information Server 4</i>
Web-enabled GIS development tool	<i>Autodesk AutoCAD Map 3.0 Autodesk Map Guide 3.0</i>
Server-side programming language	<i>Sun Microsystems Java Development Kit vl.2</i>
Client-side programming language	<i>Sun Microsystems Java Development Kit vl.2 JavaScript</i>
DBMS	<i>Microsoft SQL Server 7.0</i>

Table 5.2 Software Development Configuration

## **5.4 Summary of Feasibility Study for Setting up Online Teaching Resources over LSGI's Intranet**

In the feasibility study, the scope of setting up online teaching resources over LSGI's intranet have been defined. The various format of educational resources to be shared among users in LSGI department have been identified. The present situation and existing problems in the department have been investigated. The overall system design in terms of user requirement, teaching resources management requirement, data requirement, cartographic requirement, functional requirement, and software development configuration have been devised.

From the study it is found that the educational resources to be shared among users in an academic department could be divided into two main discrete resources, namely, mapping related data and non-mapping part. With such a clear distinctions in the format of the educational resources, it is obvious that the development of a prototype system to fulfil the overall system requirement could also be safely separated into two parallel streams. This separation of development does not mean inconsistency among components as far as the development follows the same specifications and uses the same suite of technology.

Hence, this research project will put more emphasis on the handling of mapping related part with a Web-based approach. The handling of non-mapping part will only be slightly touched on.

## **5.5 Prototype Design**

This section will cover the logical and physical designs of the prototype system. The prototype design will be mainly based on the system requirements as stipulated in section 5.3 above. The specifications for the prototype are established based on the assumption that (a) it is to verify and validate the functionality of the proposed solution as a starting point and (b) it helps refining the system requirements in the future subsequent stages.

### **5.5.1 Overall Design**

The basic architecture of the prototype follows a three-tier client-server model. The block diagram in Figure 5.1 shows the overall system architecture. To simplify implementation, the system is physically divided into two major components: (a) Teaching Resources Sharing Component (TRSC); and (b) Map Data Retrieval Component (MDRC). There is no direct interconnection between these two components in the initial prototype development stage and they are working individually to handle two separate requests from the users – i.e., those requests related to resources sharing and those requests related to a predefined map data set. It is, however, anticipated to enhance the system by combining these two major components into one unified system in the future development.

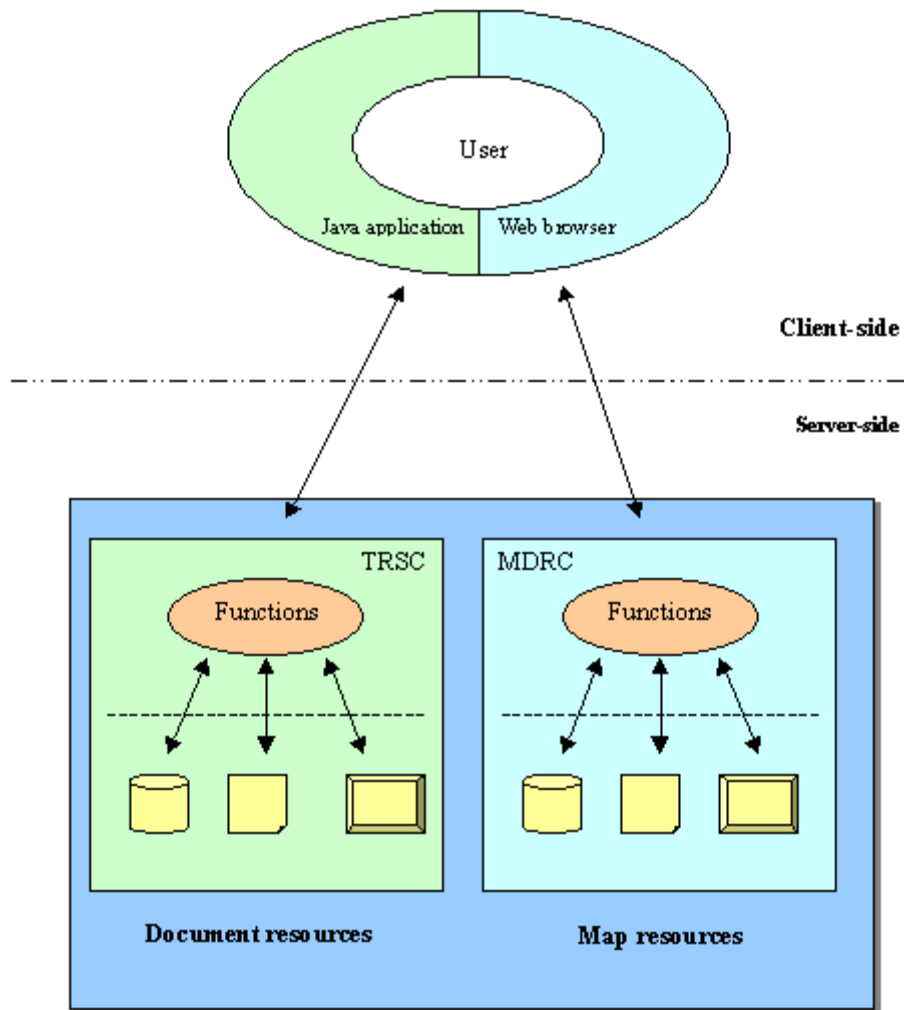


Figure 5.1 Overall prototype system architecture.

The TRSC is divided into 4 modules and they are designed to take up different functions related to database management, file management, policy execution and internal communication, and user interaction. The division of responsibility in this component not only allows each module to specialize on functions of each logical purpose, but also reduce the implementation overhead. Figure A-1 shows the system architecture overview for this component. Detailed descriptions of each module are provided in Appendix A.

The MDRC is basically an *Autodesk MapGuide* application that enable the development of client-side and server-side GIS applications on the Web. Just like any other GIS, *Autodesk MapGuide* combine resource data, such as spatial data (including vector and raster data) and attribute data (SQL databases, dBase databases, etc.) into a map window file (MWF) –Each MWF contains the complete specifications of the map and references to resources on the server. When the user opens a Web page that contains an MWF file or clicks on a link to an MWF file, the Web browser automatically loads the *Autodesk MapGuide Viewer* to display the map.

In the client-tier of the MDRC, a Web browser such as *Internet Explorer* or *Netscape Navigator* running on a PC is responsible for generating requests to the map server and displaying the received data.

The *Autodesk MapGuide Server* is the software running in the server-tier of the MDRC. The *MapGuide Server* interprets the request to determine which data to provide, and then sends the map data according to the specifications made in the request. The *MapGuide Server* consists of three components [Autodesk, 1998]:

- A. **Autodesk MapGuide Server Agent (MapAgent)**. The MapAgent is an interface between the web server and the *MapGuide Server*. It receives requests from *MapGuide Viewer* via the web server and forwards the requests to the Autodesk MapGuide Server Service.
  
- B. **Autodesk MapGuide Server Service and Allaire Cold Fusion Application Server (Server Service)**. The Server Service is a Windows

NT service that interprets and processes the requests for map data distributed by the MapAgent and *Cold Fusion* database application server, formatting the data as requested by the users, and then sending back the data to the *MapGuide Viewer* across the web.

- C. **Autodesk MapGuide Server Admin** (Server Administrator). The Server Administrator provides a set of operational control over *MapGuide Server* such as log file generation, database access control, and data source directories access control.

Figure 5.2 shows the system architecture for this component. A workflow of a single request-and-reply cycle of the MDRC is illustrated in Figure 5.3.

### **5.5.2 Interface Design**

In order to achieve a user-friendly interface that will provide simplified data entry procedures, comprehensible displays, and rapid informative feed-back. The interface design for the prototype system is following as far as possible the following principles [Shneiderman, 1998]:

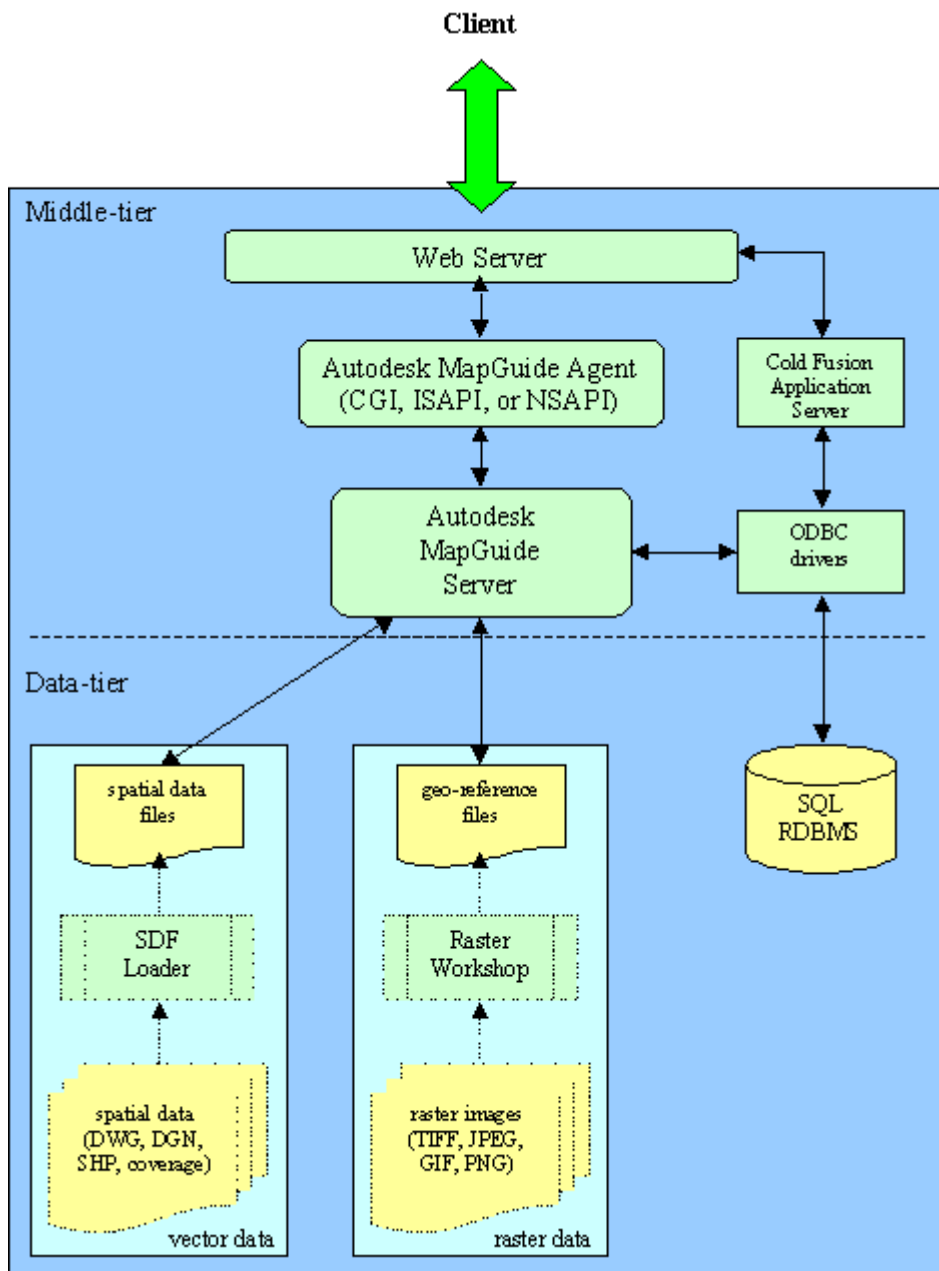


Figure 5.2 Overall architecture of Map Data Retrieval Component.



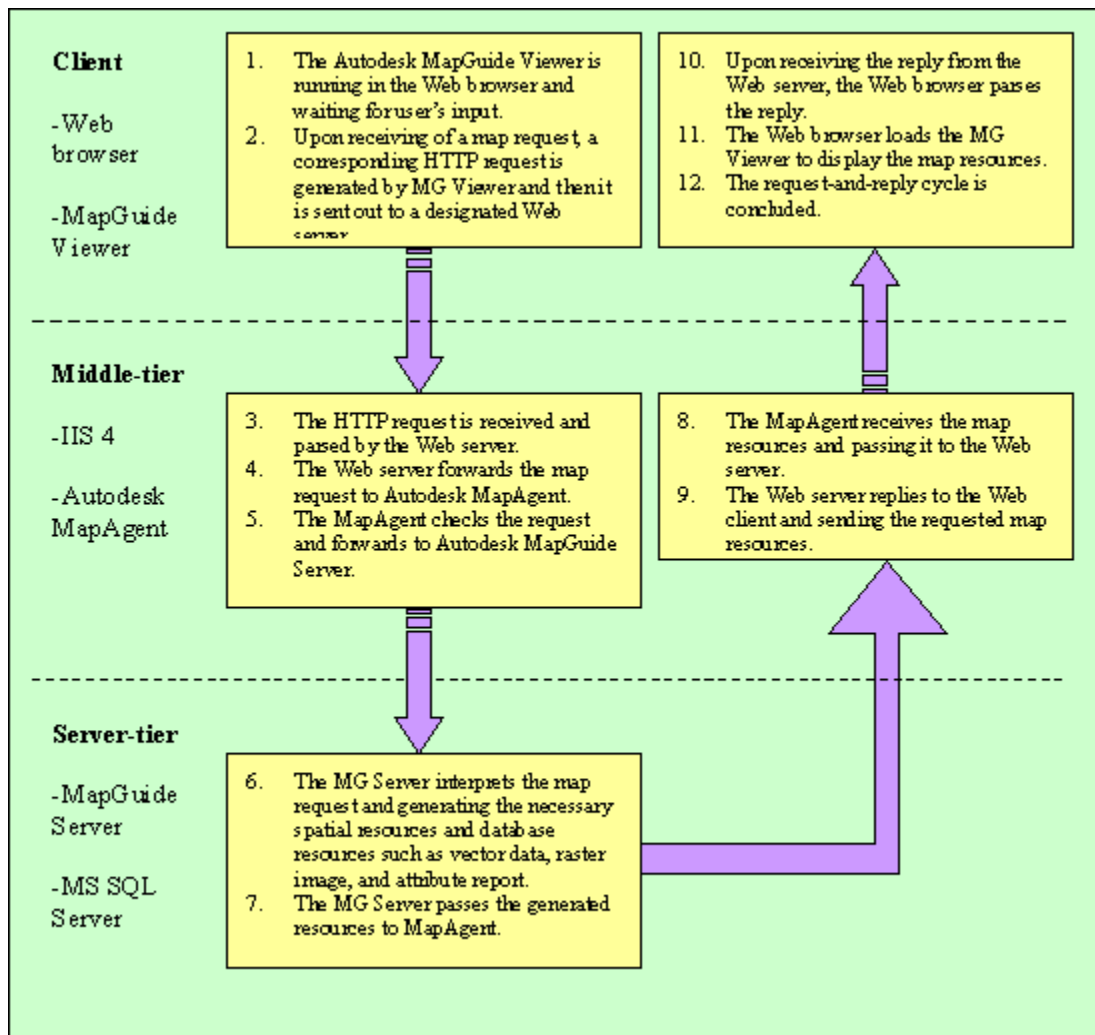


Figure 5.3 Work flow in the Map Data Retrieval Component.

- A. Strive for consistency.
- B. Enable frequent users to use shortcuts.
- C. Offer informative feedback.
- D. Design dialogs to yield closure.
- E. Offer error prevention and simple error handling.

- F. Permit easy reversal of actions.
- G. Support internal locus of control.
- H. Reduce short-term memory load.

### 5.5.2.1 Interface Design for TRSC

For the four modules of the TSRC component described in section 5.5.1 above, only the Client Module requires front-end interfaces to interact with the users. Basically, there are three types of interfaces:

- A. **Login Interface.** This interface allows all the users including the administrator to receive authentication before accessing the TRSC.
- B. **File Sharing Interface.** This interface contains the core functions for teaching resources sharing.
- C. **Administration Interface.** This interface only permits administrator to adjust system parameters during operations.

Detail description for each of these interfaces is included in Appendix A.

### 5.5.2.2 Interface Design for MDRC

The web-based interface in the client-side of the MDRC is developed in JavaScript scripting language. A walkthrough illustrating the functions and interfaces provided by MDRC is included in Appendix E.

### 5.5.3 Exploring Design Choices

It is found that the information related to the Web-enabled GIS packages is quite limited and is generally presented in an abstract way. Therefore, one of the critical decisions is to identify a suitable Web-enabled GIS package which provides the necessary capabilities to develop the required functions of the Map Data Retrieval Component. After investigating the popular Web-enabled GIS packages, a table of comparison is presented in Table 5.3.

Product Specification	Autodesk MapGuide v3	ESRI ArcView IMS v3 (Beta 3)	Intergraph GeoMedia Web Map v3
Architecture	Three-Tier Client-Server	Three-Tier Client-Server	Two-Tier Client-Server
Supported OS (Server-Side)	<i>Windows NT 4.0</i>	<i>Windows NT 4.0</i> <i>Sun Solaris 2.6</i>	<i>Windows NT 4.0</i>
Web Server	<u>NT Platform</u> <ul style="list-style-type: none"> <li>• <i>Microsoft IIS 4</i></li> <li>• <i>Netscape Enterprise Server</i></li> </ul>	<u>NT Platform</u> <ul style="list-style-type: none"> <li>• <i>Microsoft IIS 4</i></li> <li>• <i>Netscape Enterprise Server</i></li> </ul> <u>Solaris Platform</u> <ul style="list-style-type: none"> <li>• <i>Netscape Enterprise Server</i></li> </ul>	<u>NT Platform</u> <ul style="list-style-type: none"> <li>• <i>Microsoft IIS 4</i></li> </ul>

Product	Autodesk	ESRI	Intergraph
Specification	MapGuide v3	ArcView IMS v3 (Beta 3)	GeoMedia Web Map v3
Supported Web Browser	<i>Internet Explorer 4.0+</i> <i>Netscape 4.0+</i>	<i>Internet Explorer 4.0+</i> <i>Netscape 4.0+</i>	<i>Internet Explorer 4.0+</i> <i>Netscape 4.0+</i>
Development Tool	<i>Java</i> <i>JavaScript</i> <i>VBScript</i> <i>Active Server Pages</i> <i>Cold Fusion</i>	<i>ArcIMS SDK</i> <i>Java</i> Servlet Engine (Java 2) <i>Active Server Pages</i> <i>Cold Fusion</i> <i>JavaScript</i> <i>VBScript</i>	<i>Java</i> <i>JavaScript</i> <i>VBScript</i> <i>Active Server Pages</i>
Map Sharing Format	<u>Raster Format</u> PNG, GIF, JPEG and TIFF, GeoTIFF  <u>Vector Format</u> Spatial Data File (A proprietary file format for transferring across the Web)	<u>Raster Format</u> PNG, GIF, JPEG and TIFF	<u>Raster Format</u> JPEG  <u>Vector Format</u> ActiveCGM
Functionality	<ul style="list-style-type: none"> <li>• Display cursor real world coordinates</li> <li>• Display viewing window scale</li> <li>• Display run-length distance</li> <li>• zone buffering</li> </ul>	<ul style="list-style-type: none"> <li>• Spatial analysis such as buffering and overlay query are available in the Java viewer solution.</li> </ul>	<ul style="list-style-type: none"> <li>• Spatial analysis capabilities are provided by an add-on component called GeoMedia Web Enterprise.</li> </ul>
Advantage	<ul style="list-style-type: none"> <li>• Maintain the same map data visual quality in the process of zoom in/out</li> </ul>	<ul style="list-style-type: none"> <li>• Plug-in is not required</li> </ul>	<ul style="list-style-type: none"> <li>• Maintain the same map data visual quality in the process of zoom in/out</li> </ul>
Disadvantage	<ul style="list-style-type: none"> <li>• The plug-in <i>MapGuide Viewer</i> is required in the client browser</li> </ul>	<ul style="list-style-type: none"> <li>• Cannot maintain the same map visual quality in the process of zoom in/out because of the inherited raster data transferring format.</li> </ul>	<ul style="list-style-type: none"> <li>• The plug-in <i>ActiveCGM</i> is required in the client browser</li> </ul>

Table 5.3 Comparison among different Web-enabled GIS packages.

The selection of a Web-enabled GIS package for the development of the Map Data Retrieval Component is therefore based on the following two categories of criteria.

I. From the user point of view, the system design should focus on:

- A. Reduction of workload of an administrator in file sharing and ease of management of shared files.
- B. Efficiency of file access among different users over the Internet.
- C. Flexibility of users to index their shared files, and create logical relationships associated to these files.
- D. Ease of installation of the client program, and standardization of various operating platforms.
- E. Maintain a reasonable cartographic quality for zooming the map data.

II. From the implementation point of view, the system design should focus on:

- A. Ease of implementation.
- B. Use of networks, and the ability to communicate with multiple users in a distributive environment.
- C. Independence of operating platforms.

D. Independence of use of databases of different vendors.

E. Integration of development tools to facilitate the necessary functions.

*Autodesk MapGuide v3* suite is selected as the back-end software for the development of the MDRC after two rounds of selection process. In the first round, *Intergraph GeoMedia Web Map v3* is excluded because of its limitation in support of different raster formats such as GIF, PNG and TIFF. In the second round, *ESRI ArcView IMS v3* is not selected mainly because of its inflexible support of map data transmission format – only supports raster format.

However, the adoption of *Autodesk MapGuide* suite for the development of the MDRC imposed a couple of minor technical problems: (a) a conversion of existing commonly used vector format to the proprietary SDF vector file format is required; (b) an understanding of GeoTIFF raster format is needed in order to correlate the geo-referenced raster images with the SDF vector file; and (c) installation of the plug-in *MapGuide Viewer* is required in the client browser.

## **CHAPTER 6**

### **RESULTS**

The system design covered in Chapter 5 was trying to identify and investigate the feasibility of setting up a system to facilitate online sharing of teaching resources for the Department of LSGI from a conceptual point of view. It was understandable that some of the technical difficulties and constraints could only be identified during the implementation stages. Problems arising from the prototype implementation could be screened and categorized into two areas. Those problems arising due to the inadequacy of the existing software and hardware could be resolved by work-around methods or upgrading the hardware. However those problems due to the limitations or pre-maturity of the employed software technology could not be easily overcome for the time being.

#### **6.1 Implementation of Prototype**

The prototype system design was the most intensive work phase of the project. Implementation was relatively easy and straight forward. The implementation of the

system followed two paths of activities to meet the system requirements as stated in Section 5.3 and the prototype design specifications as stated in Section 5.5.

### **6.1.1 Implementation of MDRC**

As stated in Section 5.5.3, all currently available web-enabled GIS packages did not support SVG as the transferring format for vector and did not recognize XML as a meta-language for the web page. Also, most of the currently available web browsers (e.g. *Internet Explorer version 5.0* or earlier, *Nescape Navigator version 4.6* or earlier) neither support SVG nor recognize XML fully. Therefore, after much deliberation, it was desirable to follow a non-XML based GIS package for the implementation of MDRC within a tight schedule and *Autodesk MapGuide* software package was selected as the developing basis of MDRC. The implementation stages of MDRC were taken as follows:

- A. Intensive study of *Autodesk MapGuide* software package.
- B. Web site architecture design and development.
- C. Web page design and development.
- D. Spatial and attribute information collection and transformation.



### 6.1.2 Implementation of TRSC

For implementing TRSC, the work was scheduled into five phases using the modular implementation approach.

- A. **Derivation of mechanisms to satisfy the functional requirements of TRSC.** Detailed discussions on these mechanisms were included in Appendix B.
- B. **Design of database.** The database to be used was *Microsoft SQL Server 7.0*. In this implementation phase, a database with defined schemas, basic data, basic user information, and basic relationships between files and tables were devised. Appendix C gave a detailed description for the schemas and the relationships between tables.
- C. **Derivation of index file format for file transmission.** The detailed description for the index file format was included in Appendix D.
- D. **Development of Java-based FTP client and FTP server.** The development of this Java-based module was desirable to facilitate the in-house requirements for resources sharing. Furthermore, it provided a framework that was capable of transferring embedded indices in conjunction with the normal file transmission using standard TCP/IP protocol. The rationales and detailed description for such development were included in Section B.7 in Appendix B.

E. **Development of graphical interfaces.** The description and specifications for the four modules of TRSC were contained in sections A.11 through A.14 in Appendix A.

## **6.2 Discussion of Results**

### **6.2.1 Results Achieved**

#### **6.2.1.1 Results Achieved in Developing MDRC**

The MDRC implemented in this project materialized a basic set of spatial query functionality on the Web. Users could make use of their free of charge Web browser to navigate the Web sites containing online map data and perform the associated basic GIS functions. The walkthrough demonstrated in Appendix E clearly showed that the Web was an ideal medium for providing wider accessibility of geographical information to a larger number of users through a corporate intranet or the Internet cost-effectively. The findings obtained during the development and testing of MDRC gave a clear and positive picture that the future development of GIS should be heading towards the Web-based solution. The significant results achieved in developing MDRC to help improve map data sharing and integration were discussed specifically in the following sub-sections.

#### 6.2.1.1.1 Integration of different sources of map data

The map data in the MDRC were coming from different sources with different map data accuracy. For example, the coast lines of Hong Kong were from the Land Information Centre of the Lands Department of Hong Kong Special Administrative Region of The People's Republic of China, the Tertiary Planning Unit (TPU) boundaries were from the Planning Department of Hong Kong SAR, the digital ortho-photograph was from Photogrammetric Unit of the Lands Department.

The MDRC provided facility to import map data and present the imported data to users in a consistent environment (Web page). Figure 6.1 showed that a variety of map data were provided to the users. Figures 6.2 and 6.3 illustrated that raster and vector map data could be displayed simultaneously.

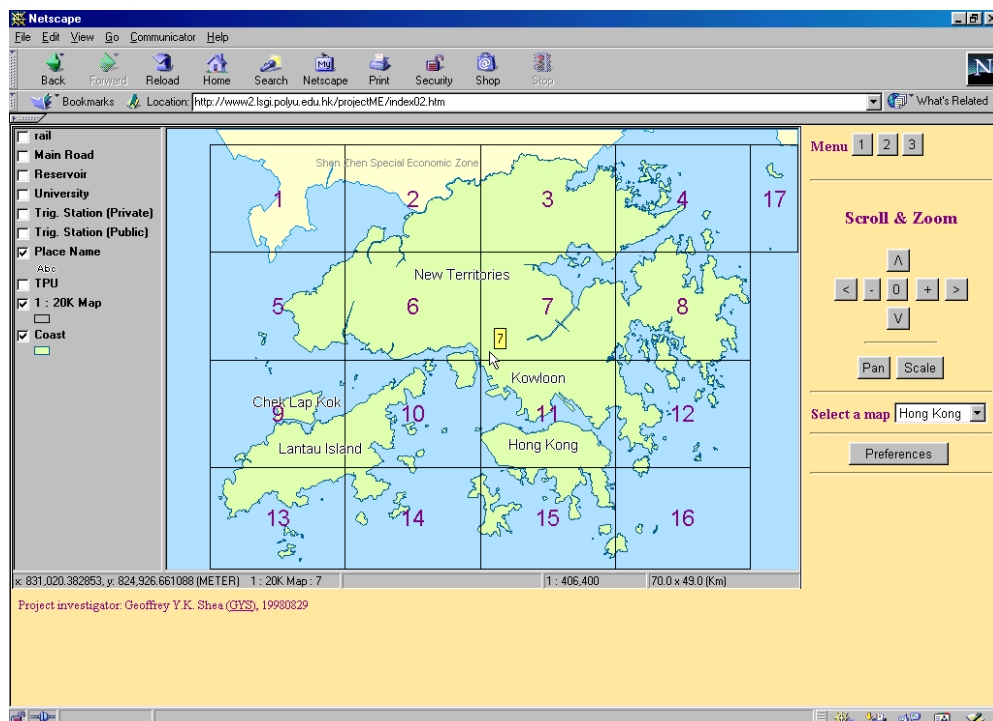


Figure 6.1 Main page of MDRC.



Figure 6.2 Simultaneous display of raster and vector map.

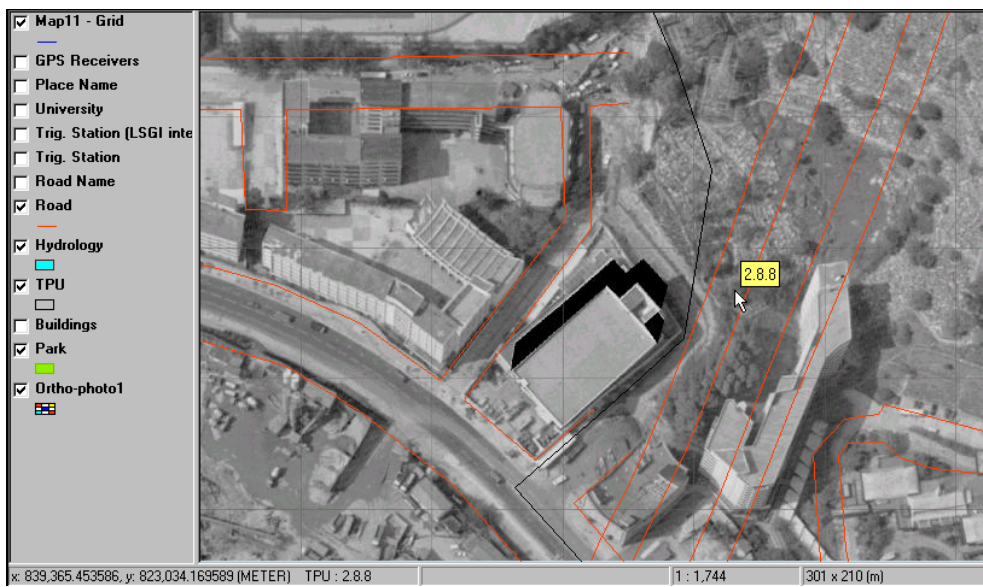


Figure 6.3 A closer look on the raster and vector map.

#### 6.2.1.1.2 Implementation of security on map data

Since different sources of data might have different access permissions to different groups or users in reality, MDRC was designed to provide security measures on map layers(data) limiting data access for selected groups or users.

Figure 6.4 was an example showing that the access to “Trig. station (Private)” layer was allowed only if the correct username and password were entered.

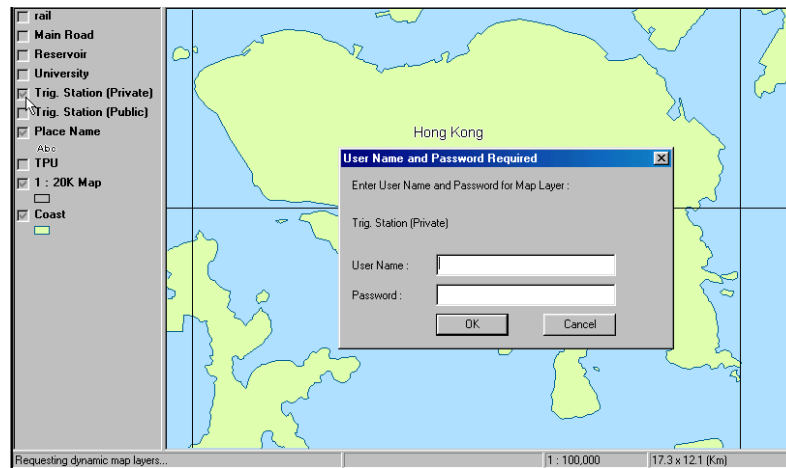


Figure 6.4 Logon window for username/password protected layer.

#### 6.2.1.1.3 Selective(thematic) display of map data

All available map data were grouped into map layers on the MDRC. The system provided a flexible means for users to select layer(s) to be displayed/turned-off on the displaying window by checking/un-checking the check box next to the displayable map layer name. Figure 6.5 illustrated that 3 map layers (Place Name, 1:20K Map and Coast) were being displayed on the main window.



Figure 6.5 Using legend to display map layers selectively.

#### 6.2.1.1.4 Multiple display symbology with the same map data set

The MDRC possessed the capability to display the same set of map data using different symbology at different scales. Figures 6.6 and 6.7 demonstrated that two line styles were used to represent the same data set at scales 1:20000 and 1:10000 respectively.

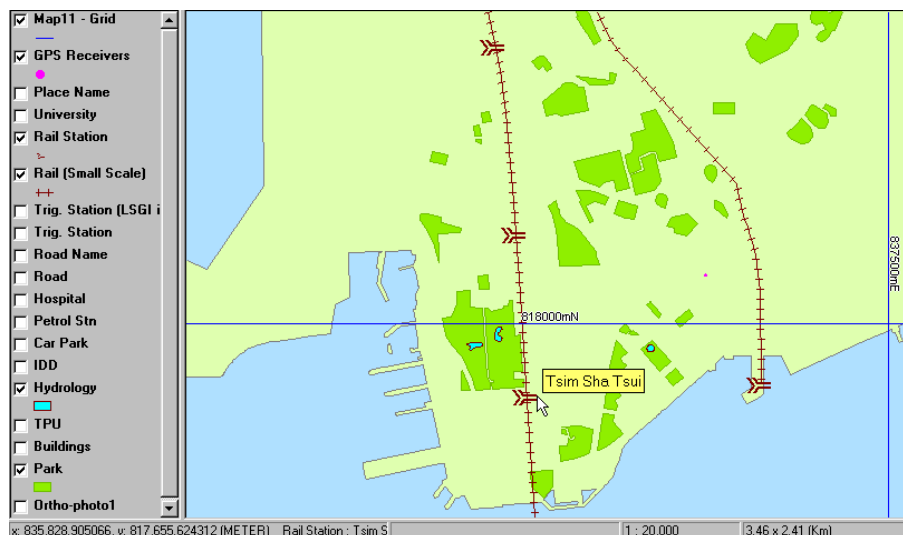


Figure 6.6 Line style used for Rail layer at 1:20000.



Figure 6.7 Line style used for Rail layer at 1:10000.

#### 6.2.1.1.5 Basic GIS queries on map data

There were 3 basic map query functions provided in the MDRC: (a) measuring distance on the map; (b) selecting map objects in 3 different ways – using the mouse, by defining a radius and by defining a closed polygon; and (c) generating reports on selected objects. Figure 6.8 showed an example of measuring distance. Figure 6.9 illustrated the map objects selection by polygon and Figure 6.10 displayed a report on the selected map objects.

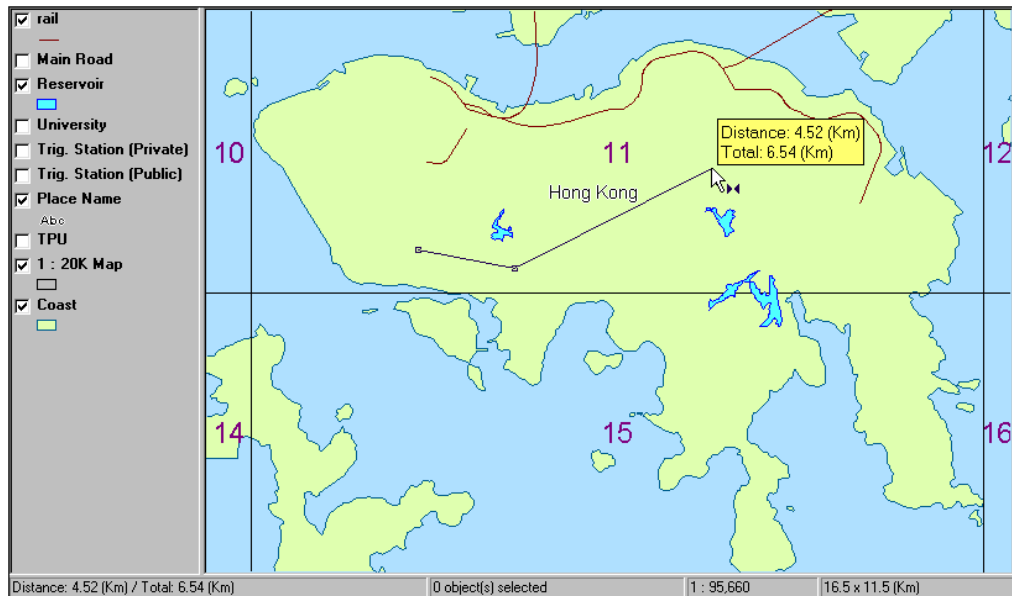


Figure 6.8 A measuring distance example.

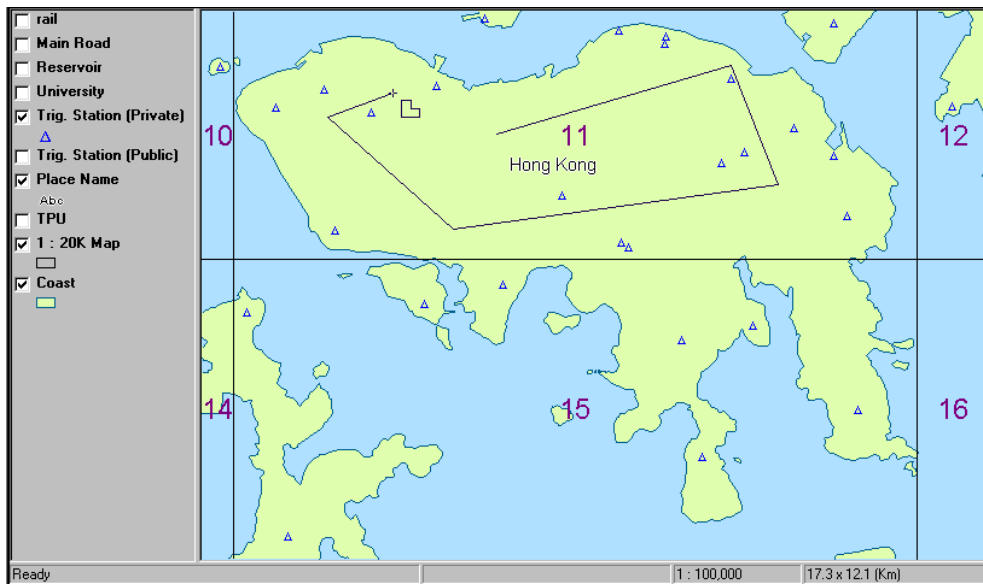


Figure 6.9 Map objects selection by polygon.



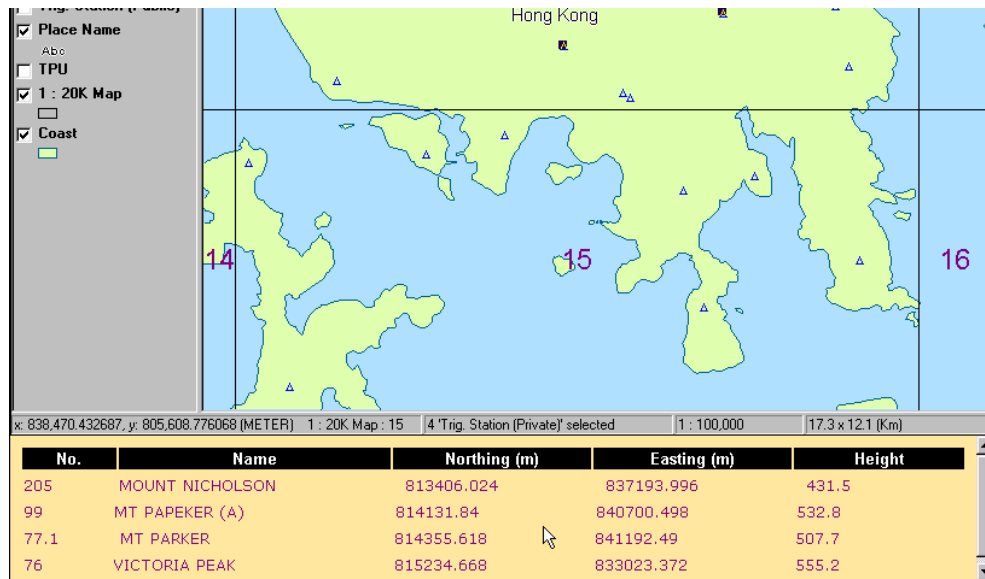


Figure 6.10 Report on the selected map objects.

#### 6.2.1.1.6 Basic GIS analysis function

Other than a report that could be generated from the selected map objects, zone buffering could also be performed on the selected map objects. Zone buffering was applicable to either point objects, line objects or areal objects. A variety of choices were provided for users to define the buffering properties such as hatch patten, color, line style etc. Figure 6.11 showed the available buffering properties and Figure 6.12 was an example of the buffering result.

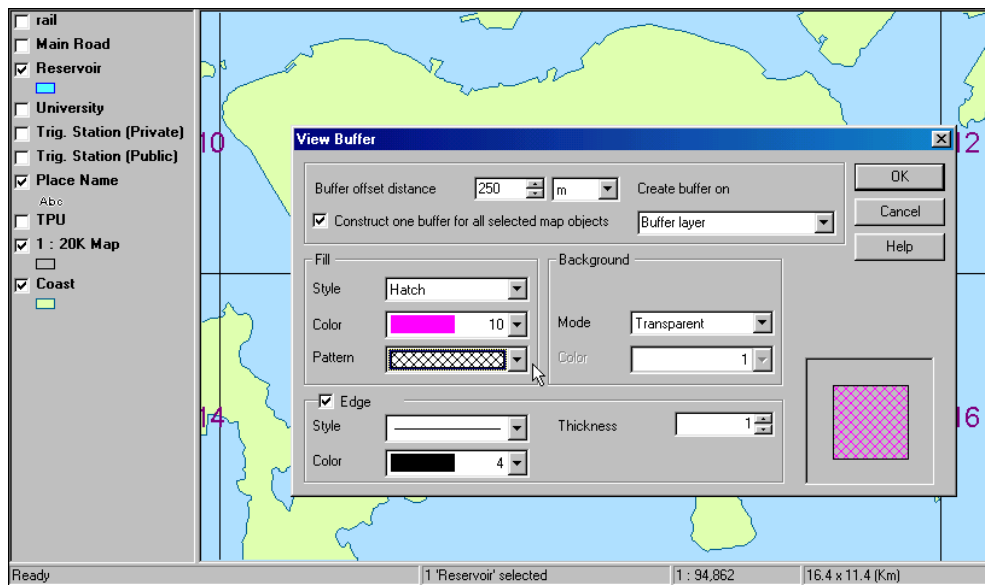


Figure 6.11 Zone buffering properties.

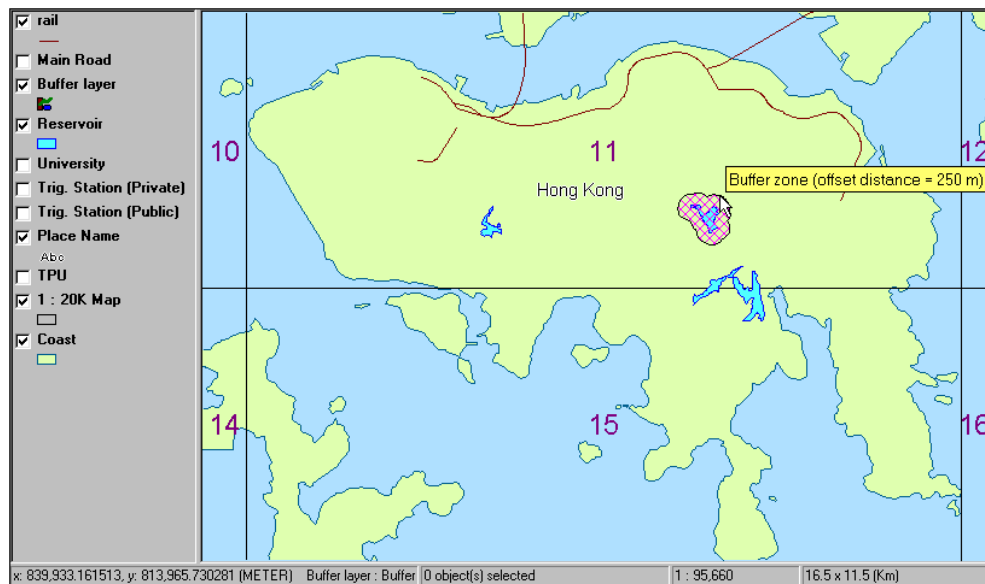


Figure 6.12 Result of zone buffering.

### 6.2.1.2 Results Achieved in Developing TRSC

The results achieved in the TRSC were encouraging and would benefit from future enhancement. The significant results achieved were:

- A. A detailed functional requirement specification for TRSC was devised and mechanisms to facilitate the TRSC functions were developed and tested. Basically there were 6 mechanisms developed employing the Client-Server computing model: (a) mechanism to store the resources; (b) mechanism to input and output the resources; (c) mechanism to control the system and provide security; (d) mechanism to access the system; (e) mechanism to manage the resources; and (f) mechanism to facilitate network communications between users and the system. These six mechanisms were presented to the users (including system administrator) through a set of graphical user interfaces. The main GUI of the client module was shown in Figure 6.13. The full description for the functional requirement specifications for TRSC was given in Appendix A and the mechanisms to facilitate the TRSC functions were described in Appendix B.
  
- B. The master database design was tested to be in Third Normal Form. The normalized database was tested and it could be used to obtain information of master records with joins. The database design and schemas for TRSC were illustrated in Appendix C.

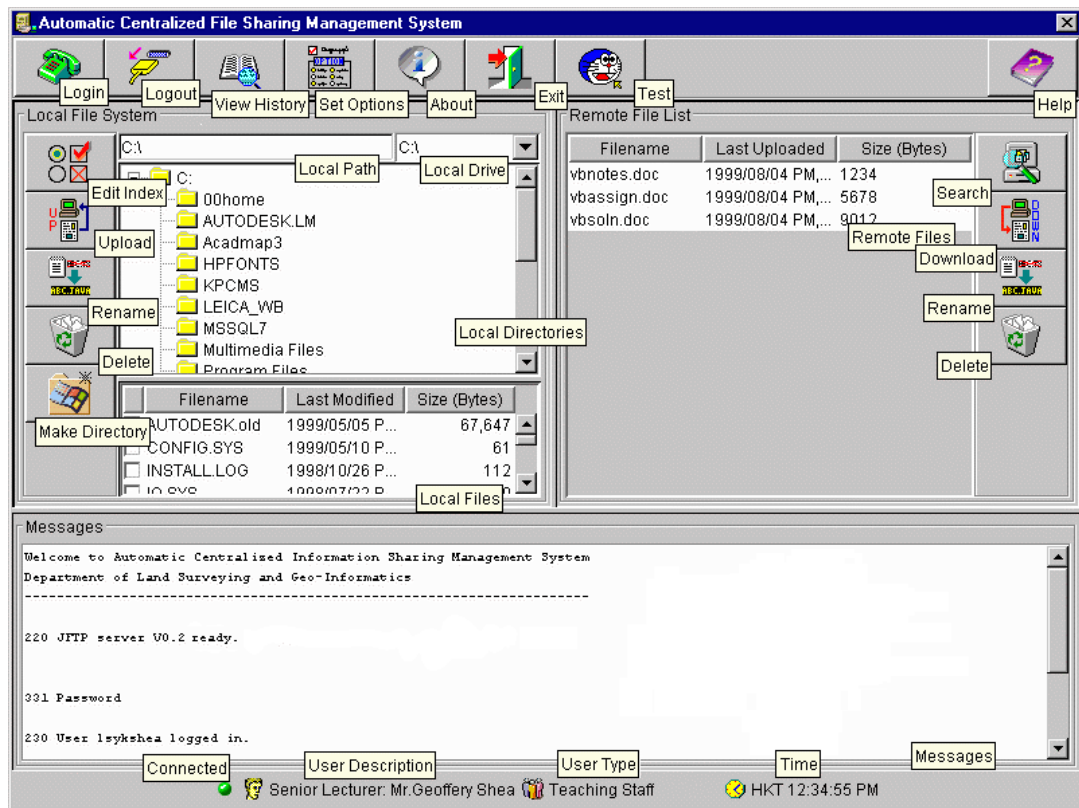


Figure 6.13 Prototype graphical user interface of Client Module

C. The basic communications employed in the sixth mechanism (mechanism to facilitate network communications between users and system) were complied with an international standard protocol called File Transfer Protocol<sup>21</sup>. In this application, in addition to file transfer, additional information are required during uploading and searching. These additional information were defined as *indices* and stored in *index files*. These *index files* had to be transferred to the server side together with each file manipulation operation. As a result, a new transfer protocol was devised.

<sup>21</sup> The official specifications are listed in memo "Request For Comments : 959" in October 1985 by J. Postel and J. Reynolds. The specifications of RFC959 can be found at <http://sunsite.auc.dk/RFC/rfc/rfc959.html>.

The new protocol was based on File Transfer Protocol that was specified in RFC959 with some modifications. The modification consisted of two parts: (a) introduction of new commands – LOGIN, UPLD, SRCH; and (b) termination of some of the original commands – CWD, LIST, RMD, MKD, PWD. A detailed description of the modification was given in Section B.7 of Appendix B.

- D. A Java FTP client was developed. Testing indicated that this Java FTP client could successfully to communicate with other FTP servers using the basic user commands specified in RFC 959.
- E. A Java FTP server was developed and passed the basic testing to communicate with other FTP client using the basic user commands specified in RFC 959.
- F. Both the Java FTP client and the Java FTP server could communicate with each other and could facilitate file transfer using the basic user commands specified in RFC 959.
- G. One of the four index files – *profile for login* – as stated in Appendix D was implemented and tested to be correct. The Server Interface Module generated a *profile for login* index file for a given user from the information stored in the master database and the Client Interface Module obtained such index file – *profile for login* – from the Server Interface

Module, parsed the profile correctly, and displayed the appropriate user information on the interface of the Client Interface Module.

### **6.2.2 Difficulties Encountered**

There were a number of technical difficulties that could not be discovered until the actual implementations were carried out. As development tools – to name a few such as Java, *JavaScript*, *Microsoft SQL Server*, *Autodesk MapGuide* software package, networking – were so advanced, substantial time and efforts were involved in familiarization and testing.

Nevertheless, there were two major issues encountered during implementation that were worth mentioning in the following sections.

#### **6.2.2.1 Failure Implementation of Overlay Analysis**

During the implementation stages of MDRC, it was found that some of the main functionality such as network and overlay analysis commonly available in GIS were not available with the *Autodesk MapGuide* software package. Therefore, finding a feasible solution to implement an overlay analysis in the MDRC was added.

For a server-side overlay analysis application, two stages were involved. First, the server responded to the client's request by invoking the overlay analysis icon.

Second, the server passed back the overlay analysis result to the client. The total time required to complete an overlay analysis was roughly equal to the time required to complete the analysis on the server plus the time required to transmit the result from the sever to the client. Therefore, it was technically possible to implement a server-side overlay analysis application. However, this application was not appropriate for processing on the server side for the following reasons:

- A. A client who submitted a request for such a sophisticated spatial analysis had to wait for a very long time before the result was received by the client. A long idling time for a request on the Internet was obviously unacceptable from the user point of view. This long waiting time was purely deduced from the working experience of overlay analysis performed on a standalone environment.
- B. The long waiting period created by the overlay analysis on the server would subsequently create a long waiting queue in the server. The server might crash or performance was degraded to an unacceptable level when a large number of such requests coming in all at the same time, resulting in the waiting queue buffer of the server was being full.

Alternatively, the implementation of overlay analysis on the client-side had been considered. Theoretically, all GIS operations could be performed by the client in a client-side environment provided that the client's hardware was powerful enough to perform this kind of sophisticated spatial analysis. In fact, most of the infrequent

general users could not fulfil this high-end hardware requirement. This client-side application also involved a large amount of customization effort.

After much deliberation, the implementation of overlay analysis function in the MDRC was dropped.

#### **6.2.2.2 Successful Implementation of Modified FTP Protocol**

In the beginning of the actual implementation of the modified FTP protocol, the *indices* were designed to be interchanged between the client and the server by the use of message transfer technique used in TCP/IP model. However, due to my not-so-advanced Java programming skill, the Java program so formed was too buggy and difficult to maintain.

After some investigations, it was found that the Java package Remote Method Invocation (RMI) could provide some sort of solution for information transfer when the indices were implemented as some known objects in the server. However, the system dependency on the services and the policy control of a Web server violated the central control in file sharing. Also, the service dependency could be difficult to maintain as well. Therefore, this solution was not pursued.

Finally, the design was settled by simply putting all the relevant indices in an ordinary flat file with no more than a hundred text lines. The transferral of the index



files was embedded with every file uploading and downloading. Both the communication and management problems were solved with this approach.

Since the successful development of this protocol was the key component of the TRSC, quite a long time had been spent in this stage. Now each file on the file server was associated with a particular index file. With the implementation of such index file, a file owner could add, edit, delete and update the content of these indices according to some policies. While a file consumer could only search and download for a pre-defined group of files and location. These file and index manipulations were dispatched as requests by user interfaces to a server via FTP protocol (refers to Appendix A.16 through A.18 for a detailed description of the client module interface design).

## **6.3 Anticipated Future Enhancement**

### **6.3.1 MDRC Enhancement**

Having gone through the implementation stages of the MDRC, a questionnaire was conducted to find out the feedback from the users. A group of 24 users comprising 4 teaching staff and 20 students were selected. The feedback from the selected group of users revealed that several issues were critically affecting the usefulness and effectiveness of the MDRC component in serving as a map facilitator. The concerned issues were:

- A. Adding online spatial data editing and updating functions.
- B. Relieving the dependence of proprietary vector transferring format on the Web and changing to a more neutral format such as SVG.
- C. Providing online function for merging/integrating spatial data with different formats or from different vendors.
- D. Offering more sophisticated GIS analysis functions than the ordinary query functions, e.g. network analysis and overlay analysis functionality.

These requests were quite obvious and natural from the user point of view and they were not too difficult to achieve in a standalone or single user working environment where users have full access to their data. However, the MDRC was mainly built around a commercial map handling machine from Autodesk (*Autodesk MapGuide*) that directly governs the system architecture design of MDRC. *MapGuide* did require a lot of pre-processing of spatial data before the maps were published online. Also *MapGuide* was unable, as of this writing, to offer functionality related to online spatial editing and sophisticated GIS analysis functions. Therefore, it seemed that the only way to satisfy the request for more and better functionality during MDRC enhancement phase was to start over the design of MDRC by adapting the same approach used by TRSC. That means the redesigned MDRC should be free from the dependence of any commercial map engine in providing online GIS functionality. Furthermore, the use of Java technology for TRSC implementation demonstrated that the application developed by Java was

much more flexible in system design and implementation stages. In summary, the MDRC enhancement had to start over using modular approach in system design to ensure the spatial resources were handled properly with the new Internet technology such as XML, SVG and PNG.

A group of inner-Sydney local councils, known as the Inner Metropolitan Regional Organisation of Councils (IMROC), through its “Sydney’s Information Highway” project exhibited a couple of advantages in the implementation of an open-sourced Web-based central infrastructure for information sharing: (a) provides greater flexibility and extensibility in database management and applications development/modification; (b) maintains autonomy in data contribution and ownership; (c) accesses information seamlessly; and (d) eliminates information duplication and inconsistency. The successful employment of SVG and XML in the “Sydney’s Information Highway” project shed light on the future enhancement of MDRC.

### **6.3.2 TRSC Enhancement**

The current prototype of TRSC was focused on the implementation of a dedicated infrastructure for automatic central file management and efficient file access. It did not include the specific application on processing the shared materials, such as implementing a browser to show thumbnails of graphic files and to search and browse a map file before downloading/uploading. However, the working ideas and

the infrastructure developed so far in this prototype could provide flexibility and extensibility for further enhancement.

In the current design of the TRSC, there were four index files to be used. Since the content of these index files were not the same, at least 4 logical functions were required to parse the files. As stated in Section 6.2.1.2 above, only the *profile for login* had been implemented and tested to be correct, the remaining three logical parsers had to be developed to parse the other three index files based on their clear definition as stated in Appendix D.

Currently, the TRSC was designed and implemented to be a standalone Java application. System security and performance were the main reasons for implementing the prototype as a standalone application. Because the findings from the prototype indicated that full development of TRSC was feasible, it should plan to modify the TRSC into an Java applet so that users could access the teaching resources with a Web browser connected either within a local area network or via the Internet with a modem outside the campus.

### **6.3.3 Integrated Information Sharing System**

Having demonstrated the solution for file sharing and map data retrieval in two separate procedures, it was anticipated to integrate the file sharing and map information sharing within one unified system. Then an automatic and transparent mechanism in the user interface was provided to select the set of functions for file sharing and the set of functions for map information sharing.

## **CHAPTER 7**

### **CONCLUSIONS AND RECOMMENDATIONS**

#### **7.1 Conclusions**

##### **7.1.1 Objectives and Strategy**

Integration of diverse data sources for GIS was an important process in providing global accessibility for spatial data and GIS applications. With the widespread use of digital data for all walks of life in navigation and spatial decision making, the Web technology had become increasingly important. However, in spite of international efforts for decades, there were still many fundamental problems to be solved at different levels. A set of emerging Internet technology was flourishing which could provide a totally different framework of ideas for data integration. Therefore, this had been investigated as the main objective of this study.

To achieve this objective, a “divide and conquer” strategy had been employed in the study. The main objective was sub-divided into five sub-objectives: (a) determining the mapping requirements of graphic and non-graphic spatial data for online GIS application; (b) analyzing the requirements of spatial data integration for

online environment; (c) investigating a suitable method for integrating different formats of spatial data; (d) studying the feasibility and applicability of setting up the online GIS; and (e) developing a prototype for online sharing of teaching resources over an academic departmental intranet.

### **7.1.2 Mapping Requirements of Graphic and Non-graphic Spatial Data for Online GIS Application**

The history and fundamental of data integration were thoroughly investigated as described in Section 2.1. Three technical difficulties (detailed in Section 2.2) in the formulating of data integration strategy were identified: (a) proprietary data formats; (b) lack of data information; and (c) data transmission restriction. These technical difficulties could be solved by employing a proposed Three-Tier Client-Server framework as described in Sections 2.5 and 2.6. The reasons were readily revealed in Chapter 4.

### **7.1.3 Requirements of Spatial Data Integration for Online Environments**

The general requirements of spatial data integration for online environments were investigated (detailed in Chapters 3 and 4). The three-Tier Client-Server computing model was found to be the most efficient model for online GIS applications. Conceptually and theoretically, this model improved the overall performance of a

Web application in system throughput, reliability and functionality (Section 3.4.2 refers).

The critical examination of the emerging Internet technology on XML, SVG, PNG and Java had been carried out as discussed in Section 4.2. It concluded that online GIS applications based upon XML, SVG, PNG and Java would provide developers and users with more interactivity, more functionality and more manageable data structure.

#### **7.1.4 Suitable Method for Integrating Different Formats of Spatial Data**

A conceptual framework for integration of diverse data sources was suggested in Sections 2.4, 2.5 and 2.6. The framework provided an environment that would be independent of the underlying GIS data structure. The proposed framework was based on the Three-Tier Client-Server computing model, which included four loosely coupled modules (Application Interface, Presentation, Integrator, and Data) as shown in Figure 2.1.

A refined conceptual framework based on the emerging Internet technology on XML, SVG, PNG and Java was proposed (detailed in Table 2.1 and Figure 2.3). There were five reasons to support this proposed framework as stated in Section 2.5. The rationale behind the suggestions were fully discussed in Sections 4.2 and 4.3.



In addition, a generalized approach is suggested (detailed in Section 4.2.1.4) to implementing online GIS applications within XML framework. This included analyzing: (a) the information requirements; (b) the application requirements; and (c) the presentation requirements.

### **7.1.5 Feasibility and Applicability of Setting Up the Online GIS**

A feasibility study was conducted to find out the key components of a Web-based approach for data integration. An academic department was chosen for the study because of its similarity in working environment with other GIS organizations. Through a series of interviews with the users including teaching staff and students, four types of educational resources to be shared were identified and stated in Section 5.2.1. Present situation and existing problems were also identified and discussed in Section 5.2.2. The targeting system requirements were thus devised (detailed in Section 5.3) and summarized in terms of six categories: (a) user; (b) teaching resources management; (c) data; (d) cartography; (e) functions; and (f) software development configuration.

The feasibility study concluded that a Web-based online GIS should comprise two distinct key components to handle data. One component should handle document resources and the other should handle map resources. Based on the system requirements as stipulated in Section 5.3, a full functional requirement specification

for handling document resources was built and mechanisms to facilitate the handling of document resources were developed.

To select a Web-enabled GIS package for the development of a component to handle map resources, the criteria were based on two aspects: (a) the users point of view; and (b) the implementation point of view.

#### **7.1.6 Prototype for Online Sharing of Teaching Resources**

A prototype system based on the system requirements as stated in Section 5.3 was developed and implemented. The prototype followed the Three-Tier Client-Server computing architecture, which comprised of two components. The MDRC was designed to handle map resources and the TRSC was targeted to handle non-map (document) resources.

The current implementation of MDRC was a non-XML based online GIS and could not support SVG and PNG graphics data. This was due to the limitations of the currently available Web browsers. However, the current implementation of MDRC did give a clear and positive picture that the future development of GIS should be heading towards the Web-based solution.

The TRSC was implemented in HTML and Java and was scheduled into five phases using the modular implementation approach: (a) derivation of mechanisms to satisfy the functional requirements of TRSC; (b) design of database; (c) derivation of

index file format for file transmission; (d) development of Java-based FTP client and FTP server; and (e) development of graphical interfaces.

A detailed discussion of results achieved from the developed prototype, of difficulties encountered during implementation and of anticipated future enhancement were given in Chapter 6.

## **7.2 Recommendations**

Through this study, it was found that current WWW technology was limited in its capability for spatial data integration and delivering online functionality. It was also found that currently there was no comprehensive strategy for complete integration of diverse data for online GIS. The study indicated that a new meta-language for GIS and its strong relationship to an XML should be defined. Therefore, it is urgent to define criteria and requirements for a new meta-language for online GIS. The possible research topics for XML-based online GIS data modeling should be:

- decomposition/Migration of SDTS elements into XML environment;
- knowledge acquisition from different sources;
- knowledge representation and formalization in XML environment;

- spatial reasoning for determining the distribution of data on-demand; and
- standardization of spatial data structure for mobile users.

The study also highlighted that two graphics standards for handling and transferring vector and raster data on the Web would be the solution for relieving the strong reliance on specific vendor format. It was also found that currently Web-enabled GIS applications were providing a limited set of basic GIS query and analytical functionality. Therefore, it would be worthwhile to explore developments with online GIS that should be fully utilizing the rapidly emerging Internet technology such as XML, SVG, PNG and Java. Some possible research topics in this area should be:

- definition of interfaces and parameters for online GIS analytical operations and operators;
- development of algorithms for online on-demand editing/updating spatial data; and
- development of algorithms for on-demand map generalization for mobile users with SVG and PNG.

## REFERENCES

- Abel, D.J. (1989). "SIRO-DBMS: A Database Tool-Kit for Geographical Information Systems." *International Journal of Geographical Information Systems*, Vol. 3.
- Abel D.J., R.G. Ackland, M.A. Cameron, D.F. Smith, G. Walker and S.K. Yap (1991). "EDSS: A Prototype for the Next Generation of Spatial Information System." *Proceedings of AURISA '91, The Annual Conference of the Australasian Urban and Regional Information Systems Association Inc.*, Wellington, New Zealand, 19-22 November.
- Antenucci, J. (1991). "AM/FM/GIS in the '90: Perspectives of Three Italians." *Geo Info Systems*, Vol. 1, No. 6.
- Autodesk (1998). *Using Autodesk MapGuide Server Release 3.0*, Autodesk, Inc.
- Berners-Lee, T. (1991-3). "What is W3?"  
<http://www.w3.org/Talks/General/Concepts.html>, The World Wide Web Consortium.

Berners-Lee, T. et al. (1994). "The World Wide Web." *Communications of the ACM*, Vol. 37, No. 8, pp. 76-82.

Blunt, E. (2001). "Sydney's Information Highway - Easy access to local council information." *GIS User*, No. 44, pp. 26-27.

Bosak, J. (1997). "XML, Java, and the Future of the Web." Sun Microsystems.

Boye, J. (1999). "SVG Brings Fast Vector Graphic to Web."

<http://www.irt.org/articles/js176>, online publication of the irt.org, USA.

Bracken, I. And C. Webster (1989). "Towards a Topology of Geographical Information Systems." *International Journal of Geographical Information Systems*, Vol. 3.

Carter, J.R. (1992). "Perspectives on Sharing Data in Geographic Information Systems." *Photogrammetric Engineering & Remote Sensing*, Vol. 58, No. 11, American Society for Photogrammetry and Remote Sensing.

CSIRO (2000). "CSIRO SVG Viewer Web Site."

<http://sis.cmis.csiro.au/svg/index.html>, Commonwealth Scientific Industrial Research Organisation Australia.

Dangermond, J. (1989). "The Organizational Impact of GIS Technology." *ARC News*, Summer edition, Environment Systems Research Institute, Inc.

- DoE (Department of the Environment) (1987). "Handling Geographic Information."  
Report of the Committee of Enquiry chaired by Lord Chorley. HMSO, London.
- Epstein, E.F. (1991). "Legal Aspects of GIS." In: Maguire, D.J., M.F. Goodchild, and  
D.W. Rhind (eds.) *Geographical Information Systems: Principles and  
Applications*, Longman, London.
- Erickson, F.J. and Vonk, J.A. (1996). *Effective Internet*. Irwin, a Times Mirror  
Higher Education Group, Inc.
- ESRI (1998). "ArcView Internet Map Server Extension: Functional Overview." *An  
ESRI White Paper*, Environmental Systems Research Institute, Inc.
- ESRI (n.d.). "ArcView Internet Map Server Extension"? *A software brochure*,  
Environmental Systems Research Institute, Inc.
- Flowerdew, R. (1991). "Spatial Data Integration." In: Maguire, D.J., M.F. Goodchild,  
and D.W. Rhind (eds.) *Geographical Information Systems : Principles and  
Applications*, Longman, London.
- Gifford, F. (1999). "Internet GIS Architectures - Which Side is Right for You?" *GIS  
World*, Vol. 12, No. 5, GIS World Incorporation.
- Goodenough, D.G. (1988). "The Integration of Remote Sensing and Geographic  
Information Systems." In: Damen, M.C.J., Smit, G.S. Verstappen (eds.)

*Symposium on Remote Sensing for Resource Development and Environmental Management*, Balkema, Rotterdam.

Gould, M. and Ribalaygua, A. (1999). "A New Breed of Web-Enabled Graphics." *GIS World*, Vol. 12, No. 5, GIS World Incorporation.

Guptill, S.C. (1991). "Spatial Data Exchange and Standardization." In: Maguire, D.J., M.F. Goodchild, and D.W. Rhind (eds.) *Geographical Information Systems: Principles and Applications*, Longman, London.

Lang, L. (1992). "GIS Comes to Life." *Computer Graphics World*, Vol. 15, No. 10.

Lanter, D.P. and H. Veregin (1990). "A Lineage Meta-Database Program for Propagating Error in Geographic Information Systems." *Proceedings of GIS/LIS '90*, Vol. 1.

Lemay, L. and R. Cadenhead (1999). *Sams Teach Yourself Java 2 in 21 Days*. Sams, Indiana.

Lunetta, R.S., R.G. Congalton, L.K. Fenstermaker, J.R. Jensen, K.C. McGwire, and L.R. Tinney (1991). "Remote Sensing and Geographic Information System Data Integration: Error Sources and Research Issues." *Photogrammetric Engineering and Remote Sensing*, Vol. 57, No.6, American Society for Photogrammetry and Remote Sensing.



MapInfo (1999). "MapXtreme Java Edition - the First 100% Java Mapping Server."

<http://www.mapinfo.com>, MapInfo Corporation.

Matthews, G.E. (2000). "JPG or lossless for image archives?"

[http://www.wfu.edu/~matthews/misc/jpg\\_vs\\_gif/JpgForArchive.html](http://www.wfu.edu/~matthews/misc/jpg_vs_gif/JpgForArchive.html), online publications on G. Eric Matthews' Web site.

McKeown, D.M. and R.C.T. Lai (1987). "Integrating Multiple Data Representations

for Spatial Databases." *Proceedings of AUTOCARTO 8*, ACSM/ASPRS, Falls

Church, Virginia.

Microsoft Corporation (1997). *Personal Web Server Documentation*. Microsoft

Documentation, Microsoft Corporation.

Morrison, B. and G. McDonald (1991). "Integrating Raster and Vector Data for the

Development of a Land Information System." *Proceedings of AURISA '91, The*

*Annual Conference of the Australasian Urban and Regional Information Systems*

*Association Inc.*, Wellington, New Zealand, 19-22 November.

OGC (1998). "A Request for Technology in Support of a Web Mapping Technology

Testbed." <http://www.opengis.org/>, Open GIS Consortium.

Openshaw, S. and H.M. Mounsey (1987). "Geographic Information Systems and the

BBC's Domesday Interactive Videodisk." *International Journal of Geographical*

*Information Systems*, Vol. 1.

Petroutsos, E. (2000). *Database Programming with Visual Basic 6*. Sybex, Inc., San Francisco.

Plewe, B. (1997). *GIS Online: Information Retrieval, Mapping and the Internet*. OnWord Press, Santa Fe, USA.

Randers-Pehrson, G., and T. Boutell (Eds.) (1999). "PNG (Portable Network Graphics) Specification, Version 1.2." <http://www.cdrom.com/pub/png/spec/>, The PNG Development Group.

Rhind, D.J. (1992). "Data Access, Charging and Copyright and Their Implications for Geographical Information Systems." *International Journal of Geographical Information Systems*, Vol. 6, No. 1.

Roelofs, G. (1997). "History of the Portable Network Graphics (PNG) Format." *Linux Gazette*, January 1997.

Roelofs, G. (1999). "PNG Gaining Acceptance." <http://webreview.com/pub/1999/08/13/feature/index2.html>, online publication of the webreview.com, USA.

Sabio, V. (1999). "Compression Primer for PNG Review." <ftp://ftp.smartbounce.com/pub/png/compression-primer.txt>, online publication.

- Sall, K. (1999). Doing It with SVG (Scalable Vector Graphic).  
*<http://www.wdvl.com/Authoring/Languages/XML/SVG/DoingIt/index.html>*, online publication of internet.com Corporation, USA.
- Shelly, G.B. et al. (1996). *The Internet Introductory Concepts and Techniques*. Boyd & Fraser Publishing Company, Massachusetts, USA.
- Shneiderman, B. (1998). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. 3<sup>rd</sup> ed., Addison Wesley Longman, Massachusetts, USA.
- Strand, E.J. (1997). "GIS Takes Two Roads to the Internet: ActiveX and Java." *GIS World*, Vol. 10, No. 1, GIS World Incorporation.
- Strand, E.J. (1997). "Java Creates New Channels for GIS Information." *GIS World*, Vol. 10, No. 5, GIS Incorporation.
- Thoen, B. (1999). "Industry Trends Reflect the Future of Web-Based GIS." *GIS World*, Vol. 12, No. 6, GIS Incorporation.
- W3C (1996). "W3C Graphics Activity Statement."  
*<http://www.w3.org/Graphics/Activity>*, The World Wide Web Consortium.
- W3C (1998). "Extensible Markup Language (XML) 1.0."  
*<http://www.w3.org/TR/1998/REC-xml-19980210>*, The World Wide Web Consortium.

- W3C (1999). "The World Wide Web Consortium Releases First Working Draft of Scalable Vector Graphics (SVG) Specification." *World Wide Web Consortium Press Release*, The World Wide Web Consortium.
- W3C (1999). "Scalable Vector Graphics (SVG) 1.0 Specification W3C Working Draft 03 December 1999." <http://www.w3.org/TR/1999/WD-SVG-19991203/>, The World Wide Web Consortium.
- Waters, N. (1999). "Is XML the Answer for Internet-Based GIS." *GIS World*, Vol. 12, No. 7, GIS World Incorporation.
- Worboys, M.F., H.M. Hearnshaw and D.J. Maguire (1990). "Object-Oriented Data Modelling for spatial databases." *International Journal of Geographical Information Systems*, Vol. 4.
- Zhou, Q. (1989). "A Method for Integrating Remote Sensing and Geographic Information Systems." *Photogrammetric Engineering and Remote Sensing*, Vol. 55, No. 5, American Society for Photogrammetry and Remote Sensing.
- Zhuang, V. (1997). "Spatial Engines Drive Web-Based GIS." *GIS World*, Vol. 10, No. 10, GIS World Incorporation.

## **APPENDIX A**

### **FUNCTIONAL REQUIREMENT SPECIFICATIONS FOR TEACHING RESOURCES SHARING COMPONENT**

(Part of Thesis work by Geoffrey Y.K. Shea, “A Web-based Approach to the  
Integration of Diverse Data Sources for GIS”)

## **A.1 Introduction**

This document aims at refining the system requirement specifications and then providing functional requirement specifications for the system to be developed to the Department of Land Surveying and Geo-Informatics (LSGI). The discussions between the author and the teaching staff resulted that the system to be developed has to achieve 2 main goals:

1. Automatic centralized file management using a centralized file system to store document type educational resources with user interfaces for file manipulation with different levels of accessibility according to the predefined privilege assigned by the system administrator;
2. Information management using a centralized database to store non-document type educational resources with user interfaces for data manipulation and presentation with different levels of accessibility according the pre-defined privilege assigned by the system administrator.

Based on these goals, this document will illustrate how these two goals could be achieved with the solution described below. In this document, we have to figure out what technical problems are going to be solved for file sharing and demonstrate some possible solutions such as FTP server and LAN server. Meanwhile, we will also mention some insufficiencies in providing file sharing services by merely using FTP server or LAN server. Next, we will shortlist all favorable properties for a system to provide file sharing service. Through the discussion of system and database modeling, we will illustrate the integration of a database and a file system can facilitate automatic centralized file management in file sharing. In turn, we can obtain the preliminary system architecture. By demonstrating the interface design, we can prove that users can interact with the system with the goals without users effort in file management. In effect, we hope this document can specify all the functions related to users in the system.

## **A.2 File management**

In the following discussion, we first start with file sharing scenario, which will illustrate more details about problems in file sharing. Next we will talk about the common and currently available solution for file sharing such as using FTP server and using LAN server and their insufficiencies in providing file sharing services in a multi-user environment. Based on the analysis, we can merge some favorable properties in file sharing that cannot be achieved individually by the above solutions alone. Our basic system model can then be derived from these properties. Through specifying the details of the model, we can obtain our system design with architectural details. Based on the expected system behavior, we can divide the system internally into various functional modules. This figures of the system architecture after modularization can show that our design can simplify not only load balancing but also implementation plan. Having specified all the functions to be supported, we will show that all these functions can be achieved in the interface design. In addition, we can interact with all our expected users with these interfaces. After the following discussion, we would like to prove that the system design below as a whole can facilitate automatic centralized file management with all the educational resources in document type.

## **A.3 File sharing scenario**

In the discussion of the System Requirement Specification, all the shareable document type educational resources could be developed or reproduced so that they can be stored electronically as files. After this standardization of the storage, all the shareable files are scattered around in the local machines of individual staff members. Nowadays file sharing can be achieved commonly by using LAN server or FTP server. However, many management problems occurs during the operation with these solutions.

Staff members have developed their files of notes, assignments, diagrams, presentation, articles and so on, for given courses and given classes that they are required to give lectures in a given academic year. Both staff members and/or students need to access these resources according to their programme, their level, and their courses taken/responsible in this department. As a result, there are relationships among the entities of staff members, students, programme, courses, and academic year. As the time goes, these relationships have significant changes as well. Thus, file sharing leads to the requirement of the storage of relationships with some meaningful entities and these entities forms the indices. For example, If a teaching staff wanted to share a file related to GIS, for all his students and colleagues, then he has to identify all the entities having relationships with GIS, such as all the subjects related to GIS, all the students studied these subjects, and all the colleagues having GIS as research interests. As we can see, it is possible to associate relationships with students, colleagues, GIS, GIS-related subjects, and so on. Here we have to remember GIS is just one of tremendous examples of the relationships exists in file sharing. Since there may be more than one relationship associated to each file, we often want to use multiple indices to index and group those of same relationships. Following to the previous example, if that file is also related to mathematics, we will have difficulty in index this file with other shareable files and store all these indices. Although it is possible to use multiple levels of directories with duplication of files and directories, the static nature of the directory name in the file system does not allow flexible changes of the relationships. Consequently, this problem often results redundancy of files and directories, and it is difficult to manipulate indices so that integrity constraints can be always maintained. Besides, we also have problems to provide security control for file manipulation.

#### **A.4 File sharing problem using FTP server**

For sharing using FTP server, an administrator who is knowledgeable to the files to be shared is required to design the indices optimally for users to search files effectively and efficiently. However, using administrator's knowledge to design indices is not practical because only file owners (or developers) are the best people who are often familiar with the file contents. Administrator, in general, have no knowledge in advance. Thus, for file sharing, the administrator has to ask the file owner about some file information so that a suitable index can be built to facilitate effective and efficient searching and/or subsequent file manipulations such as "update" and "delete" operations. If the amount of the files to be shared is large, the amount of these indices will also be large. Management control turns out to be difficult because in the point of view of the administrator, the indices can be unrelated. The administrator in general has no means to figure out all of their relationships. Thus storage redundancy may occurs.

Moreover, another problem using FTP server for file sharing is that, for proper file and index management, operations like "create", "update", "delete" operations often requires administrator to work on behalf of the users for each file and index. Users have no way to participate and thus the physical communication overhead is great for file sharing. It is also not flexible for file owners to share files too. Nevertheless, we need an administrator because when rights to manipulate files and indices are returned to users, it is difficult to enforce any access control policy in a multi-user environment.

Furthermore, the nature of directory structure does not allow a file with multiple indices without file and index duplication. Although shortcuts or symbolic links are possible but the process to manipulate them have to be manual and often requires administrator's work. Using file and index duplication to achieve multiple indices, we have to pay extra storage and overhead in file and index manipulation. Without file duplication, we could not have multiple indices to achieve efficient and flexible searching.

As a summary, using FTP server to share files requires an administrator to execute management policy and to manipulate files and indices. However, it also encourage tremendous communication overhead and creates inflexibility for file owners to upload, create and update files.

### **A.5 File sharing problem using file system of LAN server**

To allow file sharing using a (centralized) file system of LAN server in a multi-user environment, there are two methods. One is to use publicly accessible (both read-enabled and write-enabled) directories and the other is to assign directories write-enabled to individual users and read-enabled to all users. For publicly accessible directories, it will be impossible to enforce security control because everybody has entire read and write access rights. For users having no participation to control rights (in Microsoft Windows NT 4.0), all the files are readable by all users. Though each user has rights to create their directories to index files, the indices among users does not have a centralized control in manipulation. Thus files of different owners have similar relationships such as types, usage or subject could not be grouped together and it results difficulty for other users to search these files with same indices completely. Thus, this solution will complicate the users to access the files.

In addition, each file owner has to either conform to standard directory naming or publicize his/her shareable directories to other users for each usage. For the former, the effect of file sharing is the same as that using FTP server. It is less optimal as each user has to index files using the standardized directory naming. For file without standardized relationships, indexing file becomes difficult. For the latter, we have to face physical communication overhead among the users again. However, both policies does not allow flexible changes of the relationships held in the directory names.

In conclusion, for file sharing using file system of LAN server, similar problems also arises due to the nature of the directory structure. Files cannot not be multiple indexed without file and index duplication.

### **A.6 File sharing problem using local file system with network services of LAN server**

Although using local file system with network services of LAN server can reduce management problems of storage, since the security control of the file system of individual users are done locally by the service of their local operating system, it results many technical difficulties to facilitate mutual communication across various platforms of different machines and operating systems. Consequently, higher level of centralized security control could not be done. Besides, all the shared files could not be indexed using multiple indices without file and index duplication. Thus, this solution also complicate the process for users to access the files.

### **A.7 Favorable properties for file sharing**

As we can see in the above possible solutions, we have discovered each one of them have their strengths and weaknesses to provide file sharing services in a multi-user environment. FTP server does not allow users to manipulate indices and requires centralized file and index management. However, the interface for file transfer processes allow users to manipulate files and indices easily. On the other hand, general file systems in the LAN cannot allow flexible storage of indices and their relationships with files. Nevertheless, physical file storage still relies on file system. While file sharing using centralized file system simplify the policy in file input and output (I/O), users cannot manipulate the indices flexibly. Similarly, file sharing using distributed file system simplify index management, it will be difficult to implement centralized file management.

We believe an practical file sharing system can combine all the strengths but drop out all the weaknesses mentioned above. By “extracting” the physical storage of the indices and their associated relationships with the files from the directory name, we would like to provide a storage of indices and their relationships outside the file system and establish a mechanism which enables flexible file and index manipulation. Consequently, we try to retain the advantages and eliminate the disadvantages of file sharing using FTP server and LAN server.



Technically, we all know that a directory can only have one string to store one name and a file can only be stored physically in one given directory. We often need to define the name of the directory name first and use its name to index and store files relevant to the directory name inside. The name also gives us information how similar indices could be grouped together as a sub-tree with another directory. What we can store the cardinality among entities represented by indices in a tree structure are only one-to-one relationships and one-to-many relationships, but not many-to-many relationships. It is because many-to-many relationships requires at least 2 fields to specify a key for indexing and one directory name can specify one and only one fields. Obviously, we cannot use the directory structure to represent all database schemas completely. It is possible to use many one-to-many relationships to emulate many-to-many relationships by grouping a particular index value (directory name) in either fields of the keys of the many-to-many relationships. However, it is this kind of grouping results the duplication of files and indices in order to enable the emulation. Although we can use symbolic links or shortcuts to complete the many-to-many relationships, the implementation does not allow flexible changes of the index value because these symbolic links of shortcuts are also static. In a multi-user environment, this implementation is not practically realizable because users needs to provide knowledge in order to index the files and thus they need to manipulate the indices. In order to store the files and maintain the relationships at the same time, we need to use a centralized file system and a centralized database, integrate them, and then establish a policy to control the users' access.

#### **A.8 System modeling and database modeling**

Before the discussion about the information and the relationship to store in the database, we need to model the enterprise schema and anticipate the system behavior.

To facilitate automatic centralized file management, our system model is described as follows: Three types of users namely staff, students and administrator have been defined. Their rights to upload and access files is controlled by administrator. A user who is allowed to upload files is defined as a file owner. A user who can access files is defined as a file consumer. A file has defined to have one and only one file owner and is associated to one access group. In the preliminary design, there are three access groups. One is guest group. Another is student and the other is staff group. A file owner can upload, search, update and delete files. File owner has to specify the application software to be associated with the file being uploaded. The application software can be selected from a pre-defined list (defined by the administrator) or supplied by user. An application software in general has its name, vendor and version. File owner can optionally specify which course(s) is/are related to the file by selecting them in another predefined lists (defined by the administrator). A file can also be associated with a string to specify subjects, keywords, or synonyms. Afterwards, file owner has to specify the location and the filename of the file to be shared. Moreover, file owner can add, edit, delete and update the content of these indices according to some policies. A file consumer can only search and download but cannot upload, update and delete files. File consumer can optionally assert the searching criteria using the indices given or the strings provided by himself/herself. Although file and index manipulation is allowed in this system, these commands are in fact dispatched as requests by user interfaces to a server, which will execute these commands on behalf of the users.

An administrator has responsibility to manipulate user accounts and classify the users into suitable access groups. Different access groups have different level of access rights to the system services. Besides, a administrator has to control some system parameters which can adjust the policy to be uphold during file and index manipulation among the users. Administrator can also manipulate the indices and files without any constraints to be upheld in the system. A user associated with a user account has a user name and a password. For each login, all the access statistics, such as user information, date and time, file manipulation commands, will be recorded.

Depending on the policy of the level of stringency of security control, the following scenario about the detailed user information could also be modeled in the database.

A department has its department code and its name. A department can hold many programmes and offer many courses. In addition, a department employs many staff and has many students. A programme has its programme code and its name. A course has its course code and its name. A student has a student identity number and student name. A staff has a name and title. Each student can join one programme in one academic year at a certain level of the programme. Each programme, according to the level of a student, offer a number of courses. A student can take courses within them. A staff is responsible to a number of courses of different programmes.

As we can imagine, file sharing will generate tremendous amount complex queries which are difficult to anticipate. However, these queries, after investigation, can be reconstructed using the joins of resultsets of some common and basic queries as below by programming. They are further categorized into two aspects namely file and index access aspect and security control aspect.

- File and index access
  - Search files uploaded by a particular file owner.
  - Search files developed by a particular application software.
  - Search files related to a particular course.
  - Search files related to particular subjects, keywords or synonyms.
  - Search file owners and consumers
  - Search application software of particular name, vendor and/or version.
  - Search course of particular name and code.
  - Search subject, keyword or synonyms by a particular string.
- Security control
  - Search files only sharable to a given group of users.
  - Search time and date of the access of a given user.
  - Search user information with a given username.
  - Search files accessed by a given user.
  - Search commands initiated by a particular users, associated with a file.

Based on the queries above, the following information and relationship will have to be stored in the database.

- File and index management and processing
  - File owner: e.g. Geoffrey Shea
  - Filename: e.g. filename\_not\_only\_in\_8.3\_format.doc
  - Access group: e.g. staff only
  - Location to access: e.g. x:\acfms\this\_is\_one\_of\_the\_implementation\_details\
  - Date and time of file uploaded: e.g. 06-01-1999 14:30:55
  - Application software used to develop a given file: e.g. Microsoft Word
  - Course code and name: e.g. LSGI149 Computing for Geomatics
  - Sub-string to specify particular subjects, keywords, or synonyms for description: e.g. notes

- Relationship between a file and its owner
- Relationship between a file and its access group
- Relationship between a file and its physical location to access
- Relationship between a file and its application software used in development
- Relationship between a file and its subject belonging
- Relationship between a file and its descriptions
- Security control processing
  - Username: e.g. rakpchung
  - Password: e.g. rakpchung
  - Access group: e.g. staff
  - Last login: e.g. 07-01-1999 16:28:58
  - Last file accessed: filename\_not\_only\_in\_8.3\_format.doc
  - Last command: e.g. search
  - Relationship between a user and his/her access group
  - Relationship between a user and his/her username and password
  - Relationship between a user and the last login
  - Relationship between a user and the last file accessed
  - Relationship between a user and the last command

To store all these data and relationships, and provide a mechanism to manage and manipulate them, we have proposed a database system. To store all the shareable files and provide a mechanism to manage and manipulate them in a multi-user environment across networks, we have proposed a centralized file system. Next we will talk about the whole system design. The system will form a mechanism to provide their mutual communication and their interaction with the users. Through the discussion of the communication, we will illustrate the integration of the database system and the file system can provide automatic centralized file management in a multi-user environment. We will also discuss some performance and implementation issues of the system design.

### **A.9 System design**

Our solution to achieve automatic centralized file management is to use a database to store the indices and their relationships. Instead of allowing users to interact with the centralized file system directly, we provide another interface to manipulate files to execute on behalf of them. This interface also provides a communication channel to the database for index manipulation. The process transparent to user is the algorithm to map the physical location of files and their associated indices. Other than policy execution, users will not know the process of index manipulations and file manipulations as well as the process to policy to maintain security control. After the integration of the database and the file system, the function of the physical storage of the indices is extracted from the directory name. Then users can build up their indices during file uploading, updating and use multiple indices during file searching.

In the database, tables of records of indices and the relationships associated with the indices will set up. For example, to set up an index about the software, we define an entity software with attributes of its name, its vendor and its version, with its name as a key, to store all the application software to be

used to develop a given file. To relate the file with a given software, its filename and the name of the software will be stored as a record. In effect, we can index all the files developed by any application software. This is possible because we have stored all the necessary information about an application software and its relationship with the files. In this system, courses, users, and brief descriptions about the file contents are also indices which is favorable for searching and will be implemented. As we can see, similar database can also be given to store these indices and thus a file could be associated with more than one indices (multiple indices).

The logical isolation of files and directories does not imply we do not need directories. Note that we still need a file system to store all the shareable files. However, the name of the directory only serves as part of a physical location of files in different groups and the group names are made transparent to users. The relationships between the directory names and the indices will be stored in a database. Thus, manipulations of indices is independent of manipulations of directories. In effect, user will not need any knowledge about the physical file location in uploading and downloading in order to facilitate file sharing. Consequently, file management will become automatic since the physical file location is determined by the indices.

Since the function to index file is now assigned to indices, we now cannot allow the users to manipulate directories. In turn, we no longer need to provide “create”, “delete”, “search”, “rename”, “select”, “copy”, “cut” and “paste” operations for directories but we need to add “create”, “search”, “delete”, “rename”, operations during index manipulation. Moreover, we need to establish a method to convert the corresponding functions for index manipulation to those for directory manipulation and data manipulation. Based on these operations, the integration of the database and file system can now allow all shareable files to be associated with multiple indices with flexible index manipulation.

To allow users to upload files to be shared from their local file system, we need to provide an interface for them to upload and download files and receive the file and index manipulation commands. To allow security control, we also need to provide administrator functions to manipulate users and adjust system parameters in the interface.

As a brief summary, the operations required for the system are derived from five aspects:

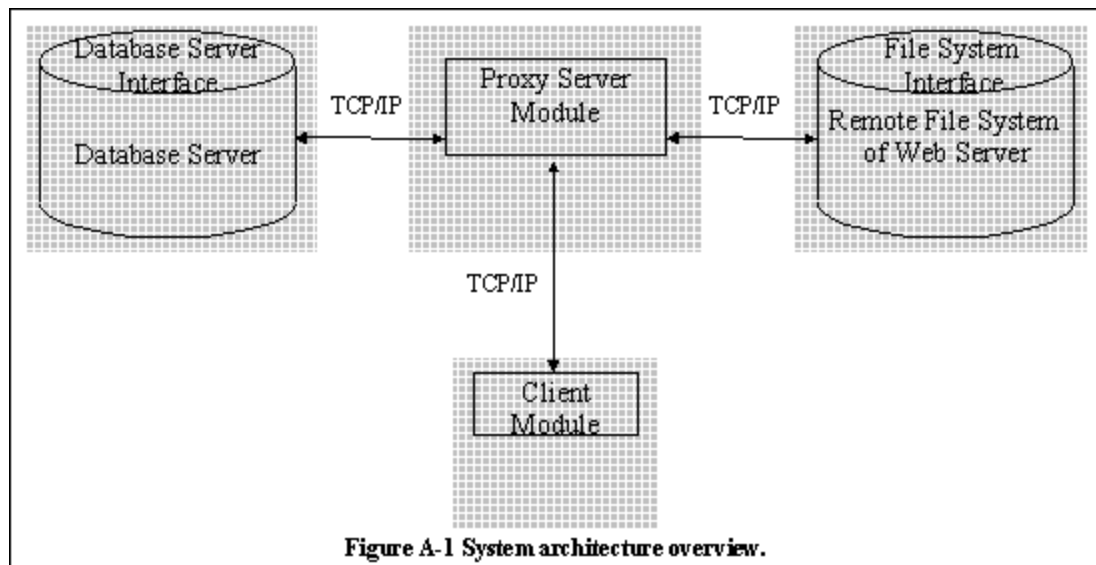
1. File manipulation: create, search, delete, update
2. File transfer: upload, download
3. Index manipulation: add, search, delete, change
4. User account manipulation: add, search, delete, change
5. System access: login, logout, record and view access statistics, adjust parameters

## A.10 System architecture

Based on the current computer facilities of this department, the following diagram describes the overview of the system architecture. To simplify implementation, the system is divided into 4 modules and they are designed to take up different functions related to database management, file management, policy execution and internal communication, and user interaction. Figure A-1 shows the system architecture overview of the current design.

In this design, the consideration is mainly based on the load balancing between data I/O and file I/O. Meanwhile, we need to isolate communication between and the users and either the database or the file system. This division of responsibility not only allow each module to specialize on functions of each logical purpose, but also reduce the implementation overhead.

To achieve platform independent execution, Java programming language is used to develop this



system and Java Development Kit (JDK) 1.2 is used to provide a PC development platform for Java.

The four modules are:

- the Database Server Interface Module, which interacts with the underlying database management system. The database management system is provided by a third-party vendor and is selected to be the Microsoft SQL Server 7.0.
- the File System Interface Module, which execute all the file manipulation requests in the centralized file system. The centralized file system is provided by Microsoft Windows NT 4.0
- the Proxy Server Module, which provides a standardized protocol to communicate and then work together. It is used to execute the policy to manipulate all the files and their relationships, and centralize all the file and data manipulation requests and then redirect to the Database Server Interface Module and the File System Interface Module.
- the Client Module, which provides different interfaces to receive users' commands to manipulate the shareable files. Different interfaces will provide different set of functions to the different types of users.

In the coming discussion, we will illustrate the architecture and the high-level data flow among the modules. The interface between each module and its operating environment will also be shown.

#### **A.11 Architecture of Database Server Interface Module**

In the Database Server Interface Module, a Java program will be developed to interact with the database, which is selected to be Microsoft SQL SERVER 7.0. It is a database server to provide all the physical data storage. The data to be stored will include all the file indices and associated relationship to enable the mechanism executed in the File Server Interface Module for automatic file management. To enable the application developed running in Microsoft Windows independent of the underlying databases, Microsoft has defined a vendor independent interface standard to connect databases from different vendors. The standard is called the Open Database Connectivity (ODBC). All DBMS running in Microsoft Windows have to provide their ODBC driver. This driver is responsible for communication with the database via the operating system. System service like network data transmission can be provided through this channel. However, since the call level interface is written in C, to facilitate communication using Java, Sun has developed database connectivity application programming interface for Java, called JDBC API, which facilitate Java to program database of independent vendors. Since we need to C interface to Java interface, a driver, called JDBC-ODBC bridge, is required to facilitate this conversion. The Database Server Interface Module is built up on top of this driver. Figure A-2 shows the architecture of the Database Server Interface Module & database server system.

Using the JDBC-ODBC bridge for Microsoft SQL SERVER, we can use Java to connect the database and access the data stored inside using Java. All the commands for index manipulations is directed from Proxy Server Module and The Database Server Interface module will centralize all these commands and then provide all the data I/O services. These services will allow users to add, search, delete and change indices.

#### **A.12 Architecture of File System Interface Module**

In the File System Interface Module, a Java program will be developed to interact with the centralized file system. This file system will be supplied by a network server. Physical file manipulation will require some file service of the underlying operating system. This module will then centralize all commands for file manipulation from the users. All these commands directed to this module will be executed here on behalf of the users. Since file indices will have been set up in the database, searching files can rely on these indices because the indices and the physical location of the file is related under the policy held in the Proxy Server Module. These relationships will also be stored in the database. As a result, the physical file location can be obtained with some database queries. The management of all the directory names will be left to the details of the policy. Figure A-3 shows the File System Interface Module and the file server architecture.

Under this design, the File System Interface Module can provide all necessary file manipulation services. These services will allow users to add, search, delete and update files.

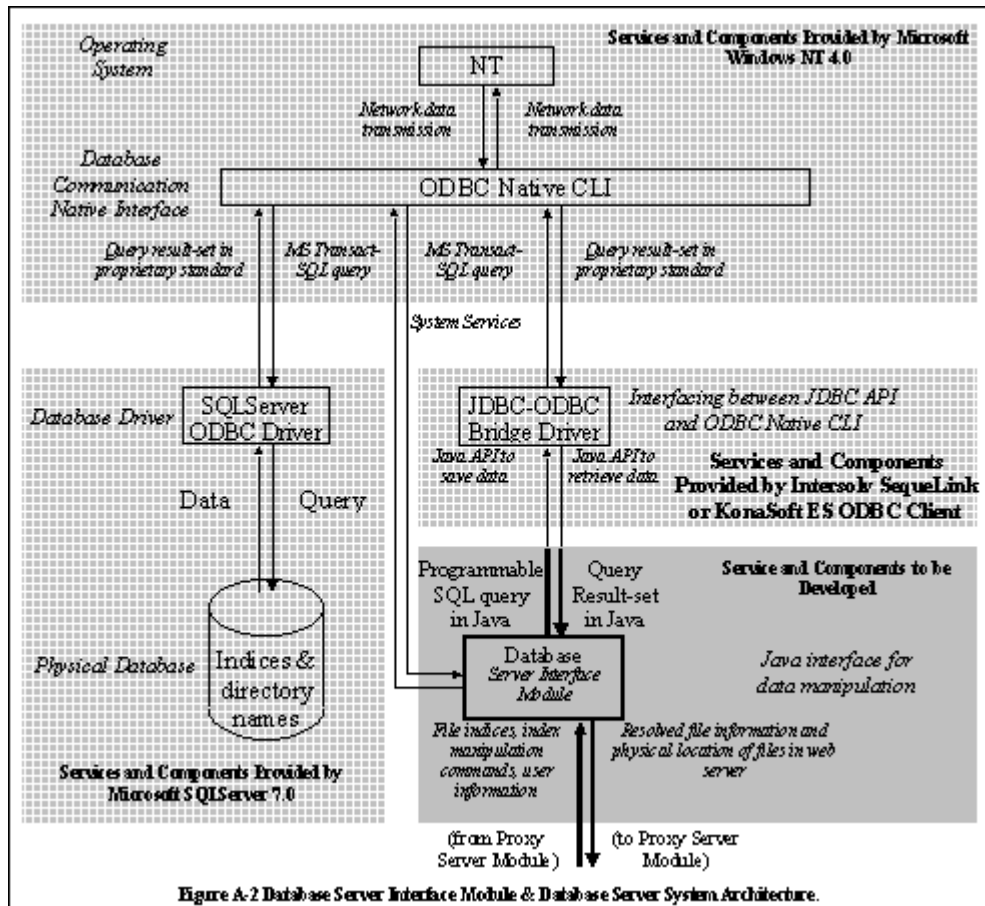


Figure A-2 Database Server Interface Module & Database Server System Architecture.

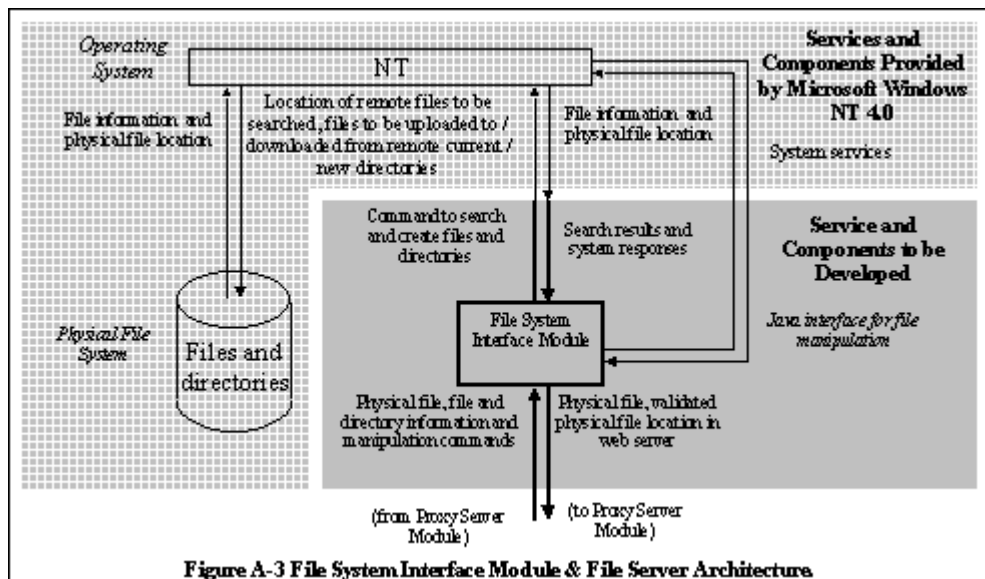
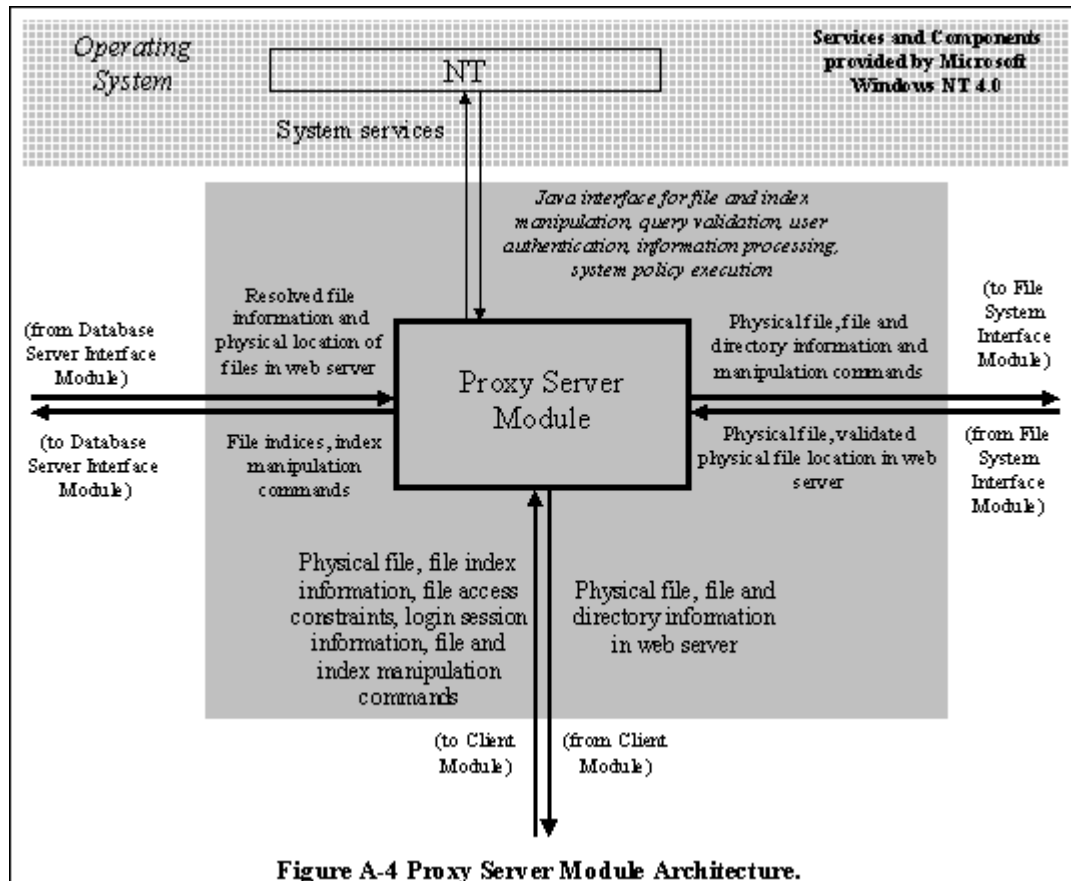


Figure A-3 File System Interface Module & File Server Architecture.

### A.13 Architecture of Proxy Server Module

In the Proxy Server Module, a Java program will be developed to receive all the user commands of index and file manipulations for analysis. Besides, this module will be used to execute some administration work, such as authentication of the users' access rights, analysis of the users commands, formulation of data records for suitable tables in the database, generation of directory names, redirection of file and index manipulation commands (under certain access control policies), responses to users and so on. Figure A-4 shows the Proxy Server Architecture.

Using this module, we can facilitate the communication among all the users, the file system, and the database.



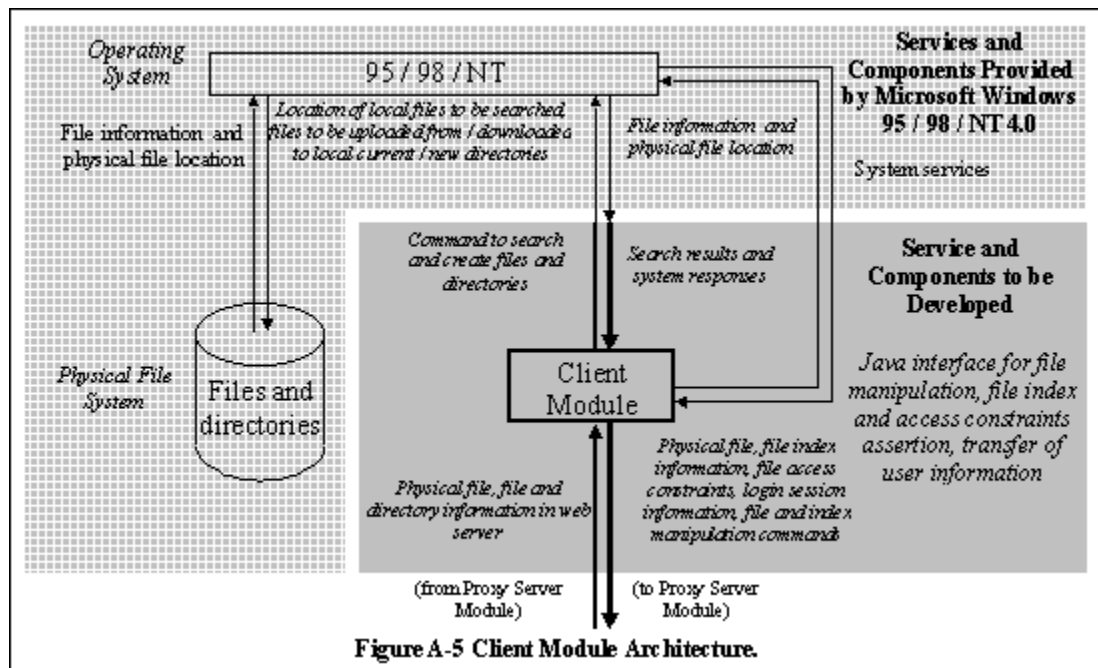
### A.14 Architecture of Client Module

In the Client Module, there are user interfaces to specify different functions for file and index manipulations. Under the access control policy to be enforced in the Proxy Server Module, users will have to login and receive authentication. For successful login, user can assert file and index manipulation commands via these interfaces to the system and then upload the files to be shared and download the shared files. After assertion, the user interfaces will analyze all the asserted information for files and indices. The result (indices and/or files) will be sent to the Proxy Server Module to process further. These processes, such as separating file and index storage, creating suitable relationships between indices and files, and so on will facilitate automatic centralized file



management. Meanwhile, the Proxy Server Module will communicate with the Client Module to update any system status. Consequently, we target automatic file management that all users will not need to bother the physical storage of the shared files, but the system can still offer them an easier way to access all the shared files via these interfaces.

Other than commands for file sharing, we also need to allow an administrator to have the most superior administrative power on the system for user management and system management. In this module, the assigned administrator (by agreement) can add, search, change, and delete users via an interface. Administrator can also adjust the system parameters, wherever suitable for the benefits of file sharing. Commands issued by administrator will also be sent to the Proxy Server Module to provide adjustments to the access control policy. On the other hand, administrator can also trace all the users' access history and monitor if there is any misbehaved events for all other users. Figure A-5 shows the architecture of the Client Module.



### A.15 Functional requirement specifications

For the Client Module, there are 3 types of interfaces namely:

1. Login Interface, which allows all the users including the administrator to receive authentication before accessing the system. There are 2 main functions to interface with the users:
  - a. Receive authentication;
  - b. Access the following interfaces depending on the user identity for successful login.
2. File Sharing Interface, which allows users to manipulate files and indices. There are 14 main functions:
  - a. Upload files;

- b. Create files;
  - c. Search files;
  - d. Update files;
  - e. Delete files;
  - f. Download files;
  - g. Locate local files to be shared;
  - h. List files under given searching criteria;
  - i. Select level of accessibility of a given file;
  - j. View level of accessibility of a given file;
  - k. Create indices;
  - l. Search indices;
  - m. Change indices;
  - n. Delete indices.
3. Administrative Interface, which allows administrator to adjust system parameters during operation. There are eight functions:
- a. Add users;
  - b. Search users;
  - c. Change users;
  - d. Delete users;
  - e. View users details;
  - f. Change users details;
  - g. View access details;
  - h. Configure system parameters.

#### **A.16 Interface Design for the Client Module**

As mentioned above, the three types of interface are namely the Login Interface, the File Sharing Interface and the Administration Interface. The main graphical user interface for the Client Module to interact with the users is shown in Figure A-6 below. In the following session, we will illustrate the design of some critical interface. It is advisable to note that the interface design of various interfaces is just a prototype. The proposed layout just only reflects that the interface can be designed to facilitate the functions listed above.

#### **A.17 Interface design prototype for the Login Interface**

Based on the requirements listed above, Figure A-7 shows the interface design for the Login Interface.

In this interface, each user has to receive authentication before obtaining file sharing services using this system. By typing in username and password in the corresponding text-fields and then pressing the button “Login”, user will accept authentication. Any system response after authentication will be given to the users. There are mainly three system responses. One is to invite users to login such as “Please login”, another is to indicate successful login such as “Login successful”, and the other is to indicate

incorrect login such as “Login incorrect”. This interface design of the user interface can also allow re-login mechanism and thus provide button “Clear” and button “Cancel” for changes in the typed username and password.

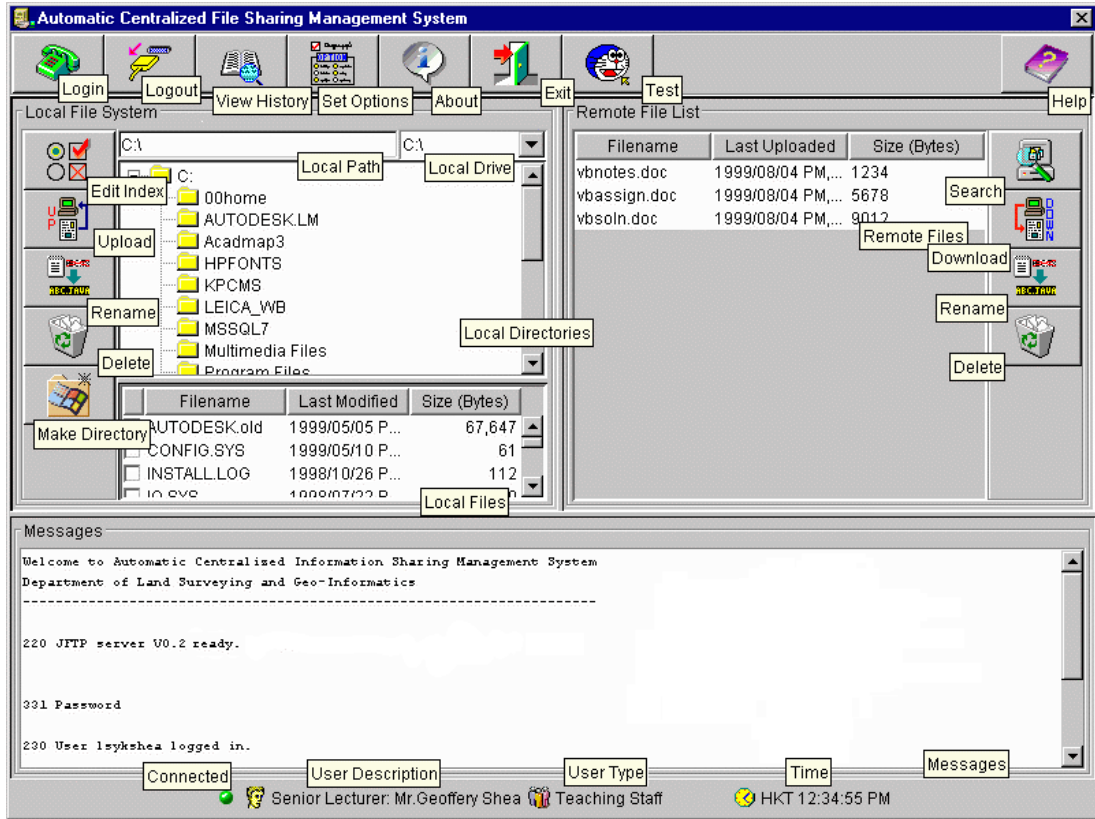


Figure A-6 Prototype graphical user interface of Client Module

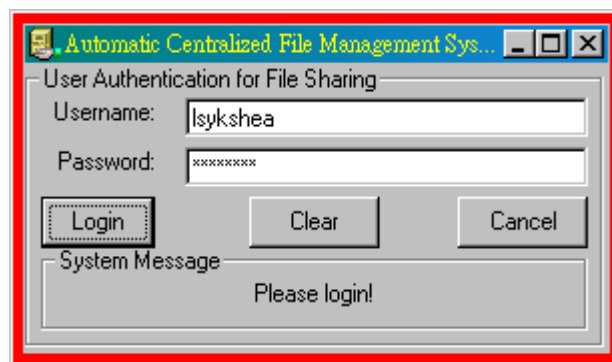


Figure A-7 Interface design for the login interface.

## A.18 Interface design prototype for the File Sharing Interface

Based on the requirements listed above, Figure A-8 shows the interface design for the File Sharing Interface.

After successful login, the system will provide a session “Start Up” for users. If the user logged in is an administrator, all the four buttons namely “Upload”, “Admin”, “System” and “Search” will be visible to him/her. If the user logged in is not an administrator, only 2 buttons namely “Upload”, and “Search” will be visible in this session. If the button “Upload” is pressed, the session “File Uploading Session” will be visible. Similarly, if the button “Search” is pressed, the session “File Searching Session” will be visible. In each session, the system status will show the user details, server date and the server time to the user. Figure A-8 demonstrates the preliminary design of the File Sharing Interface (Start Up).

As the name implied, the session “Start Up” provides choices for users to select his/her role as a file owner or a file consumer in each login session. If user wants to become a file owner, user can use the session “File Uploading Session” to upload files, create files and update files. If user wants to become a file consumer, user can use the session “File Searching Session” to search files, delete files or download files. Figure A-8a demonstrates the preliminary design of the File Sharing Interface (Uploading).

In the “File Uploading Session”, users can locate the files to be uploaded in two ways. User can either type the exact file location of the file, or use the file browser to locate the file. Based on the file contents, and other relevant file information, user can use various kinds of indices to index the files. There are two types of information which are compulsory for uploading. One is the file location (with the filename) and the other is the application used to develop the file. To allow other users to search the file and further delete or download the file easily, file owner can optionally index the file using 3 other methods. One is related to the courses, which is pre-defined by administrator. Another is related to knowledge, which can be user-defined. The other is related to file contents such as synonyms, subjects, or keywords, which is completely user-defined.

This distinction of various level of indexing allow the use of multiple indices and eliminate all the index management required in file sharing in a multi-user environment. However, the system still maintains users’ participation in the choice of indices associated to the shared files.

When file owner presses the button “Send”, the file in his/her local file system and all the associated indices will be sent to the Proxy Server Module for analysis. Those index information will be stored in remote database, while the file will be store in the remote file system. The Proxy Server will then use the policy executed in the Proxy Server to use the index information to define directory names, assert commands to create suitable directory accordingly, link up with index information with the physical location, and save the data and file accordingly by the Database Server Interface Module and the File System Interface Module. In effect, the shared file will be created as a copy in the remote file system.

Other than the files and data, it is also possible to save the uploading event and provide him/her the history of uploading or updating. Here uploading and updating are very similar. Both of them need a file transfer from the local file system to the remote file system with associated indices. The only difference is to find out that whether the file to be shared exists or not. If it does not exist, the system will create the file directly. If it exists, the system will first delete all the associated index information before the actual file creation. Here, the policy for file identity is stated as follows: Two files are said to be the same if the user is the file owner and the filenames are the same. For each updating, users will also be prompted for assertion of the action again.

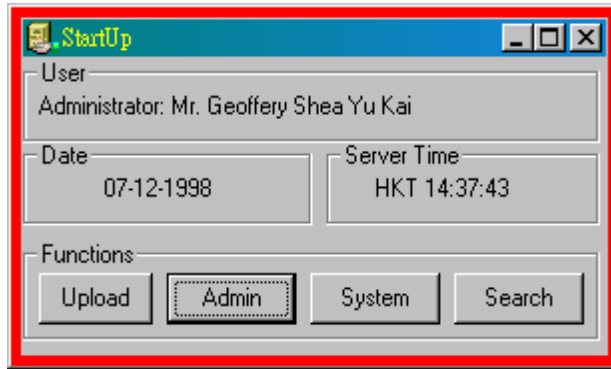


Figure A-8 Interface design for the File Sharing Interface

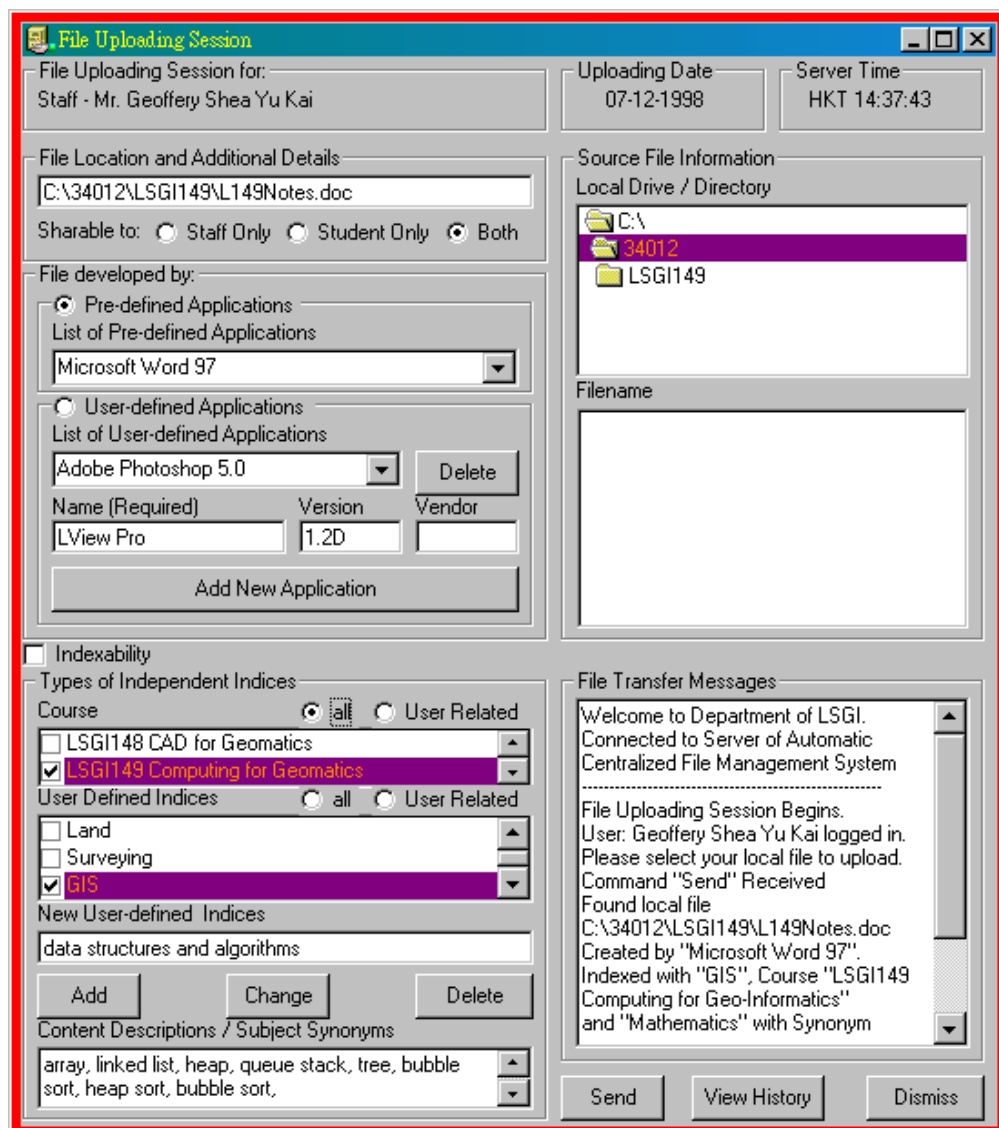


Figure A-8a Interface design for the File Sharing Interface (Uploading)

Uploading several times, users will also need to know the files uploaded to the system. As a result, the button “View History” can cater for this need. All the files uploaded and the associated indices will be shown, when he presses the button.

When downloading is finished, user can close the application by pressing the button “Dismiss”.

In the “File Searching Session”, based on the indices defined by the file owners, each file consumers can search the shareable files using these indices. After selecting the indices, the file consumer can press the button “Search” to search the files. The system will make use of these information to form queries and display the result as a list of file names. Users can then download the file by pressing the button “Download”. While a given user want to know more details about the file, user can press the button “View File Details” for this purpose. Figure A-8b demonstrates the preliminary design of the File Sharing Interface (Downloading). If user is in fact the file owner, user can also delete the file or change the file details by pressing the “Delete” button.

Depending on the choice in the radio buttons of the given file consumer, he/she can limit the searchable indices. For each user command, the system will give corresponding response in the text area. He/she can also view his/her access history by pressing the button “View History”. Finally file consumer can exit using the button “Dismiss”.

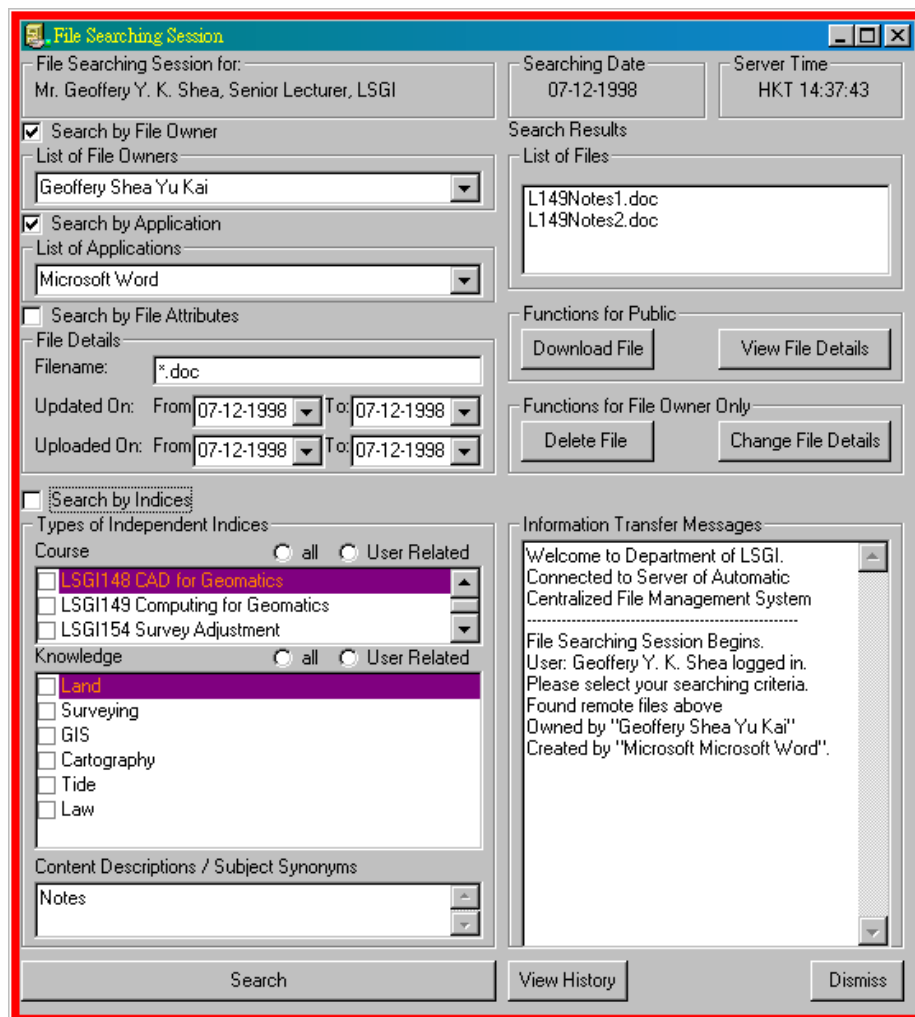


Figure A-8b Interface design for the File Sharing Interface (Downloading)

### A.19 Interface design prototype for the Administration Interface

Based on the requirements listed above, Figure A-9 shows the interface design for the Administration Interface.

In the interface “Start Up”, if the administrator presses the button “Admin”, the system will provide the administration session for the administrator. In this session, the administrator can add and delete user accounts by pressing the button “Add User” and “Delete User” respectively. A suitable dialog will be given subsequently to extract information from administrator. The administrator can also edit and view user information by pressing button “Edit Info” and “View Info”. Administrator can also trace user activities and view the access log in this session.

After the above discussion, what we have left is the discussion on the function of the button “System”. Pressing this button, the administrator can change some system parameters about the details in the policy enforced in the “Proxy Server Module”. One of the functions is the manipulation of the pre-defined indices such as those related to courses and knowledge.

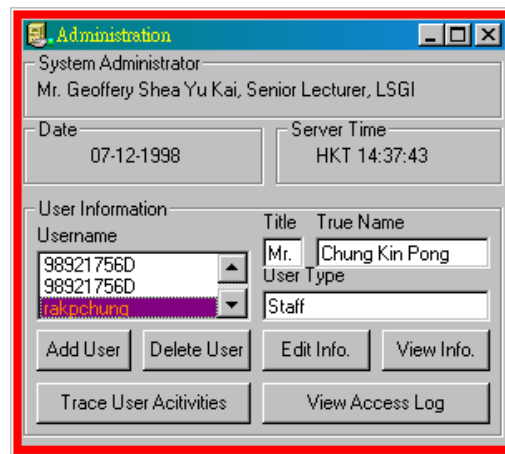


Figure A-9 Interface design for the Administration Interface

### A.20 Advantages

In preparing this design, we have investigate the implementation feasibility from the technical point of view. What we want to achieve is file sharing and our focus is to achieve automatic file management. To reduce the implementation cost, we have proposed a centralized file system. In addition, we have also proposed the use of indices to associate all logical relationships with the files and in turn enable multiple indexing of a given shareable file. This will overcome defects to maintain the many-to-many relationships by just using a directory structure in a file system and its resulting cost in using file and index duplication. Since free lunch does not exist in the world, the price of this automation in file management results from the establishment of the communication between the database and the file system. To facilitate the storage of the indices and these relationships, we have also proposed a database to complete this task. The manipulation of the files and the indices is then made transparent to users by the cooperation of all the modules described above.

To facilitate the users to interact with the system, the prototype of various interfaces have been demonstrated. Using this architecture, the management work for all users, especially the file owners, is significantly simplified and reduced. What they need to do to enable files to be shared is to assign all

the necessary indices and supply some additional information for indexing. In the view of the improvement on user-friendliness, the interface design has enabled the flexibility for users to define and manipulate their favorable indices. Besides, it also can allow users to use index search with these indices. Besides, using these indices, users without any information about the file contents are not refused to search files. It is no longer necessary for them to download the file in order to know the file content. (Of course, the more is the information contributed by the owner, the easier is the file accessed by the consumer.) This capability is very useful in file sharing among the teaching staff. In particular, image files can also be indexed with its contents! Obviously, the capabilities integrated are never achieved completely by merely using a LAN server or a FTP server.

Other than the file owners and consumers, the administrator can also obtain benefits in file sharing using this system. Note that the programs have done all of the work in the file management, the programs, the file owners and the administrator will share the work in the index management. In the program, all the work of index I/O and query formulation will be assigned to the programs. Since users can now participate in index manipulation, the administrator will share the remaining work resulted from pre-defined indices. When comparing with the work in file management solely done by the FTP site administrator, the workload for the administrator here is greatly reduced. Nevertheless, we did not forget to enforce security control and access control while maintaining the above capabilities. We believe, by using this design, we can achieve file sharing with automatic centralized file management.



## **APPENDIX B**

### **MECHANISMS TO FACILITATE THE TEACHING RESOURCES SHARING COMPONENT FUNCTIONS**

(Part of Thesis work by Geoffrey Y.K. Shea, “A Web-based Approach to the Integration of Diverse  
Data Sources for GIS”)

## **B.1 Introduction**

In order to facilitate the functional requirements stipulated in Section 5.3.5, the following mechanisms were devised for the Teaching Resources Sharing Component (TRSC). Such mechanisms were designed before the actual implementation of the prototype.

## **B.2 Mechanism to store the resources**

After investigation, a central file system and a central database would be suitable for this application because policies could be developed, embedded and executed inside a system being implemented. It could not only reduce platform dependency in storage and security but also simplify the implementation. Based on the computing facilities in the department of LSGI, the file system of a machine with Microsoft Windows NT Server 4.0 installed would be used as the central file system of this application. Database server Microsoft SQL Server 7.0 was selected to be the central database server. The reason of using Windows NT Server 4.0 resulted from the fact that this operating system could share part of a file system for the application to use. Besides, this operating system provided an open interface to interact with database of different vendors. The interface is called Open Database Connectivity (ODBC). Using the NT Server and the SQL Server, the system could store files and data, using the Server Module.

## **B.3 Mechanism to input and output the resources**

Since the system to be developed would be distributed, 2-tier or 3-tier client-server based architecture would be favorable in this application. The second tier in the 2-tier system or the third tier in the 3-tier system would be responsible for the input and output of all the shared files. In the current design, it would be a joint two 3-tier systems with common first tier and a merged second tier. One of the third tier would be connected to the database and the other would be connected to the file system. The responsibility to input and output the resources would be shared by these third tiers. Formatted manipulation requests of files and indices would be passed to and from the Proxy Server Module. When Java programming language could be employed in development, its API could provide functions to manipulate files in the file system and indices in the database management system. Basically, the Java class package `java.io` and `java.sql` in JDK enabled the system being developed to embed these capabilities. Using these packages, the File System Interface Module could manipulate files and the Database System Interface Module could manipulate indices.

## **B.4 Mechanism to control the system and provide security**

The control of the system could be divided into three aspects:

### 1. Authentication

In providing the control on the access of the system, the system could provide authentication process to every user before any legal file transfer request. In effect, all the users would be expected users. Addition of users would be done by some pre-defined administrators. As the database would be designed to store some system-wide user information, user records could be added in the database to provide rights for users to access the system.

### 2. File access

As shown in one of the database schemas, file owner would be assigned to become one of the attributes of a given file. In the current design, any modification like update, and deletion of a given shared file had to be done by the user who upload that file. For supervisor control, administrators could also have the rights to do so.

### 3. Connection control

As Java could provide TCP/IP standard in network communications, it would be possible to embed connection control policy in filtering unexpected connections from un-trusted hosts.

### **B.5 Mechanism to access the system**

Users were required to get authenticated by the system before any legal access of any shared files. The Server Module would manipulate all the files and indices on behalf of the users. The system as a whole was designed to provide seamless FTP functions for them to get and put the shared files, and help them to index the files with user-defined additional information, without any concerns on the file management and security in a sharing access environment.

To adapt the window-based operating systems like Microsoft Windows NT 4.0 or X-Windows, the user interface provided by the Client Module would be graphical so that user can assert the user commands and obtain the system responses conveniently in the same session window. The interface would be used to collect the pre-defined user commands. In receiving events like clicking some buttons, the system could allow users to input and output the shared files according to these commands. These user commands would be first analyzed in the Client Module and those involving file and index manipulations would be redirected to the Server Module for processing. After the reception, the Server Module would reformat and separate the file and index manipulation commands, and then complete them on behalf of the users. The manipulation result, whether successful or unsuccessful, would be returned to the Client Module, and then illustrated to the users on the interface.

Using this mechanism, the system could be designed to have the following advantages:

- Access and security policies could be established, embedded and executed solely in the Proxy Server Module, and then different levels of services could be provided according to different types of users.
- File management and manipulations would be accomplished solely by the File System Interface Module.
- Index management and manipulations would be accomplished solely by the Database System Interface Module.

To enable the users to access the system, the following commands would be available to the users in the user interface.

#### 1. *Execute the Client Module*

The system would be deployed as a Java application. To execute this application, users would be required to install a plug-in, called Java Runtime Environment, to standard their computer systems as a Java platform in order to run the application. When this plug-in had been installed, only a line of command in the prompt could allow the client program to be executed for file sharing.

#### 2. *Exit the Client Module*

This command would enable users to quit the client process.

#### 3. *Connect to the Server Module*

This command would allow users to connect the executing server process. Users who had not connect to the Server Module would be disallowed to access the resources. To connect the server, username, password would be required for logon the server. If the authentication were successful, he/she would be allow to upload, search, download, update, and delete the resources. If the authentication were failed, he/she could not be able to issue the above operations.

4. *Disconnect from the Server Module*

When a user finished the session, he/she could disconnect from the server using this command.

5. *Edit file information (or indices)*

This command would allow users to edit the file indices. These indices would be used to associate a file for subsequent access in the server. All these indices would be stored in the database management system of the server automatically when the corresponding file had been uploaded to the server. After a file had been associated with an index, there would be an indication for that file, which could be used to distinguish from those without indices.

6. *Clear file indices*

This command would allow users to clear the file indices of those file which had been associated with indices before.

7. *Select files from the local file system*

To select files to be shared, the graphical user interface would provide user interface controls over the local file system. User could select drives, directories and files from these controls, by using a mouse.

8. *Upload files from the file system of the user*

After a file in the local file system had been selected, this command would allow user to upload the file to the server.

9. *Rename local files*

Sometimes, user could download files of the same name. They could then use this command to rename the local files of the same name and then further the downloading process.

10. *Delete local files*

Sometime, user could download files of the same name. They could then use this command to delete the local files of the same name and then further the downloading process.

11. *Create directory in the local file system*

Sometime, user would like to download files to certain new directories. They could then use this command to create new directory in the local file system.

12. *Search the Server Module (using indices)*

In the Client Module, a graphical user interface would be provided to users to use indices to search the server. Based on the selected criteria, the Server Module would return the search result to the Client Module and then display to the users.

13. *Select files from the remote file system*

Having obtained the search result, all the expected files would be listed in the graphical user interface of the Client Module. User could select any one of the items in the list for further downloading.

14. *Download files from the Server Module*

This command would allow user to download the selected file from the Server Module to the Client Module.

15. *Rename remote files in the server (according to some pre-defined rights)*

If the user was found to be the owner of the selected file or the administrator, then the Client Module would allow the user to issue this command to rename the remote file in the server.

16. *Delete remote files in the server (according to some pre-defined rights)*

If the user was found to be the owner of the selected file or the administrator, then the Client Module would allow the user to issue this command to delete the remote file in the server.

In effect, the Server Module and the Client Module would be designed to compromise a suitable communication interfaces so that to meet the following two conditions: (1) all the file and indices manipulation request could be redirected from the Client Module and the Server Module; (2) the communication protocol could favor one server and multiple clients scenario.

## **B.6 Mechanism to manage the resources**

In this application, additional information would be used for efficient file retrieval. The mechanism to achieve this objective was designed to make use of indices. The distinction between using indices rather than using directory names to group files, or formally speaking, maintaining some logical relationships among files would be illustrated in the following paragraphs.

When files are to be indexed using the directory names, because of the hierarchical nature of a tree model, which was embedded in any types of file systems, only one-to-many but not many-to-many relationships could be stored. If many-to-many relationships were to be emulated using a tree model, it would be difficult to maintain the integrity constraints of the entities represented by the directory names. As a result, it was easy to encounter the redundancy and consistency problem while managing the files effectively and manipulating file efficiently. It happened that these two criteria often led to conflicts: If no redundancy was allowed to maintain consistency, it would become convenience to enforce the integrity constraints of the relationships but inefficient to store and retrieve the files. On the other hand, if consistency was sacrificed to allow redundancy, it would be efficient to manipulate files but difficult to management the complex relationships.

These problems occurred because tree model was not designed to maintain consistent relationships in the nodes (i.e. the directories). As a result, sharing files often encountered conflicts in efficient file manipulations and effective file management. The conflicts could be spread to the distribution of work in the system design for file sharing. When efficient file manipulation was emphasized, a lot of work had to be done to manage the files. When effective file management was emphasized, a lot of work

had to be done to manipulate the files. A file sharing system would become useless when the shared files could not be managed properly.

In this application, the maintenance of the relationships among the files was designed to be the responsibility of a database. It was known that a database could maintain relationships of data efficiently and effectively. It was because standard data manipulation language Structural Query Language (SQL) had been defined for data manipulation. Thus, the remaining work to make the system workable was to construct a database and the format of the index files.

Appendix C described fully the design of database schemas and the relationships of the tables employed by the TRSC and Appendix D provided a detail description on the format of the index files to be used by the TRSC.

## **B.7 Mechanism to facilitate network communications between users and the system**

The basic communications between the Client Modules and the Server Modules were governed by an international standard protocol called File Transfer Protocol, which was officially specified by J. Postel and J. Reynolds in October 1985 in RFC 959. This specification has given detailed explanations of all the necessary terms and demonstrated the idea of FTP and explained all the user commands and their functionality. Examples were also given to show the use of commands in a client and the way to deal with the commands in a server. The specification of RFC959 could be found in the Internet at

<http://sunsite.auc.dk/RFC/rfc/rfc959.html>

In this application, in addition to files transfer, additional information would be assigned during uploading and would be used in searching before downloading. These pieces of additional information were defined as indices and would also be transferred to the Server Module. The system was designed to embed a modified version of this protocol so that these indices could be transferred together with each file manipulation operations.

When indices were associated to files for uploading, or selected for downloading, they were stored temporarily in some files known as Index files. Ordinary file transfer would also require the transfer of these files. As a result, a new protocol would be required to facilitate the transferral of index files in association with ordinary file transmission. The new protocol came from the modification based on RFC959. The modified version of RFC959 would consist of two parts: (1) introduction of new commands; and (2) termination of some of the original commands.

### **B.7.1 Introduction of new commands**

In the modification, indices associated to the shared files would be created and stored as index files. These index files would be transferred between the client and the server during some of the commands in FTP. The modification would focus on how to transmit the index files so as to maintain logical transaction of shared files, and the subsequent access of the shared files. In this application, three new commands were added to provide the transfer of the indices. In fact, these commands were composed by some of the commands in the original FTP. They were:

1. *LOGIN (Login)*

The Login command is composed of three commands of the original FTP. They are namely USER command (for sending username), PASS command (for sending password) and RETR command (for retrieving file). Using a graphical user interface, users could input the information about his username and password to the system. When the user issue the new command, both the username and the password would be sent the server for authentication. If the authentication is successful, the Server Module will acknowledge the Client Module the

success and the Client Module will get the index file. This index file is specifically defined as "User profile". Each user profile is in fact stored in the database of the Server Module and they are unique. The user profile is used to provide information to the user about the indices available for assignment to files and file search.

2. *UPLD (Upload)*

The Upload command is composed of two commands of the original FTP. They are in fact two STOR commands (for storing files). The first RETR command is used to provide file transfer of an ordinary file from the Client Module to the Server Module. The functionality is equivalent to that in the original FTP. The second RETR command is designed to provide file transfer of index file an index file in which the indices are those being associated to the ordinary file in the graphical user interface. An uploading command is defined to be successful if both the ordinary file and the index file have been received by the Server Module successfully. When either one of them, or none of them cannot be received successfully, the Upload command is defined to be failed. This index file is used by the Server Module to determine how the ordinary file is to be stored in the server and retrieved from the server in the future.

3. *SRCH (Search)*

The Search command is composed of two commands of the original FTP. The first one is the STOR command and the second is the RETR command. The STOR command is to provide transfer of an index file in which the indices are the searching criteria of some expected shared files. When the index file have sent from the Client Module to the Server Module, the Server Module then analyze all the searching criteria and complete the search automatically. The search result is generated as another index file and will be retrieved by the Client in the subsequent RETR command. The search result in the second index file will be re-presented by the Client Module and display to the user in the user interface. User using the result can download and (conditionally) delete the files.

The structure of all the new commands was shown in the Figure B-1.

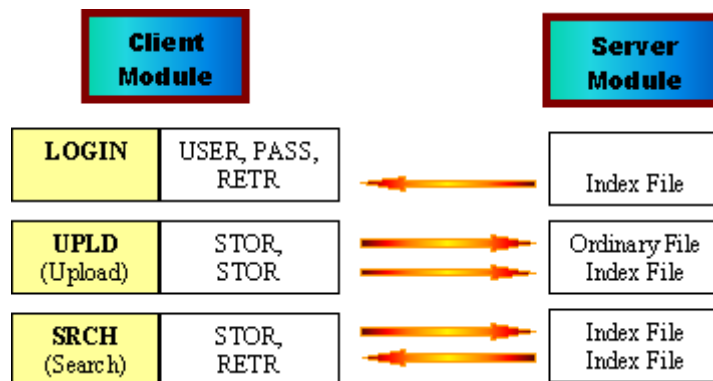


Figure B-1 New file transfer commands for TRSC.

### **B.7.2 Termination of original commands**

To provide automatic management of the shared files, user commands in the original FTP for direct directory access would be disabled. Access of the directories in the remote file system would be performed by the Server Module. The user commands being disabled in the client module were:

1. *CWD (Change Working Directory)*

CWD basically allows users to access a given directory directly for subsequent file transfer or directory browsing in the FTP server. Since the file management is done by the Server Module and not by the users, users will no longer need to change the working directory for file access. Thus, this command will be removed from the user interface in the Client Module.

2. *LIST (List Working Directory)*

LIST basically allows users to list a given directory to find if the given directory contains some expected files and subdirectories in the FTP server. Since the new command Search will help the user to find the expected files, users will no longer need the LIST command to locate those files. Thus, this command will be removed from the user interface in the Client Module.

3. *RMD (Remove Working Directory)*

RMD basically allows users to remove a given directory in the FTP server. Since the file management is done by the Server Module and not by the users, users will not be allowed to delete directories. Thus, this command will be removed from the user interface in the Client Module.

4. *MKD (Make Working Directory)*

MKD basically allows users to make a directory in the FTP server. Since the file management is done by the Server Module and not by the users, users will not be allowed to create directories. Thus, this command will be removed from the user interface in the Client Module.

5. *PWD (Print Working Directory)*

PWD basically allows users to get the current working directory in the FTP server. Since the file management is done by the Server Module and not by the users, users will no longer need the PWD command to determine the current working directory and this command will be removed from the user interface in the Client Module.

As specified in the RFC 959, the commands shown above would deal with the directory access of a given file system. Once the storage management and the access methods were automated by the system with LOGIN, SRCH and UPLD commands, the commands above would no longer be required, and should be disabled in the user interface so that users could not issue them directly. However, since the actual file management still required some of these commands to perform, their functions were still required and would be implemented in the system.



## **APPENDIX C**

### **DATABASE DESIGN AND SCHEMAS FOR TEACHING RESOURCES SHARING COMPONENT**

(Part of Thesis work by Geoffrey Y.K. Shea, “A Web-based Approach to the Integration of Diverse  
Data Sources for GIS”)

## C.1 Database Schemas for the TRSC

In the view to provide different level of access rights, the database would be designed to store user information and file information. The following list shows all the schemas to be used in the Teaching Resources Sharing Component (TRSC):-

- *acisms\_e\_department (department\_identity, department\_name)*
  - This schema would be used to store the department information, namely the department name and its identifying code in the Hong Kong Polytechnic University.
  - The field department\_name would be used to reflect the actual department in the Hong Kong Polytechnic University.
  - The field department\_identity would be used to show maintain the relationships between the curriculums, courses, staff, and students.
- *acisms\_r\_department\_curriculum (department\_identity, curriculum\_identity)*
  - This schema would be used to store the relationships between a department and the curriculums belongs to it.
- *acisms\_r\_department\_staff (department\_identity, staff\_identity)*
  - This schema would be used to store the relationships between a department and the staff of to it.
- *acisms\_r\_department\_course (department\_identity, course\_identity)*
  - This schema would be used to store the relationships between the department and the course it offered. The main distinction between the schema 2 is that a department may have same curriculums which were offered same or different courses.
- *acisms\_r\_department\_student (department\_identity, student\_identity)*
  - This schema would be used to store the relationships between the department and the students of it.
- *acisms\_e\_curriculum (curriculum\_identity, curriculum\_name)*
  - This schema would be used to store the curriculum information, namely its name and the unique curriculum code used in the Hong Kong Polytechnic University.
- *acisms\_e\_staff (staff\_identity, staff\_name, title, status, user\_identity)*
  - This schema would be used to store the staff information, namely the name, the title, and the status (like visiting staff, temporary staff, and so on), of a given teaching staff member.
  - The field user\_identity would be used to provide a relationship between the user information and the system information.
- *acisms\_e\_course (course\_identity, code, course\_name)*

- This schema would be used to store the course information, namely its code, and its name.
  - The field `course_identity` would be used to provide system-wide identity to distinguish different courses. This would reduce the dependency on the course code since the course code could be mutable by some senior staff member.
- *acisms\_e\_student* (***student\_identity***, *student\_name*, *sex*, *user\_identity*)
- This schema would be used to store the student information, namely the student identity, the name, and sex of a given student, studying in the Hong Kong Polytechnic University and taking some courses offered by this department.
  - The field `user_identity` would be used to provide a relationship between the user information and the system information.
- *acisms\_r\_curriculum\_course* (***curriculum\_identity***, ***course\_identity***, *academic\_year*, *subject\_type*)
- This schema would be used to store the relationships between curriculums and the courses being offered by the curriculum. In the feasibility study, it was found that a course could be offered by two different curriculums and a curriculum could offer two different courses. As time would pass, which could give changes to some courses and hence its name. The field `academic_year` would be used to distinguish different courses being offered at different times.
- *acisms\_r\_curriculum\_student* (***curriculum\_identity***, ***student\_identity***, *academic\_year*, *study\_year*)
- This schema would be used to store the relationships between curriculums and the students belonging to a given curriculum. In the feasibility study, it was found that a student could study for example a undergraduate degree and a postgraduate degree within a few years. As a result, changes of the student identity and the curriculum identity at different times could be made here.
- *acisms\_r\_staff\_course* (***curriculum\_identity***, ***course\_identity***, ***staff\_identity***, *academic\_year*, *remarks*)
- This schema would be used to store the relationships among the curriculum, the courses, and the teaching staff. Mainly, this schema would focus on the storage of the relationships between the courses being held by different staff. In the feasibility study, it was found that in the Hong Kong Polytechnic University, same staff could held the same course of same course code of different curriculums. Without this schema, such kind of relationships could not be maintained.
- *acisms\_r\_student\_course* (***student\_identity***, ***course\_identity***, *academic\_year*)
- This schema would be used to store the relationships between students and the courses taken by the students. In the feasibility study, it was found that a student could take different courses and different students could take the same given course. As the relationships between students and course could be different at different times, the field `academic_year` would be used to indicate such differences.
- *acisms\_r\_file\_course* (***file\_identity***, ***course\_identity***)

- This schema would be used to store the relationships between the shared files and the courses.
- *acisms\_e\_file (file\_identity, filename, location, useusername, usertype, last\_uploaded, description)*
- This schema would be used to store the information of all the shared files.
  - The field file\_identity would be used to provide system-wide identity to distinguish different shared files.
  - The field filename would be used to indicate the filename of a given shared file.
  - The field location would be used to indicate the physical path of the storage location of a given shared file. In this application, this field would be generated by the server based on each transaction.
  - The field useusername would be used to indicate the owner of the shared file. Only file owners and administrators could delete and update the file in a given specific location.
  - The field usertype would be used to indicate the types of users who could have the access rights of the given shared file.
  - The field last\_uploaded would be used to indicate the time at which the given shared file was stored in the file system of the server.
  - The field description would be used to indicate the searchable textual information of the given shared file.
- *acisms\_r\_file\_application (file\_identity, application\_identity)*
- This schema would be used to store the relationships between the shared files and the applications being used to develop the shared file or the applications being used to view the shared file.
- *acisms\_e\_application (application\_identity, user\_identity, vendor, application\_name, version)*
- This schema would be used to store the information like the vendor, the name, and version of the applications, being used to associate with the shared files.
  - The field user\_identity would be used to indicate the owner of a given application index.
  - The field application\_identity would be used to provide system-wide identity to distinguish different applications.
- *acisms\_e\_user (username, type)*
- This schema would be used to store the username and his/her user type in this system. Different users may have different user type. Based on this distinction, different shared files could have different type of access rights. In the current implementation, there would be 3 types, namely, guest, students and teaching staff.
- *acisms\_e\_usertype (type, description)*
- This schema would be used to store the description and its representing symbol.

- *acisms\_r\_file\_subject (file\_identity, subject\_identity)*
  - This schema would be used to store the relationship between the shared files and the subjects.
  
- *acisms\_e\_subject\_identity (subject\_identity, user\_identity, text)*
  - This schema would be used to store the information about the subject.
  - The field *subject\_identity* would be used to distinguish different subjects.
  - The field *user\_identity* would be used to indicate the owner of this subject.
  - The field *text* would be used to store the name of the subject.
  
- *acisms\_r\_username\_user (user\_identity, usertype)*
  - This schema would be used to store the user information namely the user identity and the user type of a given user.

## **C.2 Database Design**

The relationships in the TRSC database system is shown in Figure C-1. The relationships related to user information is shown in Figure C-2 and the relationships related to file information is shown in Figure C-3.

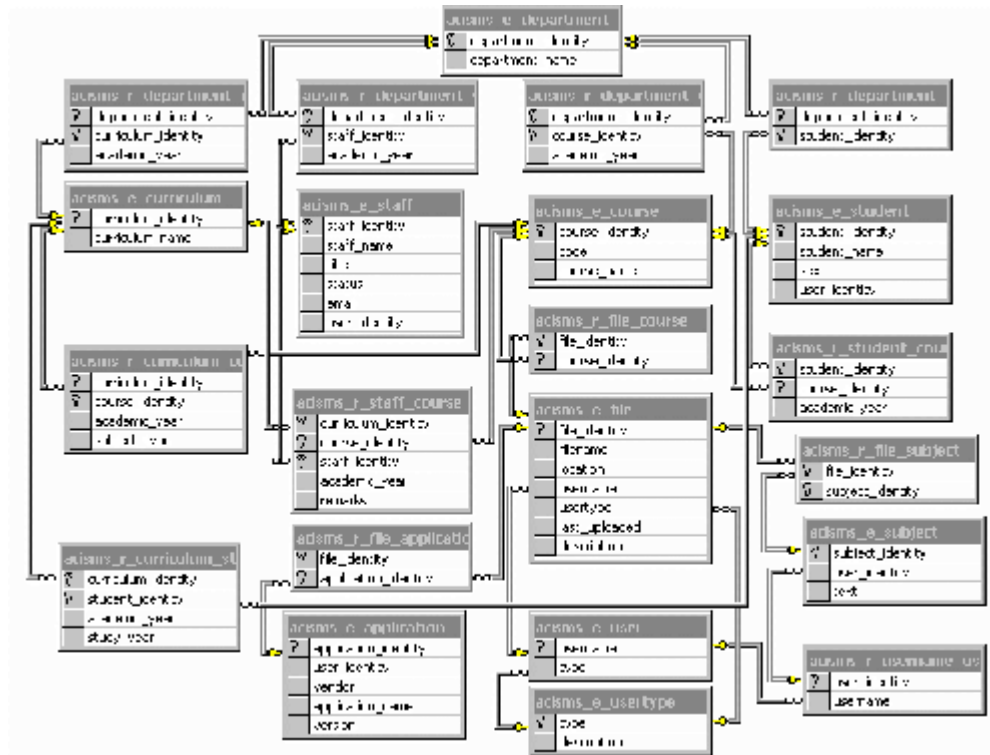


Figure C-1 The TRSC system database.

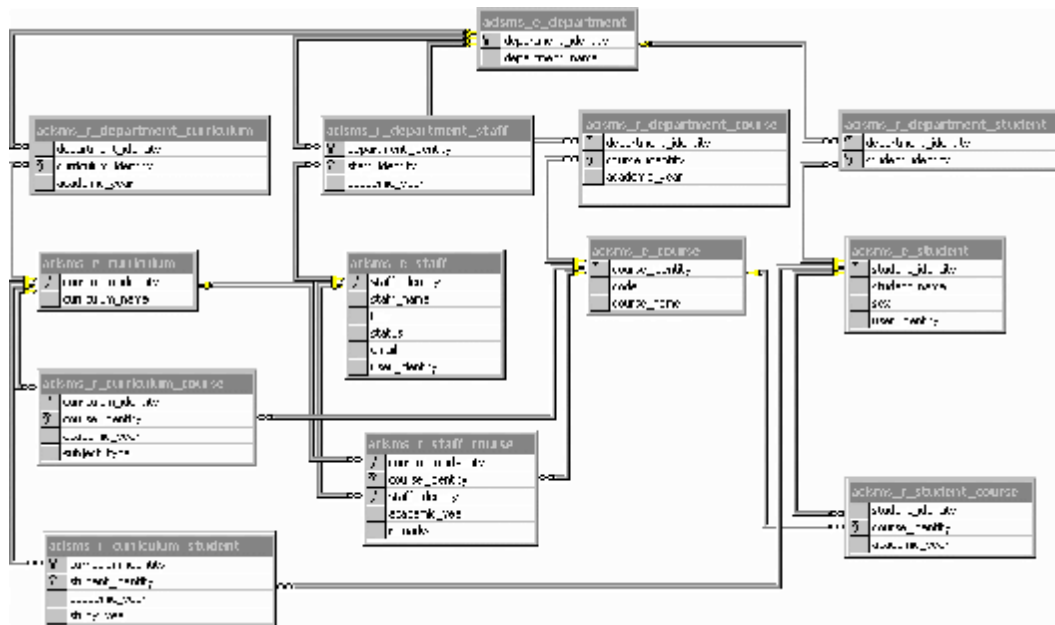


Figure C-2 TRSC database related to user information.

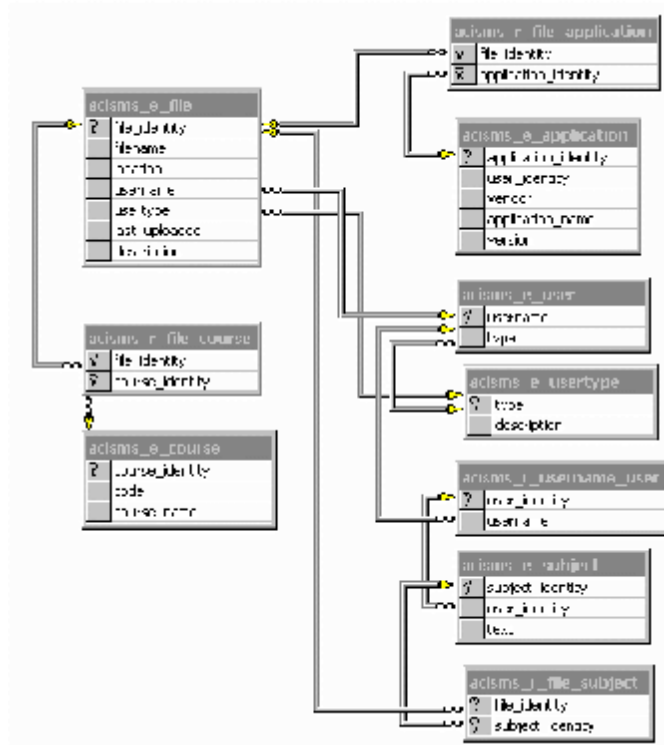


Figure C-3 TRSC database related to file information.

## **APPENDIX D**

### **INDEX FILE DEFINITIONS FOR TEACHING RESOURCES**

#### **SHARING COMPONENT**

(Part of Thesis work by Geoffrey Y.K. Shea, “A Web-based Approach to the Integration of Diverse  
Data Sources for GIS”)



## D.1 Introduction

To allow a file to be stored, users could still assign relationships associated to files. These relationships, originally embedded in the directory names of a file system using some pieces of keywords, was then extracted, and defined as "Indices" in this application. Since users should have the best knowledge to define the indices, most of the indices were defined by the users in a formatted way. After confirmation of the assignment, these indices would then be saved in a temporary file of a specified format, and then sent from the Client Module to the Server Module. This temporary file, in this application, was defined as "Index file". As a result, during uploading, two files would then be uploaded. One is the shared file and the other is the temporary index file associated to the shared file. The shared files would be sent first and the index file would be sent next. When these two files were received by the Server Module, the index file would be analyzed to obtain all the relationships previously assigned by the users. Next, a physical path in the remote file system would be formulated using the indices associated to the file. Afterwards, the Server Module would validate and then stored all the relationships, including the formulated path, in the database. All these indices would be used to facility the future file searches. In this application, the Client Module would be forbidden to manipulate the remote directories. All the directory management were done by the Server Module. File uploading was completed up to this point.

To allow a file to be retrieved, users could assign the searching criteria based on the indices. Since every file has some indices associated with it, and these indices were assumed to contain meaningful relationships among the files and could be reused among the users. As a result, file retrieval would become efficient while file management would become automatic. After confirmation, a list of filenames which satisfying the criteria would be generated by the Server Module. The Server Module could make queries to the database to identify if any shared files could have indices satisfying the searching criteria. For any shared files satisfying the criteria, their filenames would be listed and then stored in a temporary file. Then this file would be sent to the Client Module. The Client Module would then present the list and display the search result in the graphical user interface. The list would be used by users to determine which files, satisfying the searching criteria, could be downloaded or deleted. This could save a lot of time for users with or without prior knowledge about the indices.

To enable a file to be downloaded, users could select any items in the search result list. After confirmation, the selected items would be sent from the Server Module to the Client Module.

To enable a file to be deleted, the Server Module would check if the user were the file creator. A file could only be deleted either if he/she was defined to be the administrator, or if the user issuing the request were the file creator of that file.

In effect, four index files would be required in the current design: Profile for Login; Index File for Uploading; Index File for Searching (Request); and Index File for Searching (Reply).

Each of the four index files contains sufficient information to provide seamless file transfer service in a multi-user environment. The format of all the index files was designed to be in ASCII for efficient generation and transfer. For efficient parsing in the system and flexible maintenance in the file format and hence the protocol, each index file were constructed from a number field-tuple pairs. A field-tuples pair was formed by a field and a number of tuples to be used in a transfer. A field was a string to represent the name of a given field, which was prefixed by the symbol '[' and suffixed by the symbol and ']'. A tuple was string to represent a vector. The string value of each element in the vector was extracted and each of them was separated by a comma to form the string. The tuple string would be parsed by the system to obtain the meaning of each element, including the data type and its value. The data type of each element in the tuple was pre-defined and static.

The definition of all these index files would be described in the subsequent sections. A typical example of each of these index files are shown in Figures D-1 to D-4.

```

[UserIdentity] // words between '[' & '[' for system to parse fields
// used to identify user uniquely

[Username]
123456 // used to login system & access db

[Password]
3 // used to login server & access db

[UserType]
4 // used to control what user access on files

[UserDescription]
Teaching Staff // user grade

[UserDescription]
Mr. Braffery Kwon // user's greeting

[Users]
46,0518922 // user identity and username of other users
... // ... as realifers
86,049999 // "..." similar stuff deleted

[PredefinedApplications]
1,W,Microsoft,Word,97 // application identity, user identity,
... // vendor, application name, and version
8,0,Microsoft,ActiveX
... // DF predefined applications

[UserDefinedApplications]
9,SA,thhu,MathsImp,1.0
...
18,26,Sun,DR,1.2

[AllApplications]
9,SA,thhu,MathsImp,1.0
...
18,26,Sun,DR,1.2

[UserDefinedCourses]
1,AM07,Surveying & Mapping 1 // course identity, name, and num
...
94,LS1878,Engineering Surveying

[AllCourses]
1,AM07,Surveying & Mapping 1
...
94,LS1878,Engineering Surveying

[UserDefinedSubjects]
9,SA,thhu,thhu // subject identity, user identity, name
...
9,10,Programming

[AllSubjects]
96,364,STH1PH1_1000
...
A 11 Summary

```

Figure D-1 Profile for login.

```

[User] // words between '[' & '[' for system to parse
// user identity

[UserType] // user type for the course

[UserDescription] // descriptive description

[Applications]
1,1,MS,Paint,MS,Pro,5.0 // application identity,
// user identity,
// vendor,
// name,
// version
// require any further identity, must be in user defined

[Courses]
1 // course identity
2
3
4
5
6
7
8

[Subjects]
1,1,Programming // subject identity, user identity, name, and subject
2
3
4
5
6
7
8
9
// require any further identity, must be in user defined

```

Figure D-2 Index file for uploading.

```

[User] // words between '[' and '[' for system to parse fields
// Username

[User] // user identity & server at the shared file

[Application]
32/05/1999 // updated rate time

[Date]
11/05/2000 // updated rate time

[UserDescription]
Programming server // site server description

[Applications]
2 // application identity

[Courses] // course table file

[Subjects] // subject identity

```

Figure D-3 Index File for Searching (Request)

```

[User] // words between '[' and '[' for system to parse fields
209,oknotes1.doc.11.05/10/1503,2004SE // file identity, Username,
807,oknotes2.doc.11.05/11/1503,2004 // user identity,
923,oknotes3.doc.11.05/15/1503,2004 // last updated rate, file step

```

Figure D-4 Index File for Searching (Reply)

## D.2 Profile for Login

In the current design, there were 14 field-tuples pairs. The definition of each field-tuples were listed in Table D-1.

Field name	No. of expected tuples	Meaning
1. UserIdentity	1	User identity of a given user
2. Username	1	Username of a given user
3. Password	1	Password of a given user
4. UserType	1	User type of a given user
5. UserTypeDescription	1	User type description to be used in the interface in a session
6. UserDescription	1	User description to be used in the interface in a session
7. Users	$\geq 0$	All users defined in the system.
8. PreDefinedApplications	$\geq 0$	All applications defined by the administrator
9. UserDefinedApplications	$\geq 0$	All applications defined by the users other than administrator
10. AllApplications	$\geq 0$	All applications defined in the system.
11. UserRelatedCourses	$\geq 0$	All courses related to a given user.  If the user is a staff, this field specifies all the courses he/she teaches.  If the user is a student, this field specifies all the courses he/she takes.
12. AllCourses	$\geq 0$	All courses defined in the system.
13. UserDefinedSubjects	$\geq 0$	All subjects defined by a given user.
14. AllSubjects	$\geq 0$	All subjects defined in the system

Table D-1a Profile for login - fields definition.

Field name	No. of elements in a tuple	Meaning of each element in a tuple	Data type of each element in a tuple
1. UserIdentity	1	User identity of the given user	int
2. Username	1	Username of the given user	string
3. Password	1	Password of the given user	string
4. UserType	1	User type of the given user	string
5. UserTypeDescription	1	User type description of the user	string
6. UserDescription	1	User description of the user	string
7. Users	2	User identity of a user	int
		Username with the identity	string
8. PreDefinedApplications	5	Application identity of an given application	int
		User identity of a user who define the application	int
		Vendor of the application	string
		Name of the application	string
		Version of the application	string

9. UserDefinedApplications	5	Application identity of an given application	int
		User identity of a user who define the application	int
		Vendor of the application	string
		Name of the application	string
		Version of the application	string
10. AllApplications	5	Application identity of an given application	int
		User identity of a user who define the application	int
		Vendor of the application	string
		Name of the application	string
		Version of the application	string
11. UserRelatedCourses	3	Course identity of a given course	int
		Code of the course	string
		Name of the course	string
12. AllCourses	3	Course identity of a given course	int
		Code of the course	string
		Name of the course	string
13. UserDefined Subjects	3	Course identity of a given course	int
		Code of the course	string
		Name of the course	string
14. AllSubjects	3	Course identity of a given course	int
		Code of the course	string
		Name of the course	string

Table D-1b Profile for login - tuple definition.

### D.3 Index File for Uploading

In the current design, there were 5 field-tuples pairs. The definition of each field-tuples were listed in Table D-2.

Field name	No. of expected tuples	Meaning
1. File	1	The file to be shared
2. Description	0 or 1	Content description of the shared file.
3. Application	1	Application associated to (used to create/open) the shared file
4. Courses	>= 0	Courses associated to the shared file.
5. Subjects	>= 0	Subjects associated to the shared file.

Table D-2a Definition of the index file for uploading - fields definition.

Field name	No. of elements in a tuple	Meaning of each element in a tuple	Data type of each element in a tuple
1. File	3	Filename of the shared file	int
		User identity of a user who share the given file.	string
		User type of a given file	string
2. Description	1	Content description of the shared file.	string
3. Application	1 or 5	Application identity of an given application	int
		User identity of a user who define the application (not required when application was defined in the system)	int
		Vendor of the application (not required when application was defined in the system)	string
		Name of the application (not required when application was defined in the system)	string
		Version of the application (not required when application was defined in the system)	string
4. Courses	1	Course identity defined in the system	int
5. Subjects	1 or 3	Subject identity of a given subject	int
		User identity (not required when subject was defined in the system)	string
		Subject name (not required when subject was defined in the system)	string

Table D-2b Definition of the index file for uploading - tuple definition.

#### D.4 Index File for Searching (Request)

In the current design, there were 8 filed-tuples pairs. The definition of each filed-tuples were listed in Table D-3.

Field name	No. of expected tuples	Meaning
1. Filename	0 or 1	The substring of the filename of the expected shared file.
2. User	0 or 1	Owner of the expected shared file
3. DataFrom	0 or 1	Date after which the expected shared file was uploaded.
4. DateTo	0 or 1	Date before which the expected shared file was uploaded
5. Description	0 or 1	Content description to the expected shared files
6. Application	0 or 1	Application associated to the expected shared files
7. Courses	>= 0	Courses associated to the expected shared files
8. Subjects	>= 0	Subjects associated to the expected shared files

Table D-3a Definition of the index file for searching (Request) - fields definition.

Field name	No. of elements in a tuple	Meaning of each element in a tuple	Data type of each element in a tuple
1. Filename	1	The substring of the filename of the expected shared file.	string
2. User	1	User identity of a given user	int
3. DataFrom	1	Date in dd/mm/yyyy format after which the expected shared file was uploaded.	date
4. DateTo	1	Date in dd/mm/yyyy format before which the expected shared file was uploaded	date
5. Description	1	Substring of content description associated to the shared files.	int
6. Application	1	Application identity of an application associated to the shared files	int
7. Courses	1	Course identity of a course associated to the shared files	int
8. Subjects	1	Subject identity of a subject associated to the shared files	int

Table D-3b Definition of the index file for searching (Request) - tuple definition.

#### D.5 Index File for Searching (Reply)

In the current design, there were 1 field-tuples pair. The definition of each field-tuples were listed in Table D-4.

Field name	No. of expected tuples	Meaning
Filename	>= 0	File information meeting the searching criteria.

Table D-4a Definition of the index file for searching (reply) - fields definition.

Field name	No. of elements in a tuple	Meaning of each element in a tuple	Data type of each element in a tuple
1. Filename	5	File identity of a shared file	int
		Filename of the given shared file.	string
		User identity of the given shared file.	int
		Date on which the shared file had been uploaded recently.	date
		Size of the given shared file.	int

Table D-4b Definition of the index file for searching (reply) - tuple definition.

**APPENDIX E**

**WALKTHROUGH AND SCREEN SHOTS OF THE**

**MAP DATA RETRIEVAL COMPONENT**

(Part of Thesis work by Geoffrey Y.K. Shea, "A Web-based Approach to the Integration of Diverse  
Data Sources for GIS")

## E.1 Introduction

A walkthrough showing the operation of MDRC was presented in this appendix. Several screen captures and the corresponding description were used to illustrate the walkthrough.

A Web browser capable of frame display and support JavaScript version 1.1 was required to access the MDRC. This MDRC was targeted at version 4.0 or above of the two main browsers that support JavaScript: Netscape Navigator and Microsoft Internet Explorer. There might exist small discrepancies for display on the two browsers on some occasions, but these discrepancies would not affect the overall operations.

User could start the MDRC from any browser that has already set up the Internet connection by entering the following address:

<http://www.lsgi.polyu.edu.hk/staff/Geoffrey.Shea/projectME/indexMDRC.htm>

or

<http://www.lsgi.polyu.edu.hk/staff/Geoffrey.Shea/projectME/>

## E.2 Starting MDRC

The client browser received and displayed the server reply as shown in Figure E-1. User could go to the index map of Hong Kong by clicking the “Goto Hong Kong Map” push button or cancel the action simply clicking the “Cancel” push button. Since the MDRC was basically built around *Autodesk MapGuide* software package, therefore, the *MapGuide Viewer* (*Netscape Navigator* Plug-in) was required and could be downloaded freely from *Autodesk* site by clicking the small icon next to the “Cancel” push button.

The main page of this project was shown in Figure E-2. The structure of MDRC was shown in Figure E-3. Basically, the main page was divided into 4 frames and the upper left frame was designated to be the main frame for displaying map layers. The lower left frame was designed for displaying map attributes and report. The upper right and lower right frames were designed for housing menus.

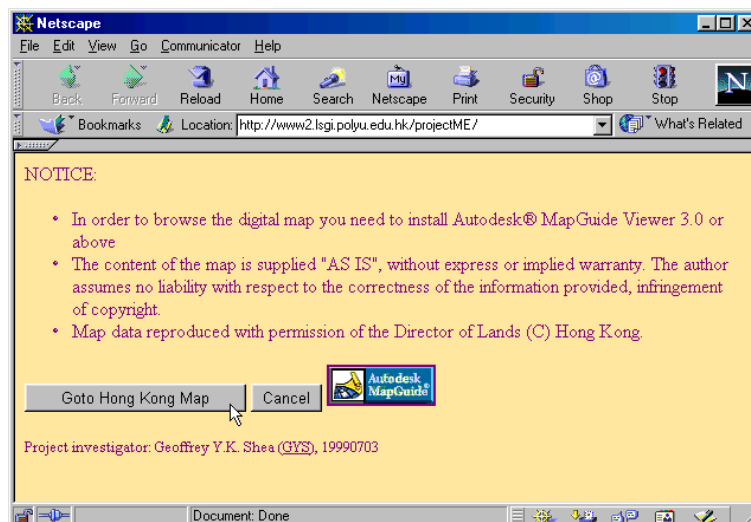


Figure E-1 Information page of MDRC.



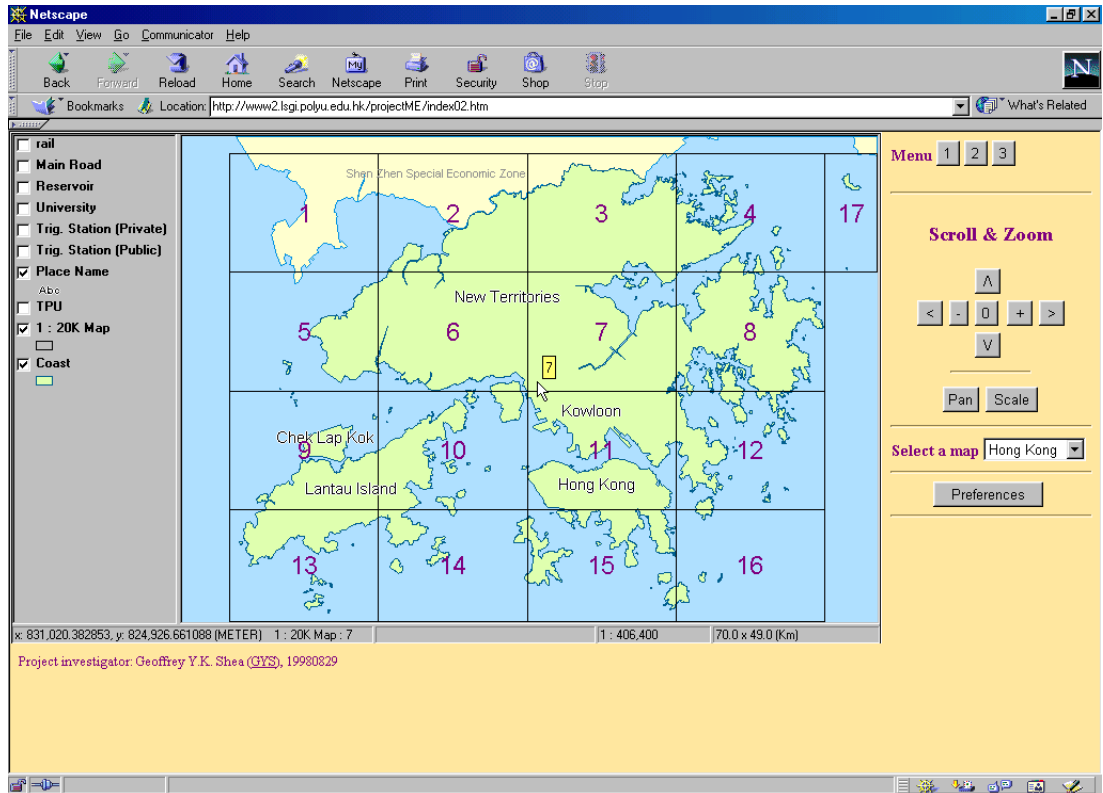


Figure E-2 Main page of MDRC.

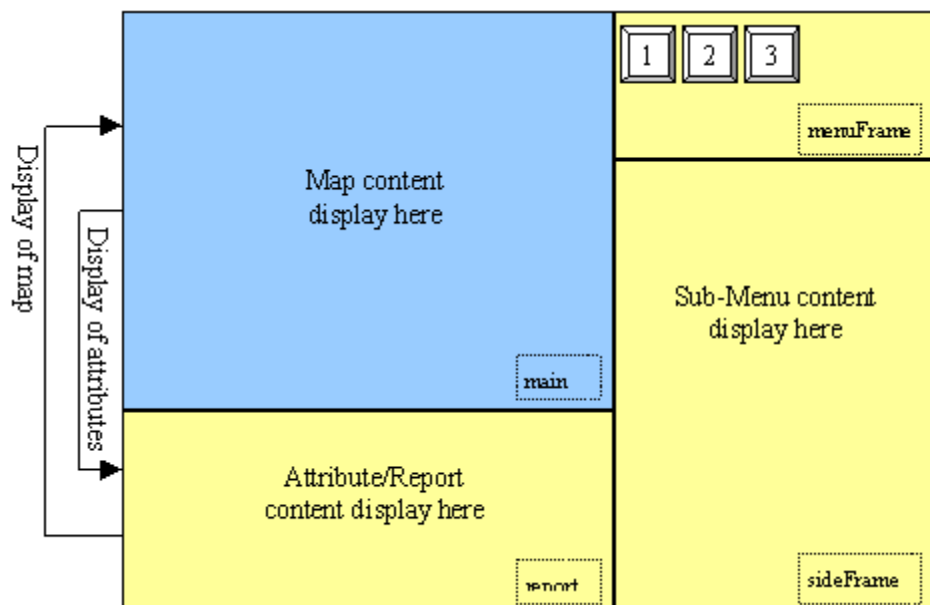


Figure E-3 Structure of MDRC main page.

### E.3 Using the Legend to Display Map Layers Selectively

The legend was displayed on the left side of the main frame. The legend was used to indicate the names of all displayable map layers dynamically. The number of map layers to be displayed on the legend was controlled by map author who published the map. A typical example of map legend was shown in Figure E-4. To select a layer for display simply click the checkbox next to the layer name. To deselect a layer just un-check the checkbox. For example, Figure E-5 illustrated the “rail” layer was displayed in association with 3 other layers.

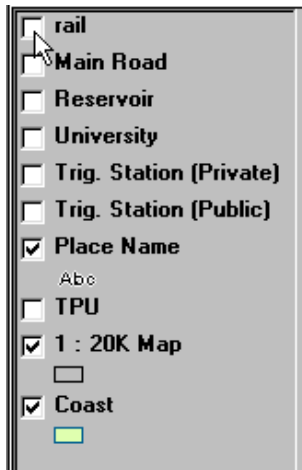


Figure E-4 The legend.

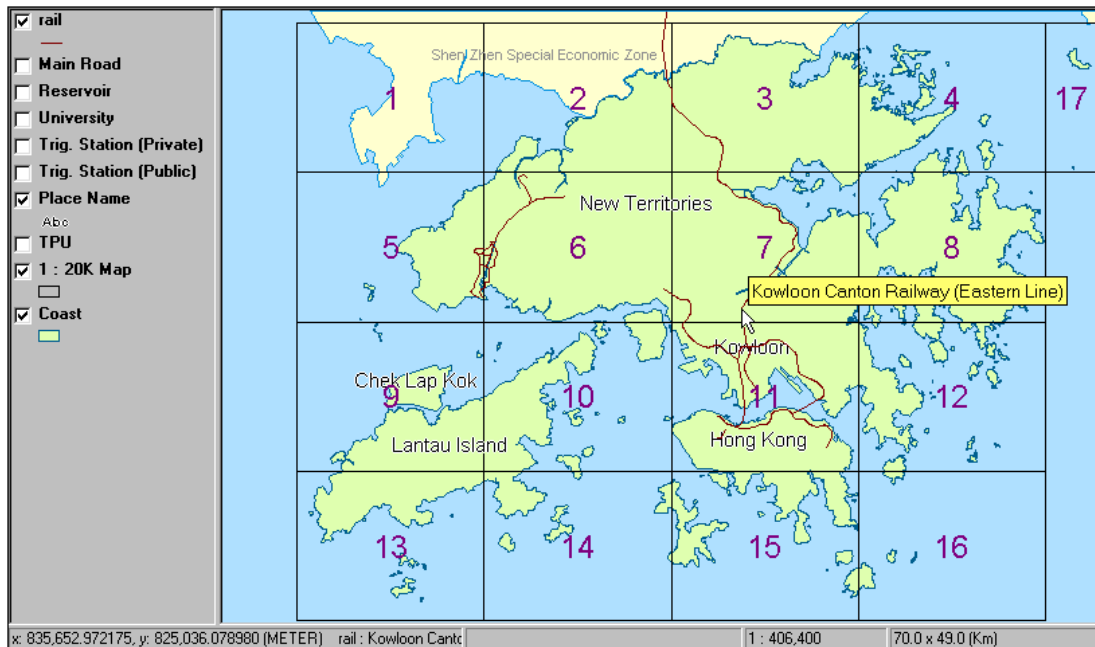


Figure E-5 Using legend to display map layers selectively.

## E.4 Using the Map Window Popup Menu

The map window popup menu was used to provide quick access to all functions available for users. To display the map window popup menu simply right-click the mouse anywhere on the map window. A typical popup menu was shown in Figure E-6. A subset of these functions were programmed as frame-based selection menus to be displayed on the lower right frame. Sub-menu page 1 and page 2 were shown in Figure E-7 and E-8 respectively.

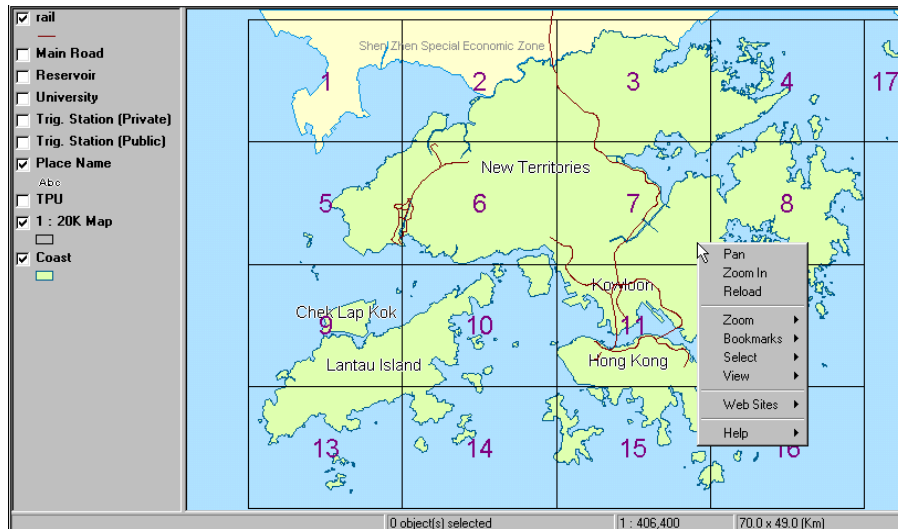


Figure E-6 A typical example of map window popup menu.

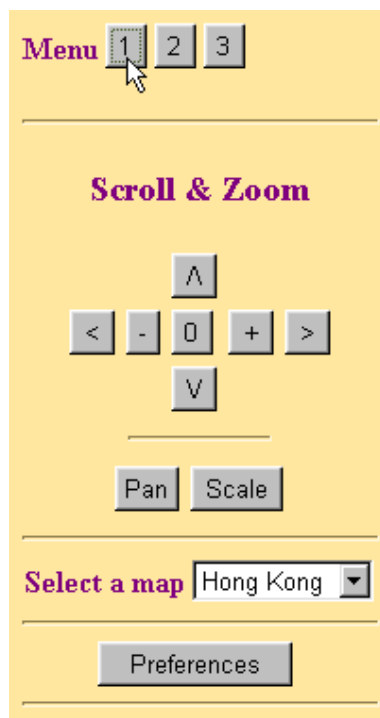


Figure E-7 Sub-menu page 1.

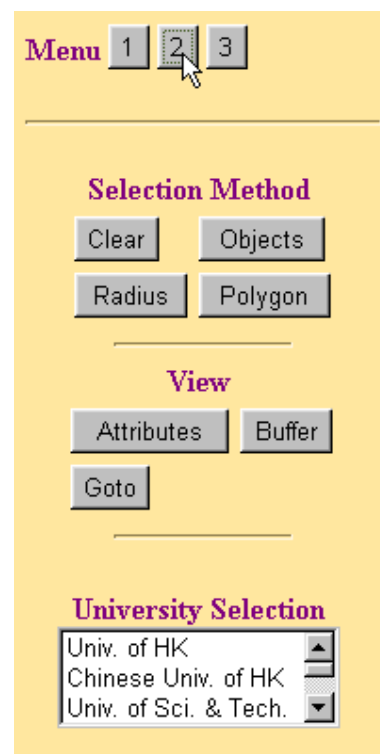


Figure E-8 Sub-menu page 2.

## E.5 Displaying Information About Map Objects

The status bar located at the bottom of the upper right frame was intended to display the coordinates of the current pointer and the current map scale, width, and height. When the mouse cursor was pointed to a map object, the status bar would display the object's layer and name. The layer name was listed first, followed by the object name. Additionally, MapGuide Viewer would also display the object name in a yellow popup that would be disappeared when pointer was moved away from the map object.

Figure E-5 illustrated the pointer was sitting over a map object named "Kowloon Canton Railway (Eastern Line)" and that belonged to "rail" layer. The current display map scale was 1:406,400 and covering an area of 70.0 x 49.0 Km<sup>2</sup>.

## E.6 Moving Around a Map

Users could move around a map using the zoom and pan commands provided by the map window popup menu. Commands related to zoom and pan functions were listed as follows (also shown in Figure E-9):

- + Pan
- + Zoom
  - ++ Zoom In
  - ++ Zoom Out
  - ++ Zoom Previous
  - ++ UnZoom
  - ++ Zoom Scale
  - ++ Zoom Width



Figure E-9 List of zoom functions.

## E.7 Querying the Map

### (A) Measuring Distances

The run-length distance on the map could be measured. To invoke the measure distance command just choose “View” and “View Distance” from the map window popup menu. Figure E-10 illustrated the run-length distance between three points.

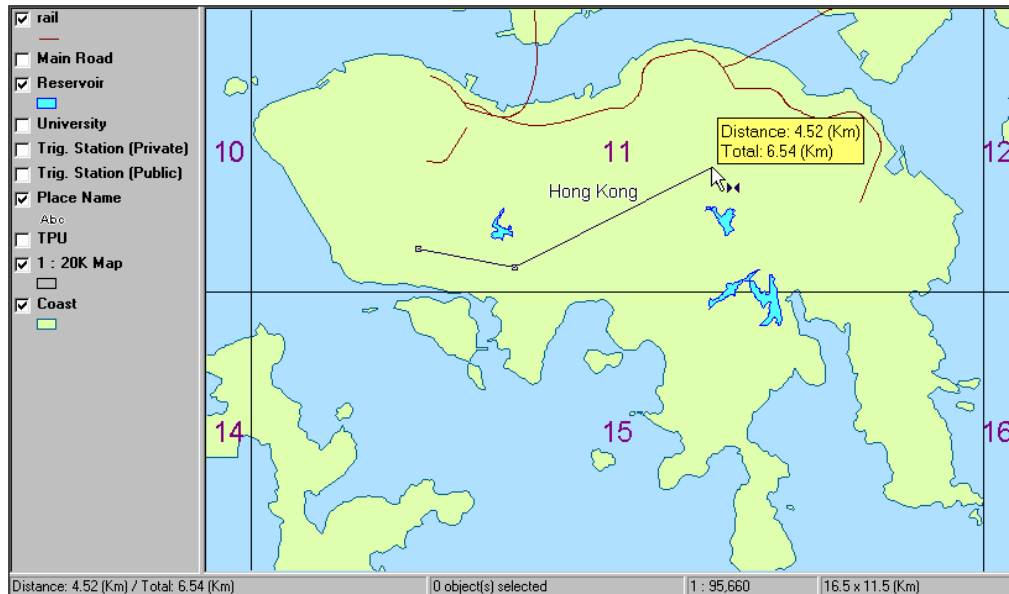


Figure E-10 A measuring distance example.

### (B) Selecting Map Objects

Users had to select map objects when they wanted to generate reports or perform zone buffering. Users could select map objects in several different ways:

- Select Map Objects Using the Mouse
- Select Map Objects by Radius
- Select Map Objects by Polygon

Figure E-11 illustrated an example of map objects selection by polygon.

### (C) Generate Reports on Selected Objects

Once the map objects were selected by any one of the above-mentioned methods, a report of the selected map objects could be displayed. To view a report choose “View” and “View Report...” from the map window popup menu. The newly generated report would be displayed in the lower left frame. An example showing the report of the selected Trig. Stations was shown in Figure E-12.

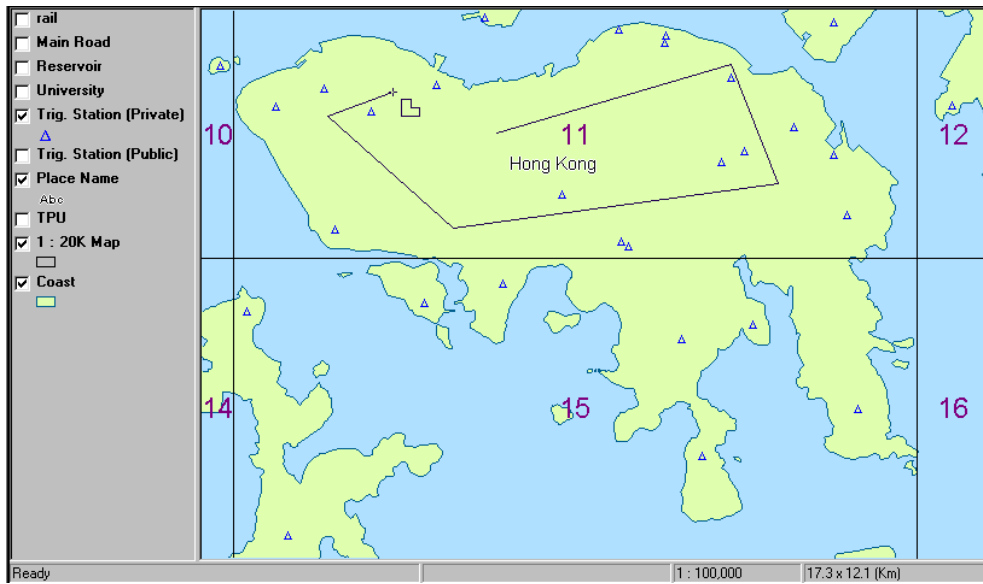


Figure E-11 Map objects selection by polygon.

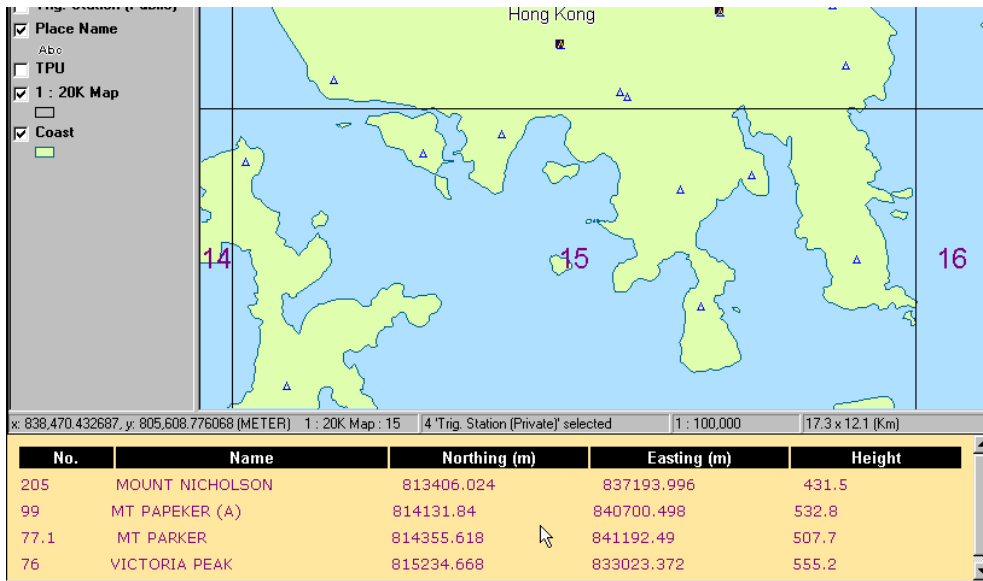


Figure E-12 Report on the selected map objects.

#### (D) Buffering Selected Objects

Other than a report that could be generated from the selected map objects, zone buffering could also be performed on the selected map objects. Zone buffering was applicable to either point objects, line objects or areal objects. A variety of choices were provided for user to define the buffering properties such as hatch patten, color, line style, etc. Figure E-13 showed the available buffering properties and Figure E-14 was an example of the buffering result.

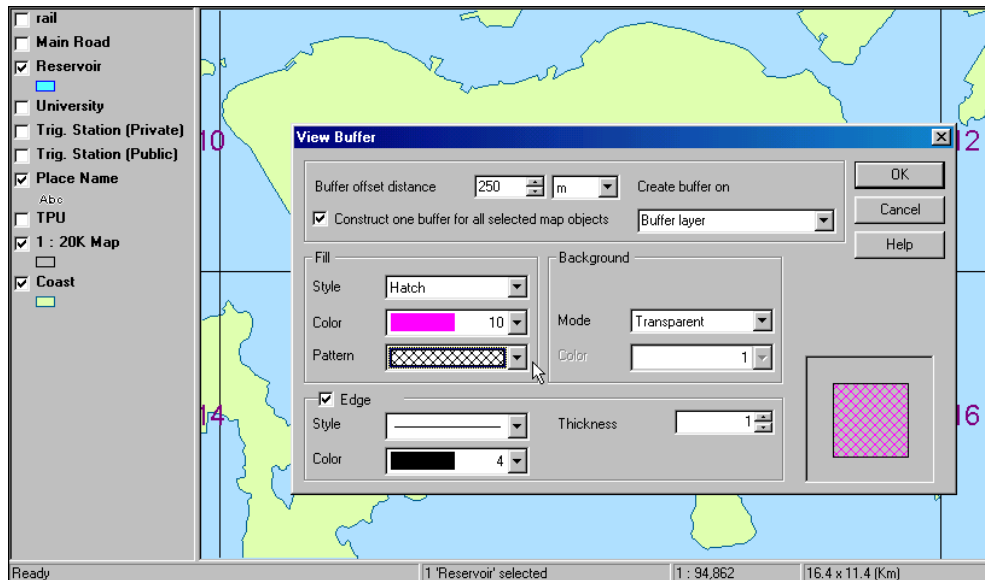


Figure E-13 Zone buffering properties.

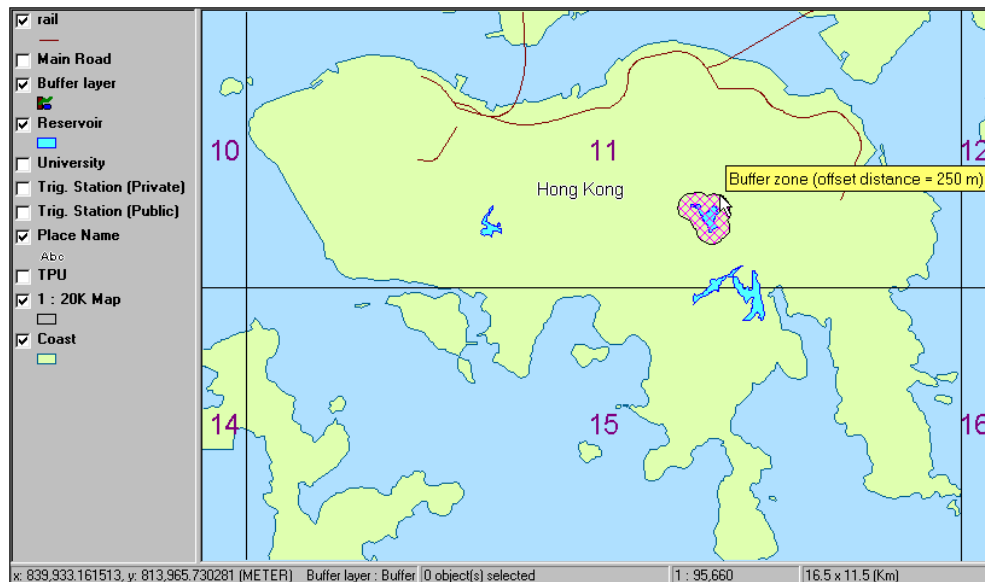
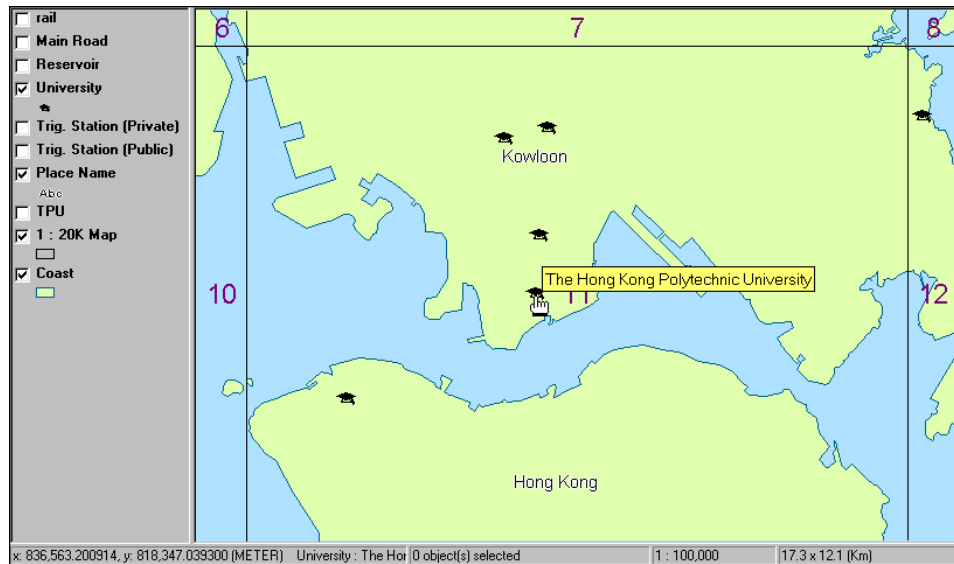


Figure E-14 Result of zone buffering.

## E.8 Miscellaneous Functions

### (A) Hyperlinked Map Objects

Every map object could be hyperlinked and directly go to the pre-defined location when map object was double clicked. The mouse cursor changed to the normal hand icon when the cursor was sitting on a hyperlinked object (see Figure E-15 for example).



### (B) Implementing Security Measures

It was possible to set up security measure on map layers limiting the data access for selected group or users. Figure E-16 was an example showing the access to “Trig. Station (Private)” layer was granted only if the correct username and password were entered.

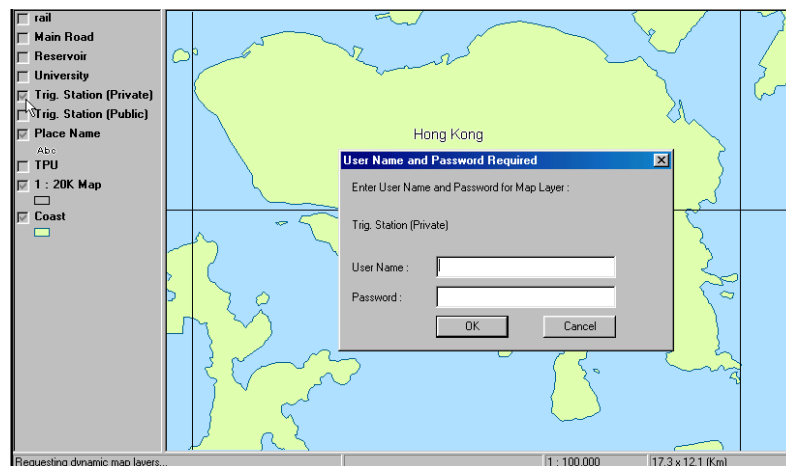


Figure E-16 Logon window for username/password protected layer.



(C) Different Display Symbology with the Same Data Set

Under the careful design of map author, the same set of map data could be displayed differently using different symbology to represent the same set of data at different scales. Figures E-17 and E-18 demonstrated that two line styles are used to represent the same data set at scales 1:20000 and 1:10000 respectively.

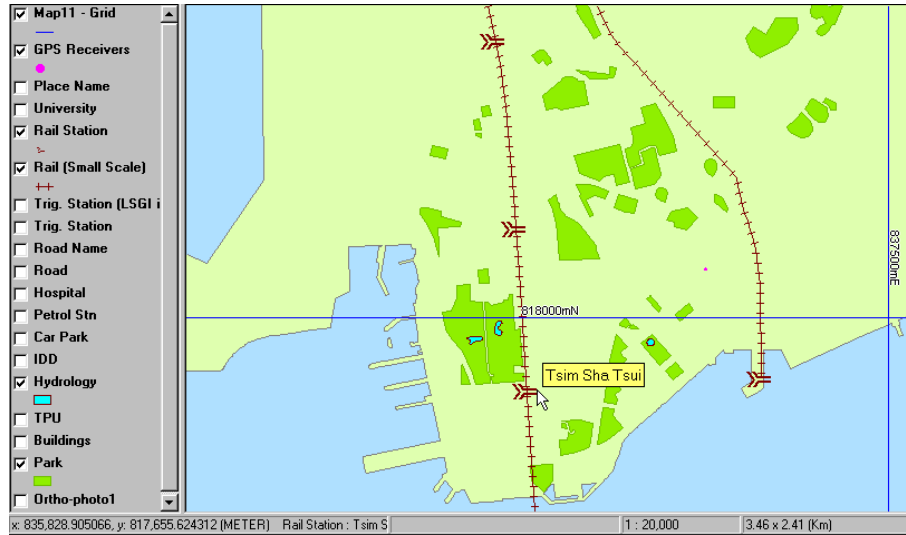


Figure E-17 Line style used for Rail layer at 1:20000.



Figure E-18 Line style used for Rail layer at 1:10000.

(D) Simultaneous Display of Raster and Vector

Raster data was considered as a map layer in *Autodesk MapGuide* allowing user to display the raster map and vector map simultaneously. The MapGuide Viewer commands were applicable to both raster and vector map as well. Figure E-19 demonstrated the display of raster and vector map together and Figure E-20 was a closer look of the same set of data.

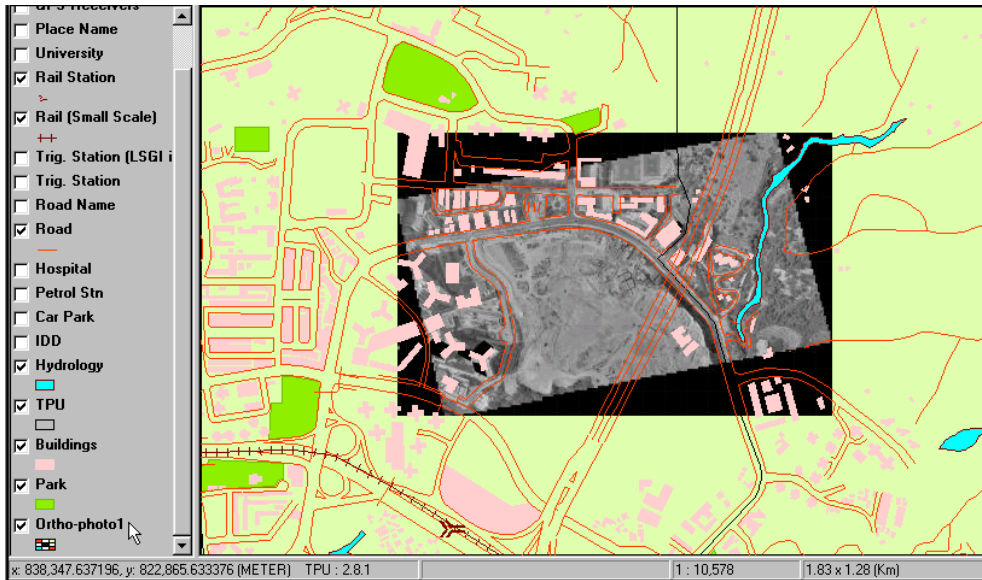


Figure E-19 Simultaneous display of raster and vector map.

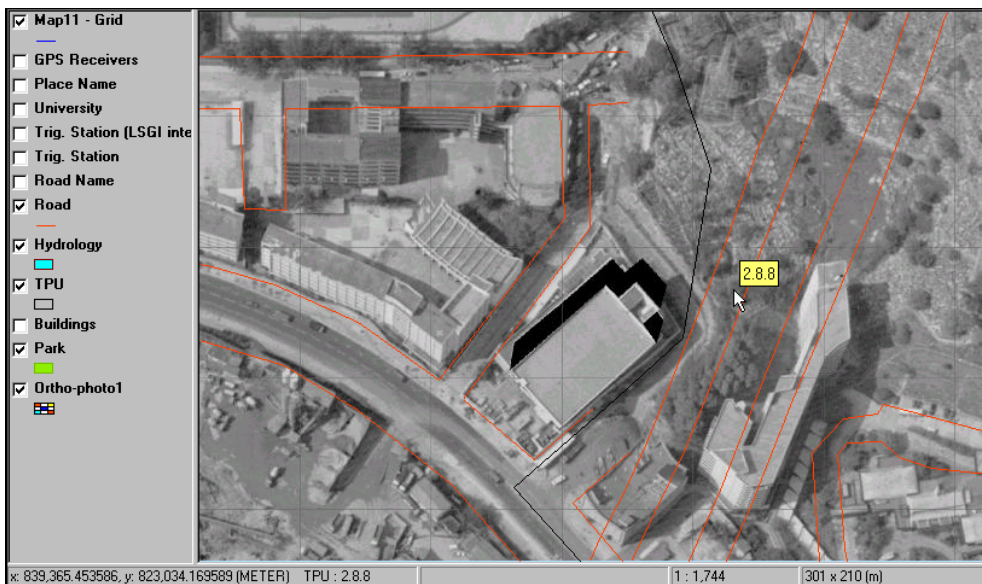


Figure E-20 A closer look on the raster and vector map.

## **APPENDIX F**

### **WALKTHROUGH AND SCREEN SHOTS OF THE WEB**

### **MAPPING WITH SVG DEMONSTRATION**

(Part of Thesis work by Geoffrey Y.K. Shea, “A Web-based Approach to the Integration of  
Diverse Data Sources for GIS”)

## F.1 Introduction

A walkthrough showing the Web Mapping with SVG Demonstration was presented in this appendix. Several screen captures and the corresponding description were used to illustrate the walkthrough.

A Web browser capable of frame display and support JavaScript version 1.1 was required to access the demonstration. This demonstration was targeted at *Netscape* version 4.7 or above or *Microsoft Internet Explorer* version 5.5 or above. There might exist small discrepancies for display on the two browsers on some occasions, but these discrepancies would not affect the overall operations.

User could start the demonstration from any browser that has already set up the Internet connection by entering the following address:

<http://www.lsgi.polyu.edu.hk/staff/Geoffrey.Shea/projectME/indexSVGWebMapping.htm>

or

<http://www.lsgi.polyu.edu.hk/staff/Geoffrey.Shea/projectME/>

## F.2 Starting Web Mapping with SVG Demonstration

The client browser received and displayed the server reply as shown in Figure F-1. User could go to the index map of Hong Kong by clicking either the “Full” or “Partial” push button or cancel the action simply clicking the “Cancel” push button. Since SVG Web Mapping was basically built around SVG specification (at the time of developing this Web page, the W3C Proposed Recommendation 19 July 2001, W3C Scalable Vector Graphics (SVG) 1.0 Specification was used), therefore, a Web browser plugin capable of displaying SVG graphics such as *Adobe SVG Viewer* 2.0 was required and could be downloaded freely from Adobe site by clicking the small animated icon.

The main page of this SVG Web Mapping Demonstration was shown in Figure F-2. The structure of SVG Web Mapping Demonstration was shown in Figure F-3. Basically, the main page was divided into 3 frames and the upper left frame is designated to be the main frame for displaying map data. The lower frame was designed for displaying map attributes and report. The upper right frame was designed for housing legend.

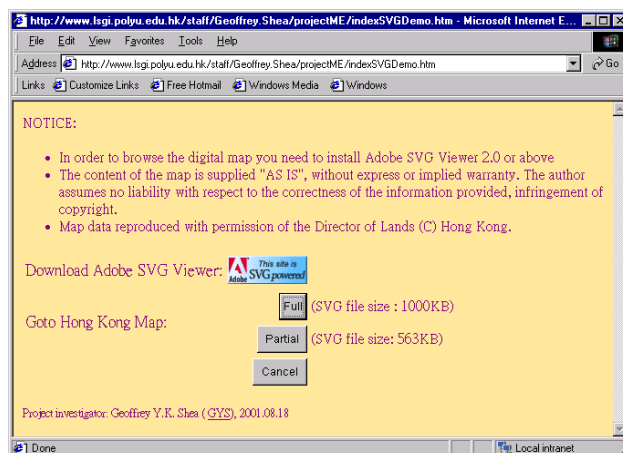


Figure F-1 Information page of SVG Web Mapping.

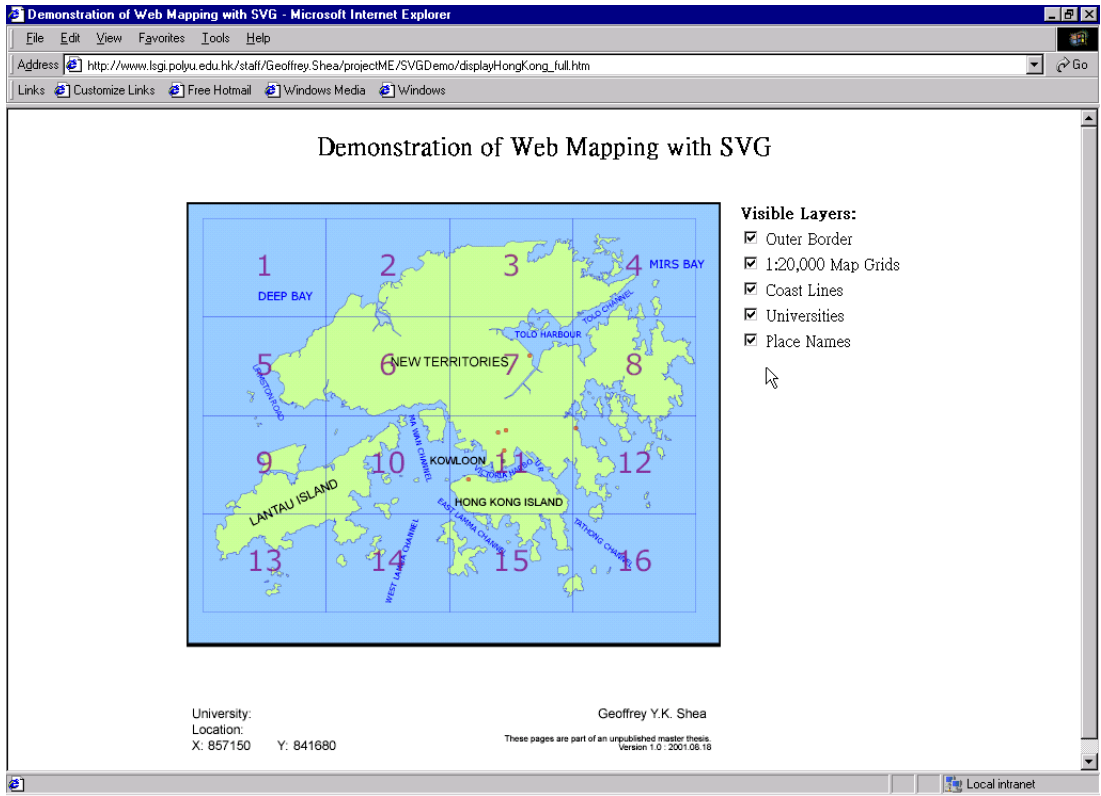


Figure F-2 Main page of SVG web mapping demonstration.

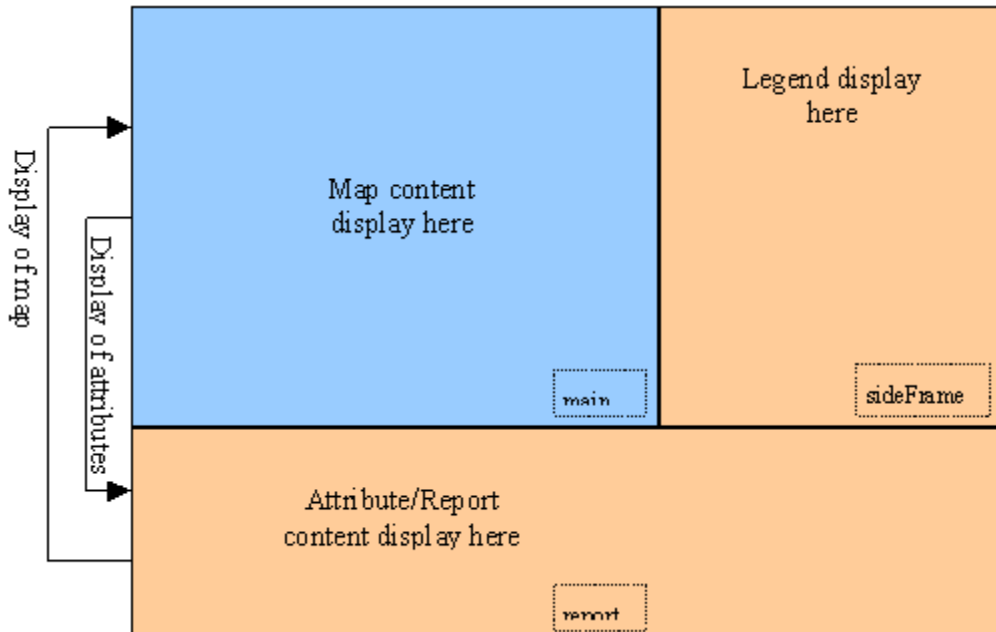


Figure F-3 Structure of SVG web mapping demonstration.

### F.3 Using the Legend to Display Map Layers Selectively

The legend appeared on the right side of the main page. The legend was used to indicate the names of all displayable map layers dynamically. The number of map layers to be displayed on the legend was controlled by map author who published the map. A typical example of map legend was shown in Figure F-4. To select a layer for display simply click the checkbox next to the layer name. To deselect a layer just un-check the checkbox. For example, Figure F-5 illustrated the “Coast Lines” layer was displayed in association with two other layers.

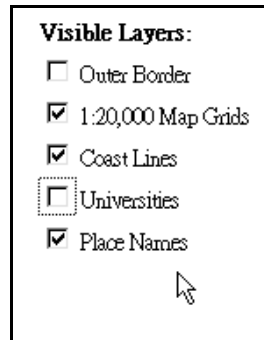


Figure F-4 The legend.

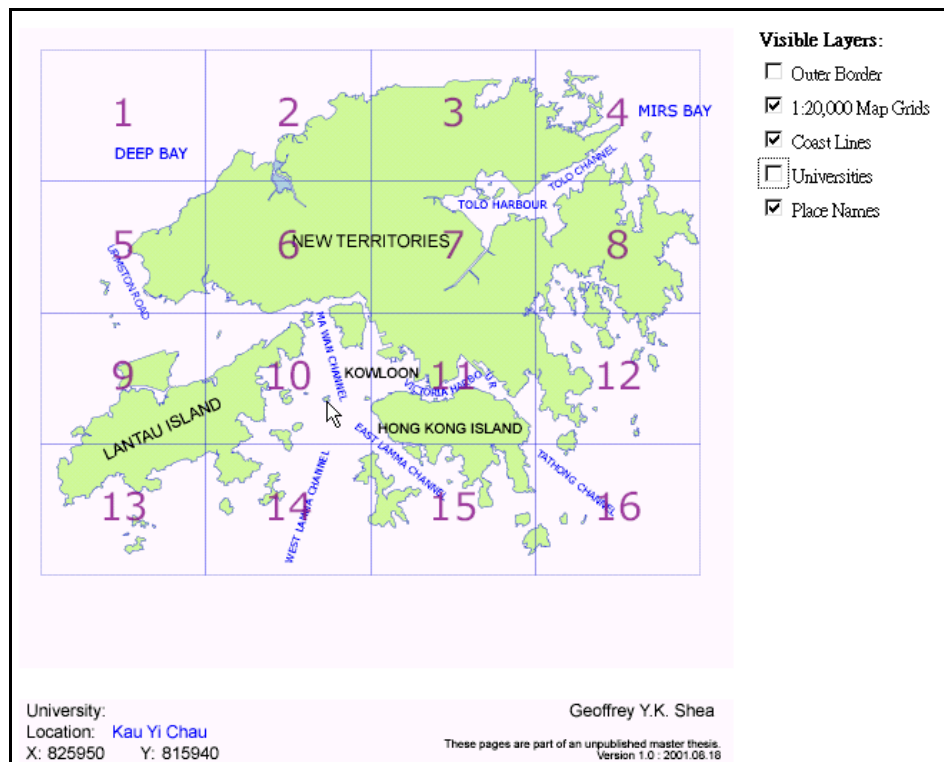


Figure F-5 Using legend to display map layers selectively.

#### F.4 Using the Map Window Popup Menu

The map window popup menu was used to provide quick access to functions available for users. To display the map window popup menu simply right-click the mouse anywhere on the map window. A typical popup menu was shown in Figure F-6.

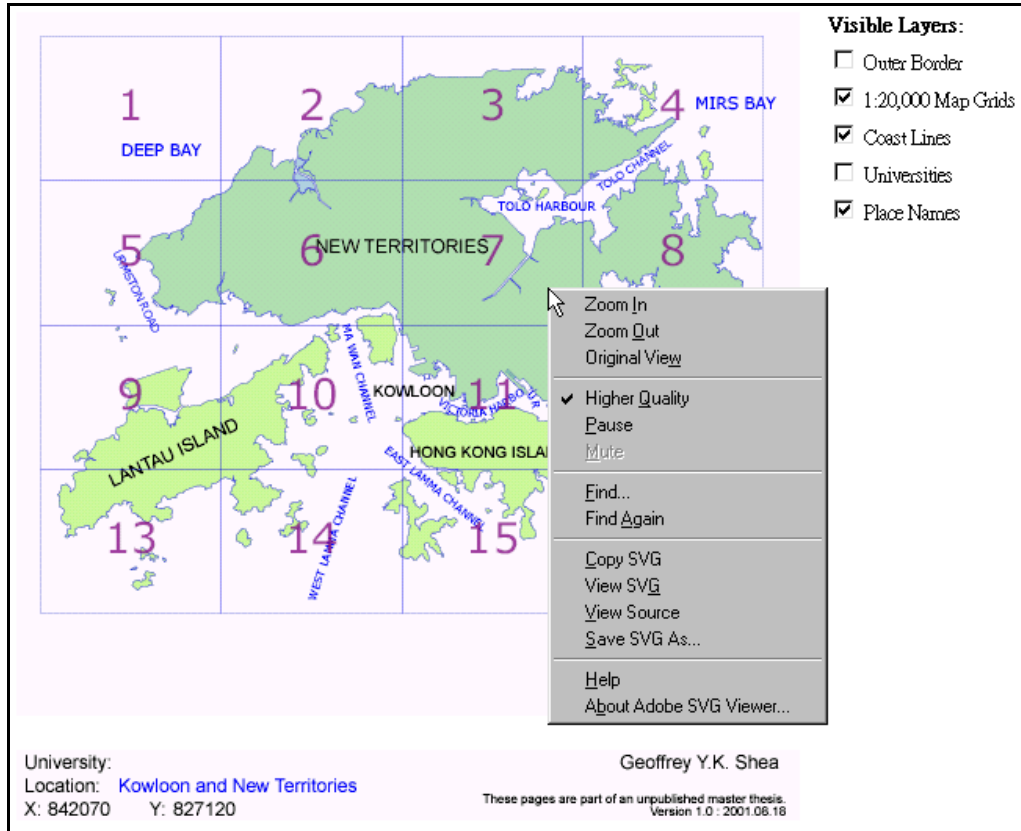


Figure F-6 A typical example of map window popup menu.

#### F.5 Displaying Information About Map Objects

The display panel located at the bottom of the window was used to display the coordinates of the current pointer and the ID of the map object that the pointer located. A fly-out text information would be displayed when pointer was moved over the red circle.

Figure F-7 illustrated the pointer was sitting over a map object named “The Chinese University of Hong Kong” and that belonged to “Universities” layer.

Figure F-8 illustrated the mouse cursor was sitting over a map object named “Lamma Island” and that belonged to “Coast Lines” layer. The map object being pointed was highlighted by different color. The “hand” pointer icon meant a hyperlink was attached to the symbol.



Figure F-7 A fly-out information about the map object.

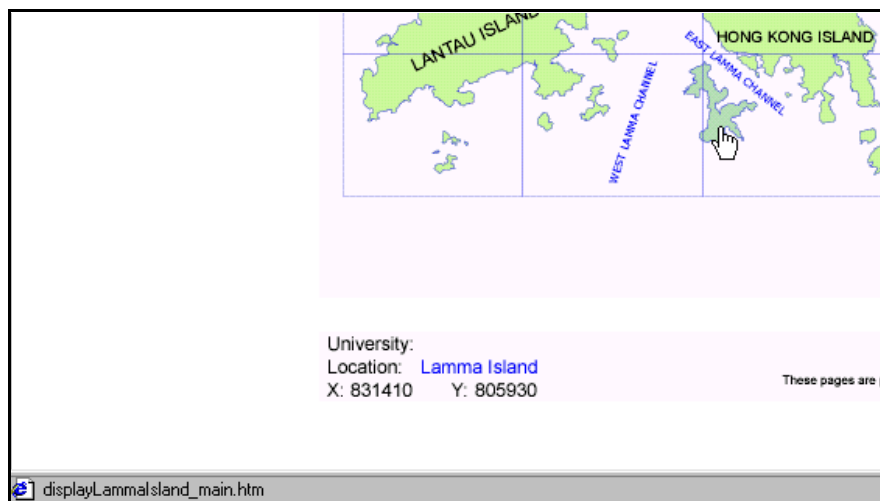


Figure F-8 A hyperlinked object.

## F.6 Moving Around a Map

Users could move around a map using the zoom and pan commands provided by the Adobe SVG Viewer. To achieve a zoom in command simply pressing the “Ctrl” button while the pointer was located on the map window. Pressing “Shift” and “Ctrl” buttons simultaneously would zoom out the map window. A pan action was achieved by pressing the “Alt” button.

## F.7 Miscellaneous Functions

### (A) Hyperlinked Map Objects

Every map object could be hyperlinked and directly go to the pre-defined location when map object was clicked. The mouse cursor would change to the normal hand icon when the cursor was sitting on a hyperlinked object (see Figures F-7 and F-8 for example). SVG was using *XLink* for all link definitions.



(B) Flexible Text Annotation

Since SVG was rendering text elements like other graphic elements, therefore, text annotation could be rendered either in a horizontal direction, vertical direction, arbitrary orientation, or even along a curved path. Figure F-9 below demonstrated the flexible text annotation function provided by SVG.

(C) Layer Opacity

Since SVG viewers simulated the “painters algorithm”, which meant that overlapping areas are “painted over” or according to opacity values, therefore, the rendering sequence was strictly according to element (object) sequence within the SVG file. Figure F-9 illustrated various usage of the opacity value on elements and groups.

(D) Animation

Since SVG supporting the ability to change vector graphics over time, animated effects could be achieved. Figure F-9 illustrated a simple flashing red circle was established to attract the attention of user. The flashing effect was achieved by changing the fill color from black to white over a period of 1 second.



Figure F-9 Text annotation on a curved path, layer opacity, and animation.

## **APPENDIX G**

### **TECHNICAL DESCRIPTION OF SVG WEB MAPPING**

#### **METHODOLOGY**

(Part of Thesis work by Geoffrey Y.K. Shea, “A Web-based Approach to the Integration of  
Diverse Data Sources for GIS”)

## G.1 Introduction

A detailed description explaining the SVG Web Mapping methodology was presented in this appendix. Several screen captures and the corresponding description were used to illustrate the various aspects of SVG specification.

A Web browser capable of frame display and support JavaScript version 1.1 was required to access the demonstration. This demonstration was targeted at *Netscape* version 4.7 or above or *Microsoft Internet Explorer* version 5.5 or above. There might exist small discrepancies for display on the two browsers on some occasions, but these discrepancies would not affect the overall operations. Since at the time of this writing (August 2001) the two commonly available browsers – Netscape Navigator and Internet Explorer – did not support direct display of SVG file, therefore, a web browser plugin such as Adobe SVG Viewer was required.

User could start the online examples from any browser that has already set up the Internet connection by entering the following address:

<http://www.lsgi.polyu.edu.hk/staff/Geoffrey.Shea/projectME/indexSVGExamples.htm>

or

<http://www.lsgi.polyu.edu.hk/Geoffrey.Shea/projectME/>

## G.2 Starting the Technical Description of SVG Web Mapping Methodology

The main page of this example site was shown in Figure F-1. The main page was divided into 4 frames with the title located at the top frame. All the available explanations about SVG elements were listed on the middle left frame. All the SVG files were displayed on the middle right frame. The corresponding SVG source code was listed on the bottom right frame.

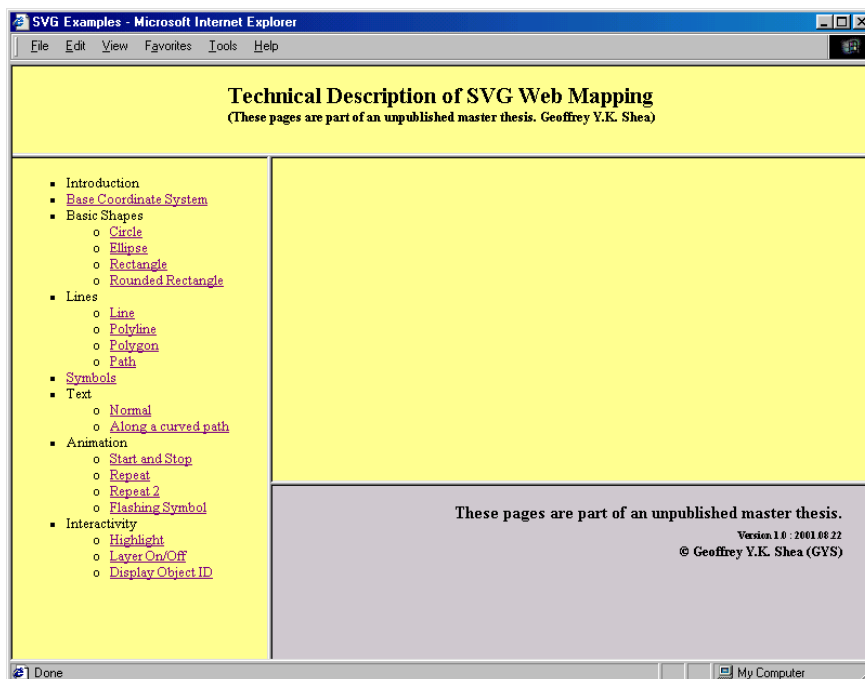
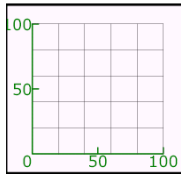


Figure G-1 Main page of technical description of SVG web mapping

### G.3 Technical Description of SVG Methodology



Setting origin at the lower left corner of the user coordinate system.

```

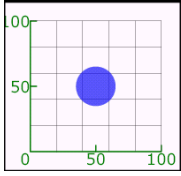
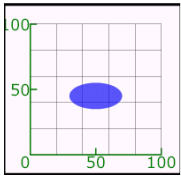
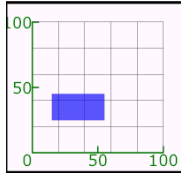
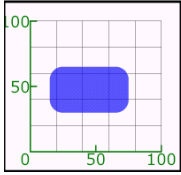
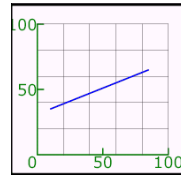
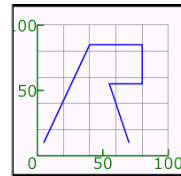
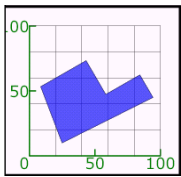
<svg width="100" height="100" viewBox="-20 -15 135 130">
<defs>
  <style type="text/css">
    <![CDATA[
      .nameFrame {font-family:Verdana;font-size:12;fill:green;}
    ]]>
  </style>
</defs>

<g id="OriginLowerLeft" transform="matrix(1 0 0 -1 0 100)">
<g id="coordinateFrameAndText">
  <rect x="-20" y="-15" width="135" height="130" stroke-width="2" fill="none"/>
  <line x1="0" y1="0" x2="100" y2="0" stroke="green"/>
  <line x1="0" y1="0" x2="0" y2="100" stroke="green"/>
  <line x1="50" y1="0" x2="50" y2="5" stroke="green"/>
  <line x1="0" y1="50" x2="5" y2="50" stroke="green"/>
  <line x1="100" y1="0" x2="100" y2="5" stroke="green"/>
  <line x1="0" y1="100" x2="5" y2="100" stroke="green"/>
  <line x1="20" y1="0" x2="20" y2="100" stroke="grey" stroke-width="0.2"/>
  <line x1="40" y1="0" x2="40" y2="100" stroke="grey" stroke-width="0.2"/>
  <line x1="60" y1="0" x2="60" y2="100" stroke="grey" stroke-width="0.2"/>
  <line x1="80" y1="0" x2="80" y2="100" stroke="grey" stroke-width="0.2"/>
  <line x1="100" y1="0" x2="100" y2="100" stroke="grey" stroke-width="0.2"/>
  <line x1="0" y1="20" x2="100" y2="20" stroke="grey" stroke-width="0.2"/>
  <line x1="0" y1="40" x2="100" y2="40" stroke="grey" stroke-width="0.2"/>
  <line x1="0" y1="60" x2="100" y2="60" stroke="grey" stroke-width="0.2"/>
  <line x1="0" y1="80" x2="100" y2="80" stroke="grey" stroke-width="0.2"/>
  <line x1="0" y1="100" x2="100" y2="100" stroke="grey" stroke-width="0.2"/>

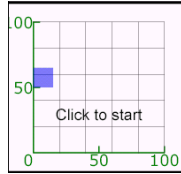
  <text class="nameFrame" x="0" y="-10" transform="matrix(1 0 0 -1 0 -20)"
    text-anchor="end">0</text>
  <text class="nameFrame" x="50" y="-10" transform="matrix(1 0 0 -1 0 -20)"
    text-anchor="middle">50</text>
  <text class="nameFrame" x="100" y="-10" transform="matrix(1 0 0 -1 0 -20)"
    text-anchor="middle">100</text>
  <text class="nameFrame" x="0" y="45" transform="matrix(1 0 0 -1 0 90)"
    text-anchor="end">50</text>
  <text class="nameFrame" x="0" y="95" transform="matrix(1 0 0 -1 0 190)"
    text-anchor="end">100</text>
</g>
<!-- ===== -->
<!-- === *****=== ***** === ***** -->
<!-- ===== -->

<!-- ===== -->
<!-- === *****=== ***** === ***** -->
<!-- ===== -->
</g>
</svg>

```

	<pre>&lt;circle   cx="50"   cy="50"   r="15"   fill="blue"   fill-opacity="0.65" /&gt;</pre>
	<pre>&lt;ellipse   cx="50"   cy="45"   rx="20"   ry="10"   fill="blue"   fill-opacity="0.65" /&gt;</pre>
	<pre>&lt;rect x="15" y="25"   width="40"   height="20"   fill="blue"   fill-opacity="0.65" /&gt;</pre>
	<pre>&lt;rect   x="15"   y="30"   width="60"   height="35"   rx="10"   fill="blue"   fill-opacity="0.65" /&gt;</pre>
	<pre>&lt;line   x1="10"   y1="35"   x2="85"   y2="65"   stroke="blue" /&gt;</pre>
	<pre>&lt;polyline   fill="none"   stroke="blue"   points="5,10 40,85 80,85 80,55 55,55 70,10" /&gt;</pre>
	<pre>&lt;polygon style="fill:blue; fill-opacity:0.65;   stroke:black; stroke-width:0.2"   points="25.00 10.00           94.28 45.00           84.28 62.32           58.30 47.32           43.30 73.30           8.66 53.30" /&gt;</pre>

	<pre>&lt;path   d="M5,5 C5,85 85,85 85,5"   style="stroke: blue; fill: none" /&gt;</pre>
	<pre>&lt;defs&gt;   &lt;symbol id="University" overflow="visible"&gt;     &lt;circle       cx="0"       cy="0"       r="15"       style="fill:blue;         fill-opacity:0.65;         stroke:black;         stroke-width:0.5"     /&gt;   &lt;/symbol&gt; &lt;/defs&gt;  &lt;use x="45" y="35" xlink:href="#University"/&gt; &lt;use x="65" y="50" xlink:href="#University"/&gt;</pre>
	<pre>&lt;circle cx="20" cy="20" r="1"   fill="black" fill-opacity="0.65"/&gt; &lt;circle cx="50" cy="40" r="1"   fill="black" fill-opacity="0.65"/&gt; &lt;circle cx="80" cy="20" r="1"   fill="black" fill-opacity="0.65"/&gt;  &lt;text x="20" y="20"   transform="matrix(1 0 0 -1 0 40)"   style="fill:blue;text-anchor:start" &gt;Left&lt;/text&gt; &lt;text x="50" y="40"   transform="matrix(1 0 0 -1 0 80)"   style="fill:blue;text-anchor:middle" &gt;Center&lt;/text&gt; &lt;text x="80" y="20"   transform="matrix(1 0 0 -1 0 40)"   style="fill:blue;text-anchor:end" &gt;Right&lt;/text&gt;</pre>
	<pre>&lt;defs&gt;   &lt;path id="curvedPath" d="M5,5 C5,85 85,85 85,5"/&gt; &lt;/defs&gt;  &lt;use xlink:href="#curvedPath"   fill="none" stroke="grey"/&gt; &lt;text style="fill:blue;text-anchor:start"&gt;   &lt;textPath xlink:href="#curvedPath"   &gt;Text on a curved path   &lt;/textPath&gt; &lt;/text&gt;</pre>



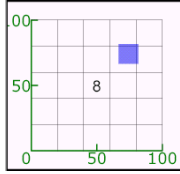
```

<script type="text/javascript">
  <![CDATA[
    function advanceToRight (evt, message)
    {
      var SVGDoc =
        evt.getTarget().getOwnerDocument();
      var text =
        SVGDoc.getElementById("text").getFirstChild();
      text.setData(message);
    }
  ]]>
</script>

<rect y="50"
  width="15"
  height="15"
  fill="blue"
  fill-opacity="0.5"
  onbegin="advanceToRight (evt, 'Running')"
  onend="advanceToRight (evt, 'Click to start')">
  <animate
    attributeName="x"
    values="0;85"
    begin="click"
    dur="3s"
  />
</rect>

<text id="text"
  x="50"
  y="25"
  transform="matrix(1 0 0 -1 0 50)"
  style="text-anchor: middle"
>Click to start</text>

```



```

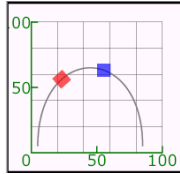
<script type="text/javascript">
  <![CDATA[
    var count = 0;
    function advanceRepeatedly(evt)
    {
      var SVGDoc =
        evt.getTarget().getOwnerDocument();
      var text =
        SVGDoc.getElementById("text").getFirstChild();
      text.setData(++count);
    }
  ]]>
</script>

<rect y="42.5"
  width="15"
  height="15"
  fill="blue"
  fill-opacity="0.5"
  onrepeat="advanceRepeatedly(evt)"
  >
  <animate
    attributeName="x"
    values="0;85"
    dur="5s"
    repeatCount="indefinite"
  />
  <animate
    attributeName="y"
    values="0;85"
    dur="5s"
    repeatCount="indefinite"
  />
</rect>

<text id="text"
  x="50"
  y="45"
  transform="matrix(1 0 0 -1 0 90)"
  style="text-anchor: middle"
  >0</text>

```





```

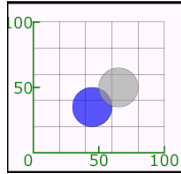
<script type="text/javascript">
  <![CDATA[
    var count = 0;
    function advanceRepeatedly(evt)
    {
      var SVGDoc =
    evt.getTarget().getOwnerDocument();
      var text =
    SVGDoc.getElementById("text").getFirstChild();
      text.setData(++count);
    }
  ]]>
</script>

<path
  d="M5,5 C5,85 85,85 85,5"
  style="stroke: grey; fill: none"
/>

<rect
  x="-5"
  y="-5"
  width="10"
  height="10"
  style="fill:red;fill-opacity:0.65"
>
  <animateMotion
    path="M5,5 C5,85 85,85 85,5"
    rotate="auto"
    dur="10s"
    repeatCount="indefinite"
  />
</rect>

<rect
  x="-5"
  y="-5"
  width="10"
  height="10"
  style="fill: blue;fill-opacity:0.65"
>
  <animateMotion
    path="M5,5 C5,85 85,85 85,5"
    dur="15s"
    repeatCount="indefinite"
  />
</rect>

```

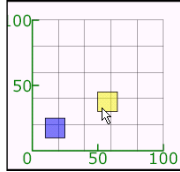


```

<defs>
  <symbol
    id="University"
    overflow="visible"
  >
    <circle
      cx="0"
      cy="0"
      r="15"
      style="fill:blue;
      fill-opacity:0.65;
      stroke:black;
      stroke-width:0.25"
    />
  </symbol>
  <symbol
    id="flashingUniversity"
    overflow="visible"
  >
    <circle
      cx="0"
      cy="0"
      r="15"
      style="fill:blue;
      fill-opacity:0.65;
      stroke:black;
      stroke-width:0.25;
      stroke-opacity:0.5"
    />
    <animate attributeName="fill"
      values="black;white"
      dur="1s"
      repeatCount="indefinite"
    />
  </circle>
</symbol>
</defs>

<use x="45" y="35"
  xlink:href="#University"
/>
<use x="65" y="50"
  xlink:href="#flashingUniversity"
/>

```

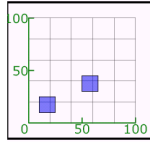


```

<script type="text/javascript">
  <![CDATA[
    function changeColorOver(evt)
    {
      var rect = evt.getTarget();
      rect.setAttribute("style", "fill:yellow;
        fill-opacity:0.5;
        stroke:black;
        stroke-width:0.5")
    }
    function changeColorOut(evt)
    {
      var rect = evt.getTarget();
      rect.setAttribute("style", "fill:blue;
        fill-opacity:0.5;
        stroke:black;
        stroke-width:0.5")
    }
  ]]>
</script>

<rect id="rectangle1"
  x="10"
  y="10"
  width="15"
  height="15"
  style="fill:blue;
    fill-opacity:0.5;
    stroke:black;
    stroke-width:0.5"
  onmouseover="changeColorOver(evt) "
  onmouseout ="changeColorOut(evt) "
/>
<rect id="rectangle2"
  x="50"
  y="30"
  width="15"
  height="15"
  style="fill:blue;
    fill-opacity:0.5;
    stroke:black;
    stroke-width:0.5"
  onmouseover="changeColorOver(evt) "
  onmouseout ="changeColorOut(evt) "
/>

```



Visible Layers:  
 Outer Border  
 Universities

### Highlight effect

2 files required:

- (a) HTML file to embed the SVG and create the reference between SVG and check boxes
- (b) SVG file containing the displayable graphic elements

#### HTML file to embed the SVG and create the reference

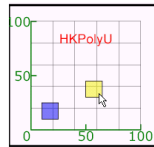
```
<HTML>
<HEAD>
<SCRIPT type="text/javascript" language="JavaScript1.2">
function display_layer (checkbox, element_id)
{
    var svgobj, svgstyle, svgdoc = document.HongKongMaps.getSVGDocument();
    // *** For each element, get the element's style object, then set
    // *** its visibility according to the state of the checkbox.
    svgobj = svgdoc.getElementById(element_id);
    svgstyle = svgobj.getStyle();
    if (!checkbox.checked) { svgstyle.setProperty('visibility', 'hidden'); }
    else { svgstyle.setProperty('visibility', 'visible'); }
}
</SCRIPT>
</HEAD>
<BODY>
<CENTER>
<P><TABLE BORDER="0" CELLPADDING="8" CELLSPACING="0">
  <TR><TD COLSPAN="2" align="center">
    <H4>Click the check box(es) to see the effect!</H4>
  </TD>
</TR>
<TR>
  <TD align="center" valign="top">
    <EMBED NAME="HongKongMaps" WIDTH="200" HEIGHT="200"
    SRC="42displayLayers.svg" type="image/svg+xml">
  </TD>
  <TD align="left" valign="top">
    <FORM NAME="display_form">
    <TABLE BORDER="0" CELLPADDING="0" CELLSPACING="2">
      <TR><TD><b>Visible Layers:</b></TD></TR>
      <TR>
        <TD><INPUT TYPE="checkbox" VALUE=""
          ONCLICK="display_layer(this, 'coordinateFrameAndText') ">Outer Border
        </TD>
      </TR>
      <TR>
        <TD><INPUT TYPE="checkbox" VALUE=""
          ONCLICK="display_layer(this, 'Universities') ">Universities
        </TD>
      </TR>
    </TABLE>
    <SCRIPT type="text/javascript" LANGUAGE="JavaScript1.2">
      for (var i = 0; i < document.display_form.elements.length; i++)
        if (document.display_form.elements[i].type == 'checkbox')
          document.display_form.elements[i].checked = true;
    </SCRIPT>
    </FORM>
  </TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

### SVG file containing the displayable graphic elements

```
<g id="Universities">
<script type="text/javascript">
  <![CDATA[
    function changeColorOver(evt)
    {
      var rect = evt.getTarget();
      rect.setAttribute("style",
        "fill:yellow;fill-opacity:0.5;stroke:black;stroke-width:0.5")
    }
    function changeColorOut(evt)
    {
      var rect = evt.getTarget();
      rect.setAttribute("style",
        "fill:blue;fill-opacity:0.5;stroke:black;stroke-width:0.5")
    }
  ]>
</script>

<rect id="rectangle1"
  x="10"
  y="10"
  width="15"
  height="15"
  style="fill:blue;
  fill-opacity:0.5;
  stroke:black;
  stroke-width:0.5"
  onmouseover="changeColorOver(evt) "
  onmouseout ="changeColorOut(evt) "
/>

<rect id="rectangle2"
  x="50"
  y="30"
  width="15"
  height="15"
  style="fill:blue;
  fill-opacity:0.5;
  stroke:black;
  stroke-width:0.5"
  onmouseover="changeColorOver(evt) "
  onmouseout ="changeColorOut(evt) "
/>
</g>
```



Visible Layers:  
 Outer Border  
 Universities

### Displaying Object ID

2 files were required to perform the effect:  
 HTML and SVG files

### SVG file containing the displayable graphic elements

```
<g id="Universities">
<script type="text/javascript">
  <![CDATA[
    function changeColorOver(evt)
    {
      var rect = evt.getTarget();
      rect.setAttribute("style",
        "fill:yellow;fill-opacity:0.5;stroke:black;stroke-width:0.5")
    }
    function changeColorOut(evt)
    {
      var rect = evt.getTarget();
      rect.setAttribute("style",
        "fill:blue;fill-opacity:0.5;stroke:black;stroke-width:0.5")
    }
  ]>
</script>
<rect id="HKPolyU"
  x="50"
  y="30"
  width="15"
  height="15"
  style="fill:blue;
    fill-opacity:0.5;
    stroke:black;
    stroke-width:0.5"
  onmouseover="me=evt.getTarget();showUniversity(me);changeColorOver(evt)"
  onmouseout="me=evt.getTarget();emptyUniversity();changeColorOut(evt)"
/>
<rect id="HKU"
  x="10"
  y="10"
  width="15"
  height="15"
  style="fill:blue;
    fill-opacity:0.5;
    stroke:black;
    stroke-width:0.5"
  onmouseover="me=evt.getTarget();showUniversity(me);changeColorOver(evt)"
  onmouseout="me=evt.getTarget();emptyUniversity();changeColorOut(evt)"
/>
</g>
<text id="fly_out_text"
  x="50"
  y="80"
  transform="matrix(1 0 0 -1 0 160)"
  style="fill:red"
  text-anchor="middle"
> </text>
```

### HTML file to embed the SVG and create the reference

```
<HTML>
<TITLE>Demonstration of Web Mapping with SVG</TITLE>
<HEAD>
<SCRIPT type="text/javascript" language="JavaScript1.2">
<!--
var svgSVGObj, svgplugin, textFlyOut;

function initMap()
{
    // Get a reference to the SVG-Document, within the DOM-Tree.
    // Retrieve the SVG document object:
    svgplugin = document.embeds[0].getSVGDocument(); //first svg-plugin

    //get reference to text-Element
    textFlyOut = svgplugin.getElementById("fly_out_text");

    //get reference to text within text-Element
    svgSVGObj = svgplugin.getDocumentElement();
    textFlyOut = textFlyOut.getFirstChild();
}
function showUniversity(university_id)
{
    var u_name = university_id.getAttribute('id');
    textFlyOut.setData(u_name);
}
function emptyUniversity(university_id)
{
    var u_name = "";
    textFlyOut.setData(u_name);
}
function display_layer (checkbox, element_id)
{
    var svgobj;
    var svgstyle;
    var svgdoc = document.HongKongMaps.getSVGDocument();

    // *** For each element, get the element's style object, then set
    // *** its visibility according to the state of the checkbox.
    svgobj = svgdoc.getElementById(element_id);
    svgstyle = svgobj.getStyle();

    if (!checkbox.checked)
    {
        // Hide layer.
        svgstyle.setProperty('visibility', 'hidden');
    }
    else
    {
        // Show layer.
        svgstyle.setProperty('visibility', 'visible');
    }
}
// -->
</SCRIPT>

</HEAD>
```

```

<BODY onload="initMap();">
<CENTER>
<P>
<TABLE BORDER="0" CELLPADDING="8" CELLSPACING="0">
  <TR>
    <TD COLSPAN="2" align="center">
      <H4>Move the cursor over the blue block to see the effect!</H4>
    </TD>
  </TR>
  <TR>
    <TD align="center" valign="top">
      <EMBED
        NAME="HongKongMaps"
        WIDTH="200"
        HEIGHT="200"
        SRC="43displayObjectID.svg"
        type="image/svg+xml"
      >
    </TD>
    <TD align="left" valign="top">
      <FORM NAME="display_form">
        <TABLE BORDER="0" CELLPADDING="0" CELLSPACING="2">
          <TR>
            <TD><b>Visible Layers:</b>
          </TD>
        </TR>
          <TR>
            <TD><INPUT TYPE="checkbox"
              VALUE=""
              ONCLICK="display_layer(this, 'coordinateFrameAndText') "
              >&nbsp;&nbsp;&nbsp;Outer Border
          </TD>
        </TR>
          <TR>
            <TD><INPUT TYPE="checkbox"
              VALUE=""
              ONCLICK="display_layer(this, 'Universities') "
              >&nbsp;&nbsp;&nbsp;Universities
          </TD>
        </TR>
        </TABLE>

        <SCRIPT type="text/javascript" LANGUAGE="JavaScript1.2">
        <!--// Make sure all checkboxes are checked when the page is loaded.
          for (var i = 0; i < document.display_form.elements.length; i++)
            if (document.display_form.elements[i].type == 'checkbox')
              document.display_form.elements[i].checked = true;
        // -->
        </SCRIPT>

      </FORM>
    </TD>
  </TR>
</TABLE>

</BODY>
</HTML>

```